사용 설명서

AWS CodeDeploy



API 버전 2014-10-06

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

AWS CodeDeploy: 사용 설명서

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

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

Table of Contents

CodeDeploy란 무엇인가요? 1
의 이점 AWS CodeDeploy 2
CodeDeploy 컴퓨팅 플랫폼 개요 3
CodeDeploy 배포 유형 개요 9
인 플레이스 배포 개요 11
블루/그린 배포 개요 12
연락을 기다리겠습니다. 16
주요 구성 요소 16
애플리케이션 16
컴퓨팅 플랫폼 17
배포 구성 18
배포 그룹 18
배포 유형 18
배포 유형 19
개정 20
서비스 역할 20
대상 수정 버전 20
기타 구성 요소 20
배포 21
AWS Lambda 컴퓨팅 플랫폼에 배포 21
Amazon ECS 컴퓨팅 플랫폼에서의 배포 24
EC2/온프레미스 컴퓨팅 플랫폼의 배포 35
애플리케이션 사양 파일 41
Amazon ECS 컴퓨팅 플랫폼에 대한 AppSpec 파일 42
AWS Lambda 컴퓨팅 플랫폼의 AppSpec 파일 42
EC2/온프레미스 컴퓨팅 플랫폼에 대한 AppSpec 파일 42
CodeDeploy 에이전트에서 AppSpec 파일을 사용하는 방법 43
시작하기 44
1단계: 설정 44
에 가입 AWS 계정 44
2단계: 서비스 역할 생성 45
서비스 역할 생성(콘솔) 47
서비스 역할 생성(CLI) 49
서비스 역할 ARN 확인(콘솔) 52

서비스 역할 ARN 확인(CLI) ... 52
3단계: CodeDeploy 사용자 권한 제한 ... 53
4단계: IAM 인스턴스 프로파일 만들기 ... 56
Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기(CLI) ... 57
Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기(콘솔) ... 61
제품 및 서비스 통합 ... 65
다른 AWS 서비스와의 통합 ... 65
Amazon EC2 오토 스케일링 ... 73
Elastic Load Balancing ... 80
파트너 제품 및 서비스와의 통합 ... 85
GitHub ... 89
커뮤니티의 통합 예제 ... 93
블로그 게시물 ... 93
자습서 ... 95
튜토리얼: Windows가 아닌 인스턴스에 WordPress 배포 ... 95
1단계: Amazon EC2 인스턴스 시작 ... 96
2단계: 원본 콘텐츠 구성 ... 99
3단계: 애플리케이션을 Amazon S3에 업로드 ... 104
4단계: 애플리케이션 배포 ... 108
5단계: 애플리케이션 업데이트 및 재배포 ... 114
6단계: 정리 ... 118
튜토리얼: Windows Server 인스턴스에 Hello World 애플리케이션 배포 ... 121
1단계: Amazon EC2 인스턴스 시작 ... 122
2단계: 원본 콘텐츠 구성 ... 124
3단계: 애플리케이션을 Amazon S3에 업로드 ... 127
4단계: 애플리케이션 배포 ... 132
5단계: 애플리케이션 업데이트 및 재배포 ... 137
6단계: 정리 ... 140
튜토리얼: 온프레미스 인스턴스에 애플리케이션 배포 ... 143
사전 조건 ... 144
1단계: 온프레미스 인스턴스 구성 ... 144
2단계: 샘플 애플리케이션 수정 버전 만들기 ... 144
3단계: 애플리케이션 수정 버전을 번들링하여 Amazon S3에 업로드 ... 149
4단계: 애플리케이션 수정 버전 배포 ... 149
5단계: 배포 확인 ... 150
6단계: 리소스 정리 ... 150

튜토리얼: Auto Scaling 그룹에 배포 152
사전 조건 153
1단계: Auto Scaling 그룹 생성 및 구성 153
2단계: Auto Scaling 그룹에 애플리케이션 배포 159
3단계: 결과 확인 169
4단계: Auto Scaling 그룹에서 Amazon EC2 인스턴스의 수 늘리기 171
5단계: 결과 다시 확인 172
6단계: 정리 175
튜토리얼: GitHub에서 애플리케이션 배포 177
사전 조건 178
1단계: GitHub 계정 설정 178
2단계: GitHub 리포지토리 만들기 178
3단계: GitHub 리포지토리로 샘플 애플리케이션 업로드 180
4단계: 인스턴스 프로비저닝 185
5단계: 애플리케이션 및 배포 그룹 만들기 186
6단계: 인스턴스에 애플리케이션 배포 188
7단계: 배포 모니터링 및 확인 191
8단계: 정리 193
튜토리얼: Amazon ECS에 애플리케이션 배포 194
사전 조건 196
1단계: Amazon ECS 애플리케이션 업데이트 197
2단계: AppSpec 파일 생성 198
3단계: CodeDeploy 콘솔을 사용하여 애플리케이션 배포 199
4단계: 정리 203
튜토리얼: Amazon ECS 서비스 배포 및 확인 테스트 203
사전 조건 206
1단계: 테스트 리스너 생성 206
2단계: Amazon ECS 애플리케이션 업데이트 206
3단계: 수명 주기 후크 Lambda 함수 생성 207
4단계: AppSpec 파일 업데이트 209
5단계: CodeDeploy 콘솔을 사용하여 Amazon ECS 서비스 배포 211
6단계: CloudWatch Logs에서 Lambda 후크 함수 출력 보기 213
7단계: 정리 214
자습서: AWS SAM을 사용하여 Lambda 함수 배포 215
사전 조건 216
1단계: 인프라 설정 216

2단계: Lambda 함수 업데이트 231
3단계: 업데이트된 Lambda 함수 배포 234
4단계: 배포 결과 보기 236
5단계: 정리 239
CodeDeploy 에이전트 작업 240
CodeDeploy 에이전트에서 지원하는 운영 체제 241
지원되는 Amazon EC2 AMI 운영 체제 241
지원되는 온프레미스 운영 체제 241
CodeDeploy 에이전트용 통신 프로토콜 및 포트 242
CodeDeploy 에이전트의 버전 기록 243
CodeDeploy 프로세스 관리 260
애플리케이션 수정 버전 및 로그 파일 정리 260
CodeDeploy 에이전트에 의해 설치된 파일 261
CodeDeploy 에이전트 작업 관리 264
CodeDeploy 에이전트가 실행 중인지 확인 265
CodeDeploy 에이전트의 버전 확인 268
CodeDeploy 에이전트 설치 270
CodeDeploy 에이전트 업데이트 283
CodeDeploy 에이전트 설치 제거 286
CloudWatch로 CodeDeploy 에이전트 로그 전송 287
인스턴스로 작업 293
Amazon EC2 인스턴스와 온프레미스 인스턴스 비교 293
CodeDeploy에 대한 인스턴스 작업 295
CodeDeploy 배포를 위해 인스턴스에 태그 지정 296
예 1: 한 태그 그룹, 한 태그 297
예 2: 한 태그 그룹, 여러 태그 298
예 3: 여러 태그 그룹, 한 태그 300
예 4: 여러 태그 그룹, 여러 태그 302
Amazon EC2 인스턴스 작업 306
CodeDeploy를 위한 Amazon EC2 인스턴스 생성 306
Amazon EC2 인스턴스 생성(CloudFormation 템플릿) 313
Amazon EC2 인스턴스 구성 323
온프레미스 인스턴스 작업 327
온프레미스 인스턴스 구성을 위한 사전 요구 사항 328
온프레미스 인스턴스 등록 330
온프레미스 인스턴스 작업 관리 363

인스턴스 정보 보기 ... 370
인스턴스 정보 보기(콘솔) ... 371
인스턴스 정보 보기(CLI) ... 371
인스턴스 상태 ... 372
상태 확인 ... 373
최소 정상 인스턴스 수 정보 ... 374
가용 영역당 최소 정상 인스턴스 수 정보 ... 377
배포 구성 작업 ... 380
EC2/온프레미스 컴퓨팅 플랫폼에 대한 배포 구성 ... 380
미리 정의된 배포 구성 ... 380
Amazon ECS 컴퓨팅 플랫폼에 대한 배포 구성 ... 385
Amazon ECS에 대해 미리 정의된 배포 구성 ... 385
CloudFormation 블루/그린 배포를 위한 배포 구성(Amazon ECS) ... 386
AWS Lambda 컴퓨팅 플랫폼의 배포 구성 ... 386
Lambda에 대해 미리 정의된 배포 구성 ... 387
주제 ... 387
배포 구성 만들기 ... 388
배포 구성 생성(콘솔) ... 388
배포 구성 생성(AWS CLI) ... 390
배포 구성 세부 정보 보기 ... 391
배포 구성 세부 정보 보기(콘솔) ... 392
배포 구성 보기(CLI) ... 392
배포 구성 삭제 ... 392
애플리케이션 작업 ... 394
애플리케이션 만들기 ... 395
현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔) ... 396
Blue/Green 배포를 위한 애플리케이션 생성(콘솔) ... 399
Amazon ECS 서비스 배포를 위한 애플리케이션 생성 (콘솔) ... 403
AWS Lambda 함수 배포를 위한 애플리케이션 생성(콘솔) ... 405
애플리케이션(CLI) 생성 ... 407
애플리케이션 세부 정보 보기 ... 407
애플리케이션 세부 정보 보기(콘솔) ... 407
애플리케이션 세부 정보 보기(CLI) ... 408
알림 규칙 생성 ... 408
애플리케이션 이름 바꾸기 ... 411
애플리케이션 삭제 ... 411

애플리케이션 삭제(콘솔) ... 411
애플리케이션(AWS CLI)을 삭제합니다. ... 412
배포 그룹 작업 ... 413
Amazon ECS 컴퓨팅 플랫폼 배포의 배포 그룹 ... 413
AWS Lambda 컴퓨팅 플랫폼 배포의 배포 그룹 ... 413
EC2/온프레미스 컴퓨팅 플랫폼 배포의 배포 그룹 ... 413
주제 ... 414
배포 그룹 만들기 ... 414
인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) ... 415
EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) ... 418
Amazon ECS 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) ... 423
Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정 ... 425
CodeDeploy Amazon ECS 배포를 위한 로드 밸런서, 대상 그룹 및 리스너 설정 ... 426
배포 그룹 만들기(CLI) ... 431
배포 그룹 세부 정보 보기 ... 432
배포 그룹 세부 정보 보기(콘솔) ... 432
배포 그룹 세부 정보 보기(CLI) ... 433
배포 그룹 설정 변경 ... 433
배포 그룹 설정 변경(콘솔) ... 434
배포 그룹 설정 변경(CLI) ... 435
배포 그룹에 대한 고급 옵션 구성 ... 436
배포 그룹 삭제 ... 439
배포 그룹 삭제(콘솔) ... 439
배포 그룹 삭제(CLI) ... 440
애플리케이션 개정 작업 ... 441
개정 계획 ... 441
AppSpec 파일 추가 ... 442
Amazon ECS 배포용 AppSpec 파일 추가 ... 442
AWS Lambda 배포를 위한 AppSpec 파일 추가 ... 445
EC2 온프레미스 배포용 AppSpec 파일 추가 ... 447
리포지토리 유형 선택 ... 451
개정 푸시 ... 453
를 사용하여 개정 푸시 AWS CLI ... 455
애플리케이션 개정 세부 정보 보기 ... 457
애플리케이션 개정 세부 정보 보기(콘솔) ... 457
애플리케이션 개정 세부 정보 보기(CLI) ... 458

애플리케이션 개정 등록 ... 459
CodeDeploy(CLI)를 사용하여 Amazon S3에 개정을 등록합니다. ... 460
CodeDeploy(CLI)를 사용하여 GitHub에 개정 등록 ... 461
배포 작업 ... 462
배포 만들기 ... 463
배포 사전 조건 ... 464
Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(콘솔) ... 466
AWS Lambda 컴퓨팅 플랫폼 배포 생성(콘솔) ... 468
EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔) ... 470
Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(CLI) ... 474
AWS Lambda 컴퓨팅 플랫폼 배포 생성(CLI) ... 475
EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(CLI) ... 476
를 통해 Amazon ECS 블루/그린 배포 생성 CloudFormation ... 480
배포 세부 정보 보기 ... 483
배포 세부 정보 보기(콘솔) ... 484
배포 세부 정보 보기(CLI) ... 485
배포 로그 데이터 보기 ... 485
Amazon CloudWatch 콘솔에서 로그 파일 데이터 보기 ... 486
인스턴스의 로그 파일 보기 ... 486
배포 중지 ... 489
배포 중지(콘솔) ... 490
배포 중지(CLI) ... 491
재배포 및 배포 롤백 ... 491
자동 롤백 ... 491
수동 롤백 ... 492
롤백 및 재배포 워크플로 ... 492
기존 컨텐츠의 롤백 동작 ... 493
다른 AWS 계정에 애플리케이션 배포 ... 496
1단계: 두 계정 중 하나에 S3 버킷 생성 ... 496
2단계: Amazon S3 버킷에 프로덕션 계정의 IAM 인스턴스 프로필에 대한 권한 부여 ... 496
3단계: 프로덕션 계정에서 리소스 및 교차 계정 역할 생성 ... 498
4단계: Amazon S3 버킷에 애플리케이션 개정 버전 업로드. ... 499
5단계: 교차 계정 역할을 위임하고 애플리케이션 배포 ... 499
로컬 시스템에서 배포 패키지 유효성 검사 ... 500
사전 조건 ... 500
로컬 배포 생성 ... 503

예제 ... 506
배포 모니터링 ... 508
자동 도구 ... 508
수동 도구 ... 510
Amazon CloudWatch 도구를 사용하여 배포 모니터링 ... 511
CloudWatch 경보를 사용하여 배포 모니터링 ... 511
Amazon CloudWatch Events를 사용하여 배포 모니터링 ... 513
를 사용하여 배포 모니터링 AWS CloudTrail ... 515
CloudTrail의 CodeDeploy 정보 ... 515
CodeDeploy 로그 파일 항목 이해 ... 516
Amazon SNS 이벤트 알림으로 배포 모니터링 ... 518
서비스 역할에 Amazon SNS 권한 부여 ... 519
CodeDeploy 이벤트에 대한 트리거 만들기 ... 520
배포 그룹에서 트리거 편집 ... 527
배포 그룹에서 트리거 삭제 ... 529
트리거의 JSON 데이터 형식 ... 530
보안 ... 532
데이터 보호 ... 532
인터네트워크 트래픽 개인 정보 ... 533
저장 데이터 암호화 ... 534
전송 중 암호화 ... 534
암호화 키 관리 ... 534
ID 및 액세스 관리 ... 534
대상 ... 535
ID를 통한 인증 ... 535
정책을 사용하여 액세스 관리 ... 536
AWS CodeDeploy 에서 IAM을 사용하는 방법 ... 537
AWS CodeDeploy에 대한 관리형(미리 정의된) 정책 ... 542
AWS 관리형 정책에 대한 CodeDeploy 업데이트 ... 550
ID 기반 정책 예시 ... 554
문제 해결 ... 561
CodeDeploy 권한 참조 ... 562
교차 서비스 혼동된 대리인 방지 ... 571
인시던트 대응 ... 573
CodeDeploy와의 모든 상호 작용에 대한 감사 ... 573
알림 및 인시던트 관리 ... 573

규정 준수 확인 574
복원력 574
인프라 보안 575
레퍼런스 576
AppSpec 파일 참조 576
Amazon ECS 컴퓨팅 플랫폼에 대한 AppSpec 파일 577
AWS Lambda 컴퓨팅 플랫폼의 AppSpec 파일 577
EC2/온프레미스 컴퓨팅 플랫폼에 대한 AppSpec 파일 577
AppSpec 파일 구조 578
AppSpec 파일 예 623
AppSpec 파일 간격 629
AppSpec 파일 및 파일 위치 확인 630
에이전트 구성 참조 631
관련 주제 640
CloudFormation 템플릿 참조 641
Amazon Virtual Private Cloud와 함께 CodeDeploy 사용 644
가용성 644
CodeDeploy에 대한 VPC 엔드포인트 생성 647
CodeDeploy 에이전트 및 IAM 권한 구성 648
리소스 키트 참조 649
리전별 리소스 키트 버킷 이름 649
리소스 키트 콘텐츠 651
리소스 키트 파일 목록 표시 653
리소스 키트 파일 다운로드 655
할당량 658
문제 해결 664
이 시나리오에서는 플릿에 상태에서 정상 인스턴스가 표시되지만 위의 오류도 표시됩니다. 664
일반적인 문제 해결 체크리스트 665
CodeDeploy 배포 리소스는 일부 AWS 리전에서만 지원됩니다. 666
이 설명서의 절차가 CodeDeploy 콘솔과 일치하지 않음 667
필요한 IAM 역할을 사용할 수 없음 667
일부 텍스트 편집기를 사용하여 AppSpec 파일 및 쉘 스크립트를 작성하면 배포에 실패할 수 있음 667
macOS에서 Finder를 사용하여 애플리케이션 수정을 번들링하면 배포에 실패할 수 있음 668
EC2/온프레미스 배포 문제 해결 668
CodeDeploy 플러그 인 CommandPoller 자격 증명 누락 오류 669

"Validation of PKCS7 signed message failed" 메시지와 함께 배포에 실패합니다. 670
동일한 파일을 같은 인스턴스 위치로 배포 또는 다시 배포하려고 하면 실패하고 오류 "The deployment failed because a specified file already exists at this location"이 표시됨 670
파일 경로가 길면 “No such file or directory(해당 파일 또는 디렉터리가 없음)” 오류가 발생함 672
오래 실행되는 프로세스로 인해 배포에 실패할 수 있음 673
배포 로그에 오류를 보고하지 않은 실패한 AllowTraffic 수명 주기 이벤트 문제 해결 674
실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결 675
실패하고 UnknownError: not opened for reading이 표시되는 DownloadBundle 배포 수명 주기 이벤트 문제 해결 676
모든 수명 주기 이벤트를 건너뛰었을 때 문제 해결 677
Windows PowerShell 스크립트에서 기본적으로 Windows PowerShell 64비트 버전을 사용하지 못함 679
Amazon ECS 배포 문제 해결 680
대체 작업 세트를 기다리는 동안 시간 초과 발생 680
알림이 계속되기를 기다리는 동안 시간 초과 발생 681
IAM 역할에 충분한 권한이 없음 681
상태 콜백을 기다리는 동안 배포 시간 초과 682
하나 이상의 수명 주기 이벤트 검증 함수가 실패하여 배포 실패 683
다음 오류로 인해 ELB를 업데이트할 수 없음: 기본 작업 세트 대상 그룹은 리스너 뒤에 있어야 함 683
Auto Scaling을 사용할 때 때때로 배포가 실패함 684
ALB만 점진적인 트래픽 라우팅을 지원하므로 배포 그룹을 생성/업데이트할 때는 대신 AllAtOnce 트래픽 라우팅을 대신 사용해야 함 685
배포에 성공했는데도 대체 작업 세트가 Elastic Load Balancing 상태 확인에 실패하고 애플리케이션이 다운됨 685
배포 그룹에 여러 로드 밸런서를 연결할 수 있나요? 686
로드 밸런서 없이 CodeDeploy 블루/그린 배포를 수행할 수 있나요? 686
배포 중에 새 정보로 Amazon ECS 서비스를 업데이트하려면 어떻게 해야 하나요? 687
AWS Lambda 배포 문제 해결 687
AWS Lambda 롤백이 구성되지 않은 Lambda 배포를 수동으로 중지한 후 배포가 실패함 687
배포 그룹 관련 문제 해결 688
인스턴스에 배포 그룹의 일부로 태그를 지정해도 애플리케이션이 새 인스턴스로 자동으로 배포되지 않음 688
인스턴스 문제 해결 688

태그를 올바르게 설정해야 함 688
AWS CodeDeploy 인스턴스에 에이전트를 설치하고 실행해야 합니다. 689
배포 중 인스턴스가 종료되면 최대 1시간 동안 배포에 실패하지 않음 689
인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석 689
실수로 삭제된 경우 새 CodeDeploy 로그 파일 만들기 690
"InvalidSignatureException – 서명 만료: [time]이 이제 [time]보다 이전" 배포 오류 문제 해결 690
GitHub 토큰 문제 해결 690
유효하지 않은 GitHub OAuth 토큰 690
GitHub OAuth 토큰 최대 수 초과 691
Amazon EC2 Auto Scaling 문제 해결 691
일반적인 Amazon EC2 Auto Scaling 문제 해결 692
“CodeDeployRole은 AmazonAutoScaling” 오류 AWS 서비스에서 작업을 수행할 수 있는 권한을 부여하지 않습니다. 693
개정 버전을 배포하기 전에 Amazon EC2 Auto Scaling 그룹의 인스턴스가 계속해서 프로비저닝 및 종료됨 693
Amazon EC2 Auto Scaling 인스턴스를 종료 또는 재부팅하면 배포에 실패할 수 있음 694
단일 Amazon EC2 Auto Scaling 그룹으로 여러 배포 그룹 연결 피하기 695
Amazon EC2 Auto Scaling 그룹의 EC2 인스턴스를 시작하지 못하고 "하트비트 제한 시간” 오류 발생 695
일치하지 않는 Amazon EC2 Auto Scaling 수명 주기 후크로 인해 Amazon EC2 Auto Scaling 그룹에 대한 자동 배포가 중지되거나 실패할 수 있습니다. 698
“배포 그룹에 대한 인스턴스를 찾을 수 없어서 배포에 실패했습니다.” 오류 699
오류 코드 707
관련 주제 710
리소스 711
참조 가이드 및 지원 리소스 711
샘플 711
블로그 711
AWS 소프트웨어 개발 키트 및 도구 711
문서 기록 713
이전 업데이트 729
AWS 용어집 748
..... dccxlix

CodeDeploy란 무엇인가요?

CodeDeploy는 Amazon EC2 인스턴스, 온프레미스 인스턴스, 서버리스 Lambda 함수 또는 Amazon ECS 서비스로 애플리케이션 배포를 자동화하는 배포 서비스입니다.

다음을 포함하여 다양한 애플리케이션 콘텐츠를 거의 무제한으로 배포할 수 있습니다.

- 코드
- 서버리스 AWS Lambda 함수
- 웹 및 구성 파일
- Executables
- Packages
- 스크립트
- 멀티미디어 파일

CodeDeploy는 서버에서 실행되고 Amazon S3 버킷, GitHub 리포지토리 또는 Bitbucket 리포지토리에 저장되는 애플리케이션 콘텐츠를 배포할 수 있습니다. 또한 CodeDeploy는 서버리스 Lambda 함수를 배포할 수 있습니다. CodeDeploy를 사용하기 위해 기존 코드를 변경할 필요가 없습니다.

CodeDeploy를 사용하면 다음 작업을 쉽게 수행할 수 있습니다.

- 새 기능을 신속하게 출시.
- AWS Lambda 함수 버전을 업데이트합니다.
- 애플리케이션 배포 시 가동 중지 방지
- 오류가 발생하는 수동 배포와 관련된 다양한 위험 없이 애플리케이션 업데이트에 따른 복잡성 처리.

이 서비스는 인프라와 함께 규모를 조정할 수 있으므로 인스턴스 하나 또는 수천 개에 쉽게 배포할 수 있습니다.

CodeDeploy는 구성 관리, 소스 제어, 지속적인 통합, 지속적인 전송 및 지속적인 배포 등을 위해 다양한 시스템과 함께 작동합니다. 자세한 내용은 제품 통합을 참조하세요.

CodeDeploy 콘솔에서 리포지토리, 구축 프로젝트, 배포 애플리케이션 및 파이프라인과 같은 리소스를 신속하게 검색할 수도 있습니다. 리소스로 이동을 선택하거나 / 키를 누른 후 리소스 이름을 입력하세요. 목록에 일치 항목이 나타납니다. 검색은 대/소문자를 구분하지 않습니다. 보기 권한이 있는 리소스

만 표시됩니다. 자세한 내용은 에 대한 자격 증명 및 액세스 관리 AWS CodeDeploy 단원을 참조하십시오.

주제

- 의 이점 AWS CodeDeploy
- CodeDeploy 컴퓨팅 플랫폼 개요
- CodeDeploy 배포 유형 개요
- 연락을 기다리겠습니다.
- CodeDeploy 주요 구성 요소
- CodeDeploy 배포
- CodeDeploy 애플리케이션 사양(AppSpec) 파일

의 이점 AWS CodeDeploy

CodeDeploy는 다음 이점을 제공합니다

- 서버, 서버리스 및 컨테이너 애플리케이션. CodeDeploy를 사용하면 서버리스 AWS Lambda 함수 버전 또는 Amazon ECS 애플리케이션을 배포하는 애플리케이션과 서버에 기존 애플리케이션을 모두 배포할 수 있습니다.
- 배포 자동화. CodeDeploy는 개발, 테스트 및 프로덕션 환경에 걸쳐 애플리케이션 배포를 완전 자동화합니다. 그리고 CodeDeploy는 인프라에 맞춰 규모를 조정할 수 있으므로 인스턴스 하나 또는 수천 개에 배포할 수 있습니다.
- 가동 중지 최소화. 애플리케이션이 EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 경우 CodeDeploy는 애플리케이션 가용성을 극대화합니다. 인 플레이스(In-place) 배포에서 CodeDeploy는 Amazon EC2 인스턴스 전체에 대해 롤링 업데이트를 수행합니다. 업데이트 시 오프라인 상태가 될 수 있는 인스턴스 수를 지정할 수 있습니다. 블루/그린 배포 시에는 최신 애플리케이션 수정이 대체 인스턴스에 설치됩니다. 선택한 경우 새로운 환경 테스트를 완료한 직후 이러한 인스턴스로 트래픽이 다시 라우팅됩니다. 두 가지 배포 유형에 대해 CodeDeploy는 사용자가 구성한 규칙에 따라 애플리케이션 상태를 추적합니다.
- 중지 및 롤백. 오류가 있는 경우 자동 또는 수동으로 배포를 중지하고 롤백할 수 있습니다.
- 중앙 집중식 제어. CodeDeploy 콘솔 또는 AWS CLI를 통해 배포 상태를 시작 및 추적할 수 있습니다. 각 애플리케이션 개정이 배포된 시점 및 Amazon EC2 인스턴스가 나열된 보고서가 제공됩니다.

- 채택 편의성. CodeDeploy는 플랫폼과 관련된 제약이 없으므로 모든 애플리케이션과 작동합니다. 사용자는 설정 코드를 쉽게 재사용할 수 있습니다. 또한 CodeDeploy는 소프트웨어 릴리스 프로세스 또는 지속적 전달 도구 체인과 통합이 가능합니다.
- 동시 배포. EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 1개 이상의 애플리케이션이 있는 경우에는 CodeDeploy를 통해 동일한 인스턴스 세트에 동시에 배포할 수 있습니다.

CodeDeploy 컴퓨팅 플랫폼 개요

CodeDeploy는 세 가지 컴퓨팅 플랫폼에 애플리케이션을 배포할 수 있습니다.

- EC2/온프레미스: 물리적 서버의 인스턴스를 설명합니다. Amazon EC2 클라우드 인스턴스나 온프레미스 서버 또는 둘 다일 수 있습니다. EC2/온프레미스 컴퓨팅 플랫폼을 사용하여 만든 애플리케이션은 실행 파일과 구성 파일, 이미지 및 기타 항목으로 구성될 수 있습니다.

  EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포에서는 인 플레이스 또는 블루/그린 배포 유형을 사용하여 인스턴스로 트래픽이 전송되는 방식을 관리합니다. 자세한 내용은 CodeDeploy 배포 유형 개요 단원을 참조하십시오.
- AWS Lambda: Lambda 함수의 업데이트된 버전으로 구성된 애플리케이션을 배포하는 데 사용됩니다.는 고가용성 컴퓨팅 구조로 구성된 서버리스 컴퓨팅 환경에서 Lambda 함수를 AWS Lambda 관리합니다. 컴퓨팅 리소스의 모든 관리는에서 수행합니다 AWS Lambda. 자세한 정보는 서버리스 컴퓨팅 및 애플리케이션을 참조하세요. AWS Lambda 및 Lambda 함수에 대한 자세한 내용은 섹션을 참조하세요AWS Lambda.

  카나리(Canary), 선형(Linear) 또는 한 번에 모두(All-at-once) 구성을 선택하여 업데이트된 Lambda 함수 버전으로 트래픽을 전송하는 방식을 관리할 수 있습니다.
- Amazon ECS: Amazon ECS 컨테이너화된 애플리케이션을 작업 세트로 배포하는 데 사용됩니다. CodeDeploy는 애플리케이션의 업데이트 버전을 새로운 대체 작업 세트로 설치하여 블루/그린 배포를 수행합니다. CodeDeploy는 프로덕션 트래픽을 원래 애플리케이션 작업 세트에서 대체 작업 세트로 다시 라우팅합니다. 배포가 성공하면 기존 작업 세트는 종료됩니다. Amazon ECS에 대한 자세한 내용은 Amazon Elastic Container Service를 참조하세요.

  카나리(Canary), 선형(Linear) 또는 한번에 모두(All-at-once) 구성을 선택하여 배포 중 업데이트된 작업 세트로 트래픽을 전송하는 방식을 관리할 수 있습니다.

> **Note**
>
> Amazon ECS 블루/그린 배포는 CodeDeploy 및 CloudFormation모두를 통해 지원됩니다. 이러한 배포에 대한 세부 정보는 다음 단원에서 설명합니다.

다음 표에는 CodeDeploy 구성 요소가 각 컴퓨팅 플랫폼에서 어떻게 사용되는지 나와 있습니다. 자세한 내용은 다음을 참조하세요.

- CodeDeploy에서 배포 그룹 작업
- CodeDeploy에서 배포 작업
- CodeDeploy에서 배포 구성 작업
- CodeDeploy의 애플리케이션 개정 작업
- CodeDeploy의 애플리케이션 작업

CodeDeploy 구성 요소	EC2/온프레미스	AWS Lambda	Amazon ECS
배포 그룹	인스턴스 세트에 수정을 배포합니다.	고가용성 컴퓨팅 인프라에서 새 버전의 서버리스 Lambda 함수를 배포합니다.	작업 세트로 배포할 컨테이너화된 애플리케이션이 있는 Amazon ECS 서비스, 배포된 애플리케이션에 트래픽을 제공하는 데 사용되는 프로덕션 및 테스트 리스너(선택

CodeDeploy 구성 요소	EC2/온프레미스	AWS Lambda	Amazon ECS
			사항), 트래픽을 다시 라우팅하고 배포된 애플리케이션의 원래 작업 세트를 종료할 시기, 트리거(선택 사항), 경보, 롤백 설정을 지정합니다.

CodeDeploy 구성 요소	EC2/온프레미스	AWS Lambda	Amazon ECS
배포	애플리케이션과 AppSpec 파일로 구성된 새로운 수정을 배포합니다. AppSpec은 배포 그룹의 인스턴스에 애플리케이션을 배포하는 방식을 지정합니다.	Lambda 함수의 한 버전에서 동일한 함수의 새 버전으로 프로덕션 트래픽을 전환합니다. AppSpec 파일은 배포할 Lambda 함수 버전을 지정합니다.	Amazon ECS 컨테이너화된 애플리케이션의 업데이트된 버전을 새로운 대체 작업 세트로 배포합니다. CodeDeploy는 프로덕션 트래픽을 원래 버전의 작업 세트에서 업데이트된 버전의 대체 작업 세트로 다시 라우팅합니다. 배포가 완료되면 원래 작업 세트가 종료됩니다.

CodeDeploy 구성 요소	EC2/온프레미스	AWS Lambda	Amazon ECS
배포 구성	배포 시 항상 정상 상태를 유지해야 하는 최소 인스턴스 수와 배포 속도를 정의하는 설정입니다.	업데이트된 Lambda 함수 버전으로 트래픽이 이동되는 방식을 정의하는 설정입니다.	업데이트된 Amazon ECS 작업 세트로 트래픽이 이동되는 방식을 정의하는 설정입니다.

CodeDeploy 구성 요소	EC2/온프레미스	AWS Lambda	Amazon ECS
개정	AppSpec 파일과 애플리케이션 파일의 조합입니다(실행 파일, 구성 파일 등).	배포할 Lambda 함수와 배포 수명 주기 이벤트 후크 중에 검증 테스트를 실행할 수 있는 Lambda 함수를 지정하는 AppSpec 파일입니다.	다음을 지정하는 AppSpec 파일입니다. • 배포할 컨테이너화된 애플리케이션이 있는 Amazon ECS 서비스에 대한 Amazon ECS 작업 정의 • 업데이트된 애플리케이션이 배포되는 컨테이너 • 프로덕션 트래픽이 다시 라우팅되는 컨테이너의 포트 • 배포 수명 주기

CodeDeploy 구성 요소	EC2/온프레미스	AWS Lambda	Amazon ECS
			이벤트 후크 중에 확인 테스트를 실행할 수 있는 Lambda 함수와 네트워크 구성 설정(선택 사항).
애플리케이션	배포 그룹과 수정 모음입니다. EC2/온프레미스 애플리케이션은 EC2/온프레미스 컴퓨팅 플랫폼을 사용합니다.	배포 그룹과 수정 모음입니다. AWS Lambda 배포에 사용되는 애플리케이션은 서버리스 AWS Lambda 컴퓨팅 플랫폼을 사용합니다.	배포 그룹과 수정 모음입니다. Amazon ECS 배포에 사용되는 애플리케이션은 Amazon ECS 컴퓨팅 플랫폼을 사용합니다.

CodeDeploy 배포 유형 개요

CodeDeploy는 두 가지 배포 유형 옵션을 제공합니다.

- 실행 중 배포: 배포 그룹의 각 인스턴스에 있는 애플리케이션이 중지되고 최신 애플리케이션 수정 버전이 설치되며 애플리케이션의 새 버전이 시작되고 유효성이 검사됩니다. 로드 밸런서를 사용하면 배포가 진행될 때 각 인스턴스를 등록 취소한 후 배포가 완료된 후 서비스로 복원할 수 있습니다.

EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포만 인 플레이스 배포를 사용할 수 있습니다. 현재 위치 배포에 대한 자세한 내용은 인 플레이스 배포 개요 단원을 참조하세요.

> **Note**
>
> AWS Lambda 및 Amazon ECS 배포는 현재 위치 배포 유형을 사용할 수 없습니다.

- 블루/그린 배포: 배포 동작은 사용하는 컴퓨팅 플랫폼에 따라 다릅니다.
  - EC2/온프레미스 컴퓨팅 플랫폼에서의 블루/그린 배포: 배포 그룹(원래 환경)의 인스턴스가 다음 단계를 거쳐 인스턴스의 다른 집합(대체 환경)으로 대체됩니다.
    - 인스턴스는 대체 환경을 위해 프로비저닝됩니다.
    - 최신 애플리케이션 수정은 대체 인스턴스에 설치됩니다.
    - 애플리케이션 테스트 및 시스템 검증과 같은 활동에 선택적 대기 시간이 발생합니다.
    - 대체 환경의 인스턴스가 하나 이상의 Elastic Load Balancing 로드 밸런서에 등록되고 트래픽이 이러한 인스턴스로 라우팅됩니다. 원래 환경의 인스턴스는 등록이 취소되고 종료되거나 다른 용도로 계속 실행될 수 있습니다.

    > **Note**
    >
    > EC2/온프레미스 컴퓨팅 플랫폼을 사용할 경우 블루/그린 배포는 Amazon EC2 인스턴스에서만 작동합니다.

  - AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼의 블루/그린: 트래픽은 canary, 선형 또는 all-at-once 배포 구성에 따라 증분으로 이동합니다.
  - 블루/그린 배포: CloudFormation CloudFormation 스택 업데이트의 일부로 트래픽이 현재 리소스에서 업데이트된 리소스로 이동합니다. 현재는 ECS 블루/그린 배포만 지원됩니다.

블루/그린 배포에 대한 자세한 내용은 블루/그린 배포 개요 섹션을 참조하세요.

> **Note**
>
> CodeDeploy 에이전트를 사용하면 애플리케이션, 배포 그룹 또는 AWS 계정 없이도 로그인한 인스턴스에서 배포를 수행할 수 있습니다. 자세한 내용은 CodeDeploy 에이전트를 사용하여 로컬 시스템에서 배포 패키지의 유효성을 검사합니다. 단원을 참조하세요.

주제

- 인 플레이스 배포 개요
- 블루/그린 배포 개요

인 플레이스 배포 개요

> **Note**
>
> AWS Lambda 및 Amazon ECS 배포는 현재 위치 배포 유형을 사용할 수 없습니다.

인플레이스 배포의 작동 방식은 다음과 같습니다.

1. 먼저, 로컬 개발 머신 또는 유사한 환경에서 배포 가능한 콘텐츠를 만든 다음 application specification file(AppSpec 파일)을 추가합니다. AppSpec 파일은 CodeDeploy에 고유합니다. 또한 CodeDeploy가 실행하게 하려는 배포 작업을 정의합니다. 배포 가능한 콘텐츠 및 AppSpec 파일을 아카이브 파일로 번들링한 다음 Amazon S3 버킷 또는 GitHub 리포지토리로 업로드합니다. 이러한 아카이브 파일을 애플리케이션 수정(또는 간단하게 수정)이라고 합니다.
2. 다음으로, CodeDeploy에 배포에 관한 정보를 제공합니다(예: 개정을 풀링하려는 Amazon S3 버킷 또는 GitHub 리포지토리 및 개정의 내용을 배포하려는 Amazon EC2 인스턴스 세트). CodeDeploy에서는 Amazon EC2 인스턴스 세트를 배포 그룹이라고 합니다. 배포 그룹에는 개별적으로 태그가 지정된 Amazon EC2 인스턴스, Amazon EC2 Auto Scaling 그룹의 Amazon EC2 인스턴스 또는 둘 다가 포함됩니다.

   배포 그룹에 배포하려는 새 애플리케이션 수정을 성공적으로 업로드할 때마다 번들이 배포 그룹의 대상 수정으로 설정됩니다. 다시 말해 현재 배포의 대상으로 지정된 애플리케이션 수정이 대상 수정입니다. 또한 이 수정은 자동 배포를 위해 풀링되는 수정입니다.
3. 다음으로, 각 인스턴스의 CodeDeploy 에이전트가 CodeDeploy를 폴링하여 지정된 Amazon S3 버킷 또는 GitHub 리포지토리에서 풀링할 항목 및 시점을 결정합니다.
4. 마지막으로, 각 인스턴스의 CodeDeploy 에이전트가 Amazon S3 버킷 또는 GitHub 리포지토리에서 대상 개정을 풀링하고 AppSpec 파일의 지침을 따르면 콘텐츠가 인스턴스에 배포됩니다.

CodeDeploy는 배포에 대한 레코드를 유지하므로 배포 상태, 배포 구성 파라미터, 인스턴스 상태를 확인할 수 있습니다.

블루/그린 배포 개요

블루/그린 배포는 새 애플리케이션 버전의 변경으로 인한 중단을 최소화하면서 애플리케이션을 업데이트하는 데 사용됩니다. CodeDeploy는 프로덕션 트래픽을 다시 라우팅하기 전에 이전 버전과 함께 새 애플리케이션 버전을 프로비저닝합니다.

- AWS Lambda: 트래픽이 Lambda 함수의 한 버전에서 동일한 Lambda 함수의 새 버전으로 전환됩니다.
- Amazon ECS: 트래픽이 Amazon ECS 서비스의 작업 세트에서 동일한 Amazon ECS 서비스의 업데이트된 대체 작업 세트로 전환됩니다.
- EC2/온프레미스: 트래픽이 원래 환경의 한 인스턴스 세트에서 대체 인스턴스 세트로 전환됩니다.

모든 AWS Lambda 및 Amazon ECS 배포는 블루/그린입니다. EC2/온프레미스 배포는 인플레이스 또는 블루/그린일 수 있습니다. 블루/그린 배포는 인플레이스(in-place) 배포보다 많은 이점을 제공합니다.

- 애플리케이션을 새 대체 환경에서 설치 및 테스트하고 트래픽을 다시 라우팅하여 프로덕션에 간단히 배포할 수 있습니다.
- EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 경우, 최신 버전의 애플리케이션으로 다시 전환하는 것이 더 빠르고 신뢰할 수 있습니다. 원래 인스턴스가 종료되지 않은 한 트래픽이 원래 인스턴스로 다시 라우팅될 수 있기 때문입니다. 인 플레이스(in-place) 배포의 경우, 애플리케이션의 이전 버전을 다시 배포하여 버전을 롤백해야 합니다.
- EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 경우 블루/그린 배포에 대해 새로운 인스턴스가 프로비저닝되고, 최신 서버 구성이 반영됩니다. 따라서 장기 실행 인스턴스에서 발생할 수 있는 다양한 문제를 예방할 수 있습니다.
- AWS Lambda 컴퓨팅 플랫폼을 사용하는 경우 원래 AWS Lambda 함수 버전에서 새 AWS Lambda 함수 버전으로 트래픽이 이동하는 방식을 제어합니다.
- Amazon ECS 컴퓨팅 플랫폼을 사용하는 경우 기존 작업 세트에서 새 작업 세트로 트래픽을 어떻게 이동할지 제어할 수 있습니다.

를 사용한 블루/그린 배포는 다음 방법 중 하나를 사용할 CloudFormation 수 있습니다.

- CloudFormation 배포용 템플릿: CloudFormation 템플릿을 사용하여 배포를 구성하면 CloudFormation 업데이트에 의해 배포가 트리거됩니다. 리소스를 변경하고 템플릿 변경을 업로드하면의 스택 업데이트가 새 배포를 CloudFormation 시작합니다. CloudFormation 템플릿에서 사용할 수 있는 리소스 목록은 섹션을 참조하세요CloudFormation CodeDeploy 참조용 템플릿.

- 를 통한 블루/그린 배포 CloudFormation: CloudFormation 를 사용하여 스택 업데이트를 통해 블루/그린 배포를 관리할 수 있습니다. 스택 템플릿 내에서 트래픽 라우팅 및 안정화 설정을 지정하는 것 외에도 블루 및 그린 리소스를 모두 정의합니다. 그런 다음 스택 업데이트 중에 선택한 리소스를 업데이트하면가 필요한 모든 그린 리소스를 CloudFormation 생성하고 지정된 트래픽 라우팅 파라미터를 기반으로 트래픽을 이동하며 블루 리소스를 삭제합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 CloudFormation을 사용하여 CodeDeploy를 통한 Amazon ECS 블루/그린 배포 자동화를 참조하세요.

> **Note**
>
> Amazon ECS 블루/그린 배포에만 지원됩니다.

블루/그린 배포를 어떻게 구성하는지는 배포에서 사용하는 컴퓨팅 플랫폼에 따라 다릅니다.

AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼에 블루/그린 배포

AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼을 사용하는 경우 트래픽이 원래 AWS Lambda 함수 또는 Amazon ECS 작업 세트에서 새 함수 또는 작업 세트로 어떻게 전환되는지 표시해야 합니다. 트래픽 이동 방식을 지정하려면 다음 배포 구성 중 하나를 지정해야 합니다.

- 카나리(Canary)
- 리니어(Linear)
- 한 번에 모두(All-at-once)

카나리(Canary), 리니어(Linear) 또는 한 번에 모두(All-at-once) 배포 구성에서 트래픽이 이동하는 방식에 대한 자세한 내용은 배포 구성 섹션을 참조하세요.

Lambda 배포 구성에 대한 자세한 내용은 AWS Lambda 컴퓨팅 플랫폼의 배포 구성 섹션을 참조하세요.

Amazon ECS 배포 구성에 대한 자세한 내용은 Amazon ECS 컴퓨팅 플랫폼에 대한 배포 구성 섹션을 참조하세요.

EC2/온프레미스 컴퓨팅 플랫폼의 블루/그린 배포

> **Note**
>
> EC2/온프레미스 컴퓨팅 플랫폼에서의 블루/그린 배포에는 Amazon EC2 인스턴스를 사용해야 합니다. 온프레미스 인스턴스는 블루/그린 배포 유형을 지원하지 않습니다.

EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 경우 다음 사항이 적용됩니다.

Amazon EC2 태그를 식별하는 하나 이상의 Amazon EC2 인스턴스 또는 Amazon EC2 Auto Scaling 그룹이 있어야 합니다. 이러한 인스턴스는 다음 추가 요구 사항을 충족해야 합니다.

- 각 Amazon EC2 인스턴스에 올바른 IAM 인스턴스 프로파일이 연결되어 있어야 합니다.
- 각 인스턴스에 CodeDeploy 에이전트가 설치되어 실행 중이어야 합니다.

> **Note**
>
> 또한 일반적으로 기존 환경의 인스턴스에서 실행되는 애플리케이션 수정이 있지만, 블루/그린 배포의 경우 반드시 이럴 필요는 없습니다.

블루/그린 배포에서 사용되는 배포 그룹을 만들 때 다음과 같은 대체 환경 지정 방법을 선택할 수 있습니다.

기존 Amazon EC2 Auto Scaling 그룹 복사: 블루/그린 배포 시 CodeDeploy는 배포 중 대체 환경에 필요한 인스턴스를 만듭니다. 이 옵션을 선택하면 CodeDeploy는 사용자가 지정한 Amazon EC2 Auto Scaling 그룹을 대체 환경의 템플릿(동일한 수의 실행 인스턴스 및 여러 가지 기타 구성 옵션을 포함)으로 사용합니다.

수동으로 인스턴스 선택: Amazon EC2 인스턴스 태그, Amazon EC2 Auto Scaling 그룹 이름 또는 둘 다를 사용하여 인스턴스를 대체 인스턴스로 계산하도록 지정할 수 있습니다. 이 옵션을 선택하면 배포를 만들 때까지 대체 환경을 위한 인스턴스를 지정할 필요가 없습니다.

작동 방식은 다음과 같습니다.

1. 원본 환경으로 작동할 인스턴스 또는 Amazon EC2 Auto Scaling 그룹이 이미 있습니다. 블루/그린 배포를 처음 실행할 때는 일반적으로 인 플레이스(in-place) 배포에서 이미 사용된 인스턴스를 사용합니다.

2. 기존 CodeDeploy 애플리케이션에서는 인 플레이스 배포에 필요한 옵션 이외에 블루/그린 배포 그룹을 만들어 다음을 지정합니다.
   - 블루/그린 배포 프로세스 중 원래 환경에서 대체 환경으로 트래픽을 라우팅할 로드 밸런서
   - 트래픽을 대체 환경으로 즉시 다시 라우팅하거나 수동으로 다시 라우팅할 때까지 대기할지 여부
   - 트래픽이 대체 인스턴스로 라우팅되는 속도
   - 대체된 인스턴스를 종료 또는 계속 실행할지 여부
3. 다음과 같은 이벤트가 발생하는 동안 배포 그룹에 대한 배포를 만듭니다.
   a. Amazon EC2 Auto Scaling 그룹을 복사하도록 선택한 경우 대체 환경에 필요한 인스턴스가 프로비저닝됩니다.
   b. 배포 대상으로 지정한 애플리케이션 수정이 대체 인스턴스에 설치됩니다.
   c. 배포 그룹 설정에서 대기 시간을 지정한 경우 배포가 일시 중지됩니다. 이러한 대기 시간에 대체 환경에서 테스트 및 확인을 실행할 수 있습니다. 대기 시간 종료 전 트래픽을 수동으로 다시 라우팅하지 않으면 배포가 중지됩니다.
   d. 대체 환경의 인스턴스가 Elastic Load Balancing 로드 밸런서에 등록되고 트래픽이 이러한 인스턴스로 라우팅되기 시작합니다.
   e. 원본 환경의 인스턴스는 등록 취소되어 배포 그룹의 사양에 따라 처리됩니다. 즉, 종료되거나 계속 실행됩니다.

를 통한 블루/그린 배포 CloudFormation

템플릿을 사용하여 리소스를 모델링하여 CodeDeploy 블루/그린 배포를 CloudFormation 관리할 수 있습니다.

CloudFormation 템플릿을 사용하여 블루/그린 리소스를 모델링할 때에서 작업 세트를 업데이트 CloudFormation 하는 스택 업데이트를 생성합니다. 프로덕션 트래픽은 선형 배포 및 베이킹 시간 또는 Canary 배포를 사용하여 서비스의 원래 작업 세트에서 대체 작업 세트로 한 번에 모두 이동합니다. 스택 업데이트가 CodeDeploy에서 배포를 시작합니다. CodeDeploy에서 배포 상태 및 기록을 볼 수 있지만 CloudFormation 템플릿 외부에서 CodeDeploy 리소스를 생성하거나 관리하지는 않습니다.

> **Note**
>
> 를 통한 블루/그린 배포의 CloudFormation경우 CodeDeploy 애플리케이션 또는 배포 그룹을 생성하지 않습니다.

이 방법은Amazon ECS 블루/그린 배포만 지원합니다. 를 통한 블루/그린 배포에 대한 자세한 내용은 섹션을 CloudFormation참조하세요를 통해 Amazon ECS 블루/그린 배포 생성 CloudFormation.

연락을 기다리겠습니다.

우리는 여러분의 의견을 환영합니다. 문의 사항이 있는 경우 CodeDeploy 포럼을 방문하세요.

주제

- Primary Components
- Deployments
- Application Specification Files

CodeDeploy 주요 구성 요소

서비스 작업을 시작하기 전에 CodeDeploy 배포 프로세스의 주요 구성 요소를 숙지해야 합니다.

주제

- 애플리케이션
- 컴퓨팅 플랫폼
- 배포 구성
- 배포 그룹
- 배포 유형
- IAM 인스턴스 프로파일
- 개정
- 서비스 역할
- 대상 수정 버전
- 기타 구성 요소

애플리케이션

애플리케이션은 배포할 애플리케이션을 식별할 수 있는 고유 이름입니다. CodeDeploy는 컨테이너 역할을 하는 이 이름을 사용하여 배포 중에 수정 버전, 배포 구성 및 배포 그룹의 올바른 조합을 참조하도록 합니다.

컴퓨팅 플랫폼

컴퓨팅 플랫폼은 CodeDeploy가 애플리케이션을 배포하는 플랫폼입니다. 다음과 같은 세 가지 컴퓨팅 플랫폼이 있습니다.

- EC2/온프레미스: 물리적 서버의 인스턴스를 설명합니다. Amazon EC2 클라우드 인스턴스나 온프레미스 서버 또는 둘 다일 수 있습니다. EC2/온프레미스 컴퓨팅 플랫폼을 사용하여 만든 애플리케이션은 실행 파일과 구성 파일, 이미지 및 기타 항목으로 구성될 수 있습니다.

  EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포에서는 인 플레이스 또는 블루/그린 배포 유형을 사용하여 인스턴스로 트래픽이 전송되는 방식을 관리합니다. 자세한 내용은 CodeDeploy 배포 유형 개요 단원을 참조하십시오.
- AWS Lambda: 업데이트된 버전의 Lambda 함수로 구성된 애플리케이션을 배포하는 데 사용됩니다. 는 고가용성 컴퓨팅 구조로 구성된 서버리스 컴퓨팅 환경에서 Lambda 함수를 AWS Lambda 관리합니다. 컴퓨팅 리소스의 모든 관리는에서 수행합니다 AWS Lambda. 자세한 정보는 서버리스 컴퓨팅 및 애플리케이션을 참조하세요. AWS Lambda 및 Lambda 함수에 대한 자세한 내용은 섹션을 참조하세요AWS Lambda.

  카나리(Canary), 선형(Linear) 또는 한 번에 모두(All-at-once) 구성을 선택하여 업데이트된 Lambda 함수 버전으로 트래픽을 전송하는 방식을 관리할 수 있습니다.
- Amazon ECS: Amazon ECS 컨테이너화된 애플리케이션을 작업 세트로 배포하는 데 사용됩니다. CodeDeploy는 애플리케이션의 업데이트 버전을 새로운 대체 작업 세트로 설치하여 블루/그린 배포를 수행합니다. CodeDeploy는 프로덕션 트래픽을 원래 애플리케이션 작업 세트에서 대체 작업 세트로 다시 라우팅합니다. 배포가 성공하면 기존 작업 세트는 종료됩니다. Amazon ECS에 대한 자세한 내용은 Amazon Elastic Container Service를 참조하세요.

  카나리(Canary), 선형(Linear) 또는 한번에 모두(All-at-once) 구성을 선택하여 배포 중 업데이트된 작업 세트로 트래픽을 전송하는 방식을 관리할 수 있습니다.

> **Note**
>
> Amazon ECS 블루/그린 배포는CodeDeploy 및 CloudFormation모두를 통해 지원됩니다. 이러한 배포에 대한 세부 정보는 다음 단원에서 설명합니다.

배포 구성

배포 구성이란 배포 중 CodeDeploy에서 사용하는 배포 규칙과 배포 성공 및 실패 조건 집합입니다. EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포의 경우, 해당 배포에 대해 정상 인스턴스의 최소 개수를 지정할 수 있습니다. 배포에서 AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼을 사용하는 경우 업데이트된 Lambda 함수 또는 ECS 작업 세트로 트래픽이 라우팅되는 방법을 지정할 수 있습니다.

EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포에서 최소 정상 호스트 개수를 지정하는 방법은 최소 정상 인스턴스 수 정보 단원을 참조하세요.

다음은 Lambda 또는 ECS 컴퓨팅 플랫폼을 사용하는 배포에서 트래픽을 라우팅하는 방식을 지정하는 배포 구성입니다.

- 카나리(Canary): 트래픽이 두 증분으로 나뉘어 이동합니다. 첫 번째 증분에서 업데이트된 Lambda 함수 또는 ECS 태스크로 이동할 트래픽의 백분율 및 두 번째 증분에서 나머지 트래픽의 이동을 시작하기 전까지 간격(분)을 지정하는 사전 정의된 카나리(Canary) 옵션 중에서 선택할 수 있습니다.
- 리니어(Linear): 트래픽이 동일한 증분 이동하며 각 증분 간에 시간 간격(분)이 동일합니다. 각 증분에서 이동할 트래픽 비율(%)과 각 증분 간의 시간 간격(분)을 지정하는 사전 정의된 선형 옵션에서 선택할 수 있습니다.
- 한 번에 모두(All-at-once): 모든 트래픽이 원본 Lambda 함수 또는 ECS 태스크 집합에서 업데이트된 함수 또는 태스크 집합으로 전부 한꺼번에 이동합니다.

배포 그룹

배포 그룹이란 개별 인스턴스 집합입니다. 배포 그룹에는 개별적으로 태그가 지정된 인스턴스, Amazon EC2 Auto Scaling 그룹의 Amazon EC2 인스턴스 또는 둘 다가 포함됩니다. Amazon EC2 인스턴스 태그에 대한 자세한 정보는 콘솔을 사용한 태그 작업을 참조하세요. 온프레미스 인스턴스에 대한 자세한 정보는 Working with On-Premises Instances 단원을 참조하세요. Amazon EC2 Auto Scaling에 대한 자세한 내용은 Amazon EC2 Auto Scaling과 CodeDeploy 통합 단원을 참조하세요.

배포 유형

배포 유형이란 배포 그룹의 인스턴스에서 최신 애플리케이션 수정 버전을 사용 가능하게 만드는 방법입니다. 배포 유형에는 두 가지가 있습니다.

- 현재 위치 배포: 배포 그룹의 각 인스턴스에 있는 애플리케이션이 중지되고 최신 애플리케이션 개정 버전이 설치되며 애플리케이션의 새 버전이 시작되고 유효성이 검사됩니다. 로드 밸런서를 사용하면 배포가 진행될 때 각 인스턴스를 등록 취소한 후 배포가 완료된 후 서비스로 복원할 수 있습니다.

EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포만 인 플레이스 배포를 사용할 수 있습니다. 현재 위치 배포에 대한 자세한 내용은 인 플레이스 배포 개요 단원을 참조하세요.

- Blue/Green 배포: 배포 동작은 사용하는 컴퓨팅 플랫폼에 따라 다릅니다.
  - EC2/온프레미스 컴퓨팅 플랫폼에서의 블루/그린 배포: 배포 그룹(원래 환경)의 인스턴스가 다음 단계를 거쳐 인스턴스의 다른 집합(대체 환경)으로 대체됩니다.
    - 인스턴스는 대체 환경을 위해 프로비저닝됩니다.
    - 최신 애플리케이션 수정은 대체 인스턴스에 설치됩니다.
    - 애플리케이션 테스트 및 시스템 검증과 같은 활동에 선택적 대기 시간이 발생합니다.
    - 대체 환경의 인스턴스가 하나 이상의 Elastic Load Balancing 로드 밸런서에 등록되고 트래픽이 이러한 인스턴스로 라우팅됩니다. 원래 환경의 인스턴스는 등록이 취소되고 종료되거나 다른 용도로 계속 실행될 수 있습니다.

    > **Note**
    >
    > EC2/온프레미스 컴퓨팅 플랫폼을 사용할 경우 블루/그린 배포는 Amazon EC2 인스턴스에서만 작동합니다.

  - AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼의 블루/그린: 트래픽은 canary, 선형 또는 all-at-once 배포 구성에 따라 증분으로 이동합니다.
  - 블루/그린 배포: CloudFormation CloudFormation 스택 업데이트의 일부로 트래픽이 현재 리소스에서 업데이트된 리소스로 이동합니다. 현재는 ECS 블루/그린 배포만 지원됩니다.

  블루/그린 배포에 대한 자세한 내용은 블루/그린 배포 개요 섹션을 참조하세요.

> **Note**
>
> Amazon ECS 블루/그린 배포는CodeDeploy 및 CloudFormation모두를 통해 지원됩니다. 이러한 배포에 대한 세부 정보는 다음 단원에서 설명합니다.

IAM 인스턴스 프로파일

IAM 인스턴스 프로파일이란 Amazon EC2 인스턴스에 연결하는 IAM 역할입니다. 이 프로파일에는 애플리케이션이 저장되는 Amazon S3 버킷 또는 GitHub 리포지토리에 액세스할 때 필요한 권한이 포함됩니다. 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 단원을 참조하십시오.

개정

수정 버전이란 애플리케이션의 버전입니다. AWS Lambda 배포 개정은 배포할 Lambda 함수에 대한 정보를 지정하는 YAML 또는 JSON 형식의 파일입니다. EC2/온프레미스 배포 개정은 소스 콘텐츠(소스 코드, 웹 페이지, 실행 파일 및 배포 스크립트)와 애플리케이션 사양 파일(AppSpec 파일)이 포함된 아카이브 파일입니다. AWS Lambda 개정은 Amazon S3 버킷에 저장할 수 있습니다. EC2/온프레미스 수정 버전은 Amazon S3 버킷 또는 GitHub 리포지토리에 저장됩니다. Amazon S3의 경우 수정 버전은 Amazon S3 객체 키 및 해당 ETag, 버전 또는 둘 다에 의해 고유하게 식별됩니다. GitHub의 경우 수정은 커밋 ID에 의해 고유하게 식별됩니다.

서비스 역할

서비스 역할은 AWS 리소스에 액세스할 수 있도록 AWS 서비스에 권한을 부여하는 IAM 역할입니다. 서비스 역할에 연결하는 정책에 따라 서비스가 액세스할 수 있는 AWS 리소스와 해당 리소스로 수행할 수 있는 작업이 결정됩니다. CodeDeploy의 경우 서비스 역할은 다음 작업에 사용됩니다.

- 인스턴스에 적용된 태그 또는 인스턴스와 연결된 Amazon EC2 Auto Scaling 그룹 이름을 읽습니다. 이를 통해 CodeDeploy는 애플리케이션을 배포할 수 있는 인스턴스를 식별할 수 있습니다.
- Amazon EC2 Auto Scaling 그룹 및 Elastic Load Balancing 로드 밸런서의 인스턴스에 대한 작업을 수행합니다.
- 지정된 배포 또는 인스턴스 이벤트가 발생할 때 알림을 전송할 수 있도록 Amazon SNS 주제에 정보를 게시합니다.
- CloudWatch 경보에 대한 정보를 검색하여 배포에 대한 경보 모니터링을 설정합니다.

자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.

대상 수정 버전

대상 수정 버전이란 리포지토리에 업로드했고 배포 그룹의 인스턴스에 배포하려는 가장 최신 버전의 애플리케이션 수정 버전입니다. 다시 말해서, 애플리케이션 수정은 현재 배포를 위해 대상 지정됩니다. 또한 이 수정은 자동 배포를 위해 폴링되는 수정입니다.

기타 구성 요소

CodeDeploy 워크플로의 다른 구성 요소에 대한 자세한 내용은 다음 주제를 참조하세요.

- CodeDeploy 리포지토리 유형 선택

- Deployments
- Application Specification Files
- Instance Health
- CodeDeploy 에이전트 작업
- Working with On-Premises Instances

CodeDeploy 배포

이 항목에서는 CodeDeploy의 배포 워크플로와 구성 요소에 대해 설명합니다. 배포 프로세스는 배포에 사용하는 컴퓨팅 플랫폼 또는 배포 방법(Lambda, Amazon ECS, EC2/온프레미스 또는 경유 AWS CloudFormation)에 따라 달라집니다.

주제

- AWS Lambda 컴퓨팅 플랫폼에 배포
- Amazon ECS 컴퓨팅 플랫폼에서의 배포
- EC2/온프레미스 컴퓨팅 플랫폼의 배포

AWS Lambda 컴퓨팅 플랫폼에 배포

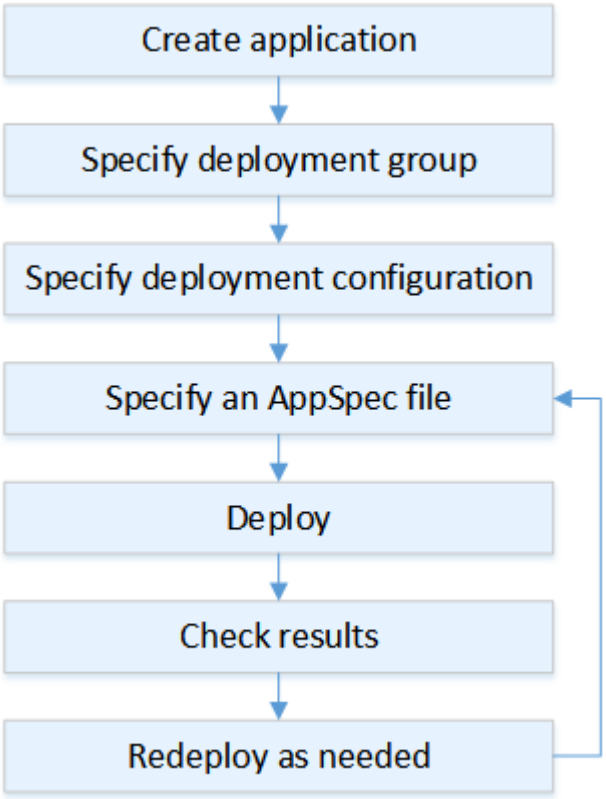
이 주제에서는 AWS Lambda 컴퓨팅 플랫폼을 사용하는 CodeDeploy 배포의 구성 요소 및 워크플로에 대한 정보를 제공합니다.

주제

- AWS Lambda 컴퓨팅 플랫폼의 배포 워크플로
- 애플리케이션 개정 업로드
- 애플리케이션 및 배포 그룹 만들기
- 애플리케이션 개정 배포
- 애플리케이션 업데이트
- 중지 및 실패한 배포
- 다시 배포 및 배포 롤백

AWS Lambda 컴퓨팅 플랫폼의 배포 워크플로

다음 다이어그램은 새로 업데이트된 AWS Lambda 함수 배포의 기본 단계를 보여줍니다.



이러한 단계는 다음과 같습니다.

1. 애플리케이션을 만든 후 배포할 애플리케이션 개정을 식별할 수 있는 고유 이름을 지정합니다. Lambda 함수를 배포하려면 애플리케이션을 만들 때 AWS Lambda 컴퓨팅 플랫폼을 선택합니다. CodeDeploy에서는 배포 중 이 이름을 사용하여 해당 배포가 올바른 배포 구성 요소(예: 배포 그룹, 배포 구성 및 애플리케이션 개정)를 참조하도록 합니다. 자세한 내용은 CodeDeploy를 사용하여 애플리케이션 생성 단원을 참조하십시오.
2. 배포 그룹 이름을 지정하여 배포 그룹을 설정합니다.
3. 배포 구성을 선택하여 트래픽이 원래 AWS Lambda 함수 버전에서 새 Lambda 함수 버전으로 이동하는 방법을 지정합니다. 자세한 내용은 View Deployment Configuration Details 단원을 참조하십시오.
4. 애플리케이션 사양 파일(AppSpec 파일)을 Amazon S3로 업로드 AppSpec 파일은 배포를 확인하는 데 사용되는 Lambda 함수 버전과 Lambda 함수를 지정합니다. AppSpec 파일을 만들지 않으려면 YAML 또는 JSON을 사용하여 콘솔에서 직접 Lambda 함수 버전과 Lambda 배포 확인 함수를 지정하면 됩니다. 자세한 내용은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하십시오.

5. 애플리케이션 개정을 배포 그룹에 배포합니다.는 지정한 Lambda 함수 개정을 AWS CodeDeploy 배포합니다. 애플리케이션을 만들 때 선택한 배포 AppSpec 파일을 사용하여 Lambda 함수 개정으로 트래픽이 이동합니다. 자세한 내용은 CodeDeploy에서 배포 만들기 단원을 참조하십시오.
6. 배포 결과를 확인합니다. 자세한 내용은 CodeDeploy에서 배포 모니터링 단원을 참조하십시오.

애플리케이션 개정 업로드

AppSpec 파일을 Amazon S3에 배치하거나, 콘솔 또는 AWS CLI에서 직접 입력합니다. 자세한 내용은 Application Specification Files 단원을 참조하십시오.

애플리케이션 및 배포 그룹 만들기

AWS Lambda 컴퓨팅 플랫폼의 CodeDeploy 배포 그룹은 하나 이상의 AppSpec 파일 모음을 식별합니다. 각 AppSpec 파일은 한 개의 Lambda 함수 버전을 배포할 수 있습니다. 배포 그룹은 또한 이후 배포를 위해 일련의 구성 옵션(경보 및 롤백 구성 등) 세트를 정의합니다.

애플리케이션 개정 배포

이제 AppSpec 파일에 지정된 함수 개정 버전을 배포 그룹에 배포할 수 있습니다. CodeDeploy 콘솔 또는 create-deployment 명령을 사용할 수 있습니다. 개정, 배포 그룹 및 배포 구성을 비롯하여 배포를 제어하기 위해 지정할 수 있는 파라미터가 있습니다.

애플리케이션 업데이트

애플리케이션을 업데이트한 다음 CodeDeploy 콘솔을 사용하거나 create-deployment 명령을 호출하여 개정 버전을 푸시할 수 있습니다.

중지 및 실패한 배포

CodeDeploy 콘솔 또는 stop-deployment 명령을 사용하여 배포를 중지할 수 있습니다. 배포를 중지하려고 하면 다음 3가지 동작 중 하나가 발생합니다.

- 배포가 중지되고 성공 상태가 반환됩니다. 이 경우 중지된 배포의 배포 그룹에서 더 이상 배포 수명 주기 이벤트가 실행되지 않습니다.
- 배포가 즉시 중지되지 않고, 대기 중 상태가 반환됩니다. 이 경우, 일부 배포 수명 주기 이벤트는 배포 그룹에서 계속 실행 중일 수 있습니다. 대기 중인 작업이 완료되면 배포 중지를 위한 후속 호출에서 성공 상태를 반환합니다.
- 배포를 중지할 수 없고 오류가 반환됩니다. 자세한 내용은 AWS CodeDeploy API 참조의 ErrorInformation 및 Common errors를 참조하세요.

중지된 배포와 마찬가지로, 실패한 배포도 일부 배포 수명 주기 이벤트가 이미 실행되었을 수 있습니다. 배포가 실패한 이유를 확인하려면 CodeDeploy 콘솔을 사용하거나 실패한 배포의 로그 파일 데이터를 분석합니다. 자세한 내용은 애플리케이션 수정 버전 및 로그 파일 정리 및 CodeDeploy EC2/온프레미스 배포에 대한 로그 데이터 보기 단원을 참조하세요.

다시 배포 및 배포 롤백

CodeDeploy에서는 이전에 배포된 개정 버전을 새 배포로 다시 배포하여 롤백을 구현합니다.

배포에 실패한 경우 또는 경보 모니터링 임계값에 도달한 경우 등 특정 조건이 충족되면 배포를 자동으로 롤백하도록 배포 그룹을 구성할 수 있습니다. 또한 개별 배포에서 배포 그룹에 대해 지정한 롤백 설정을 재정의할 수도 있습니다.

뿐만 아니라 이전에 배포한 개정을 수동으로 다시 배포하여 실패한 배포를 롤백하도록 선택할 수도 있습니다.

어느 경우에도 새 배포나 롤백 배포에 고유의 배포 ID가 할당됩니다. CodeDeploy 콘솔에서 볼 수 있는 배포 목록에는 자동 배포로 배포된 항목이 표시됩니다.

자세한 내용은 CodeDeploy를 사용하여 재배포 및 배포 롤백 단원을 참조하십시오.

Amazon ECS 컴퓨팅 플랫폼에서의 배포

이 항목에서는 Amazon ECS 컴퓨팅 플랫폼을 사용하는 CodeDeploy 배포의 워크플로 및 구성 요소에 대해 설명합니다.

주제

- Amazon ECS 배포를 시작하기 전
- Amazon ECS 컴퓨팅 플랫폼의 배포 워크플로(높은 수준)
- Amazon ECS 배포 중에 발생하는 일
- 애플리케이션 개정 업로드
- 애플리케이션 및 배포 그룹 만들기
- 애플리케이션 개정 배포
- 애플리케이션 업데이트
- 중지 및 실패한 배포
- 다시 배포 및 배포 롤백
- 를 통한 Amazon ECS 블루/그린 배포 AWS CloudFormation

Amazon ECS 배포를 시작하기 전

Amazon ECS 애플리케이션 배포를 시작하기 전에 다음을 준비해야 합니다. 일부 요구 사항은 배포 그룹을 만들 때 지정되고, 일부는 AppSpec 파일에 지정됩니다.

요구 사항	지정되는 위치
Amazon ECS 클러스터	배포 그룹
Amazon ECS 서비스	배포 그룹
Application Load Balancer 또는 Network Load Balancer	배포 그룹
프로덕션 리스너	배포 그룹
테스트 리스너(선택 사항)	배포 그룹
대상 그룹 두 개	배포 그룹
Amazon ECS 작업 정의	AppSpec 파일
컨테이너 이름	AppSpec 파일
컨테이너 포트	AppSpec 파일

Amazon ECS 클러스터

Amazon ECS 클러스터는 작업 또는 서비스의 논리적 그룹입니다. CodeDeploy 애플리케이션의 배포 그룹을 만들 때 Amazon ECS 서비스가 포함된 Amazon ECS 클러스터를 지정합니다. 자세한 내용은 Amazon Elastic Container Service 사용 설명서의 Amazon ECS 클러스터를 참조하세요.

Amazon ECS 서비스

Amazon ECS 서비스는 Amazon ECS 클러스터에서 지정된 작업 정의 인스턴스를 유지하고 실행합니다. Amazon ECS 서비스는 CodeDeploy에서 활성화되어 있어야 합니다. 기본적으로 Amazon ECS 서비스는 Amazon ECS 배포에 활성화되어 있어야 합니다. 배포 그룹을 만들 때 Amazon ECS 클러스터에 있는 Amazon ECS 서비스를 배포하도록 선택합니다. 자세한 내용은 Amazon Elastic Container Service 사용 설명서의 Amazon ECS 서비스를 참조하세요.

Application Load Balancer 또는 Network Load Balancer

Amazon ECS 배포로 업데이트하려는 Amazon ECS 서비스에서 Elastic Load Balancing을 사용해야 합니다. Application Load Balancer 또는 Network Load Balancer를 사용할 수 있습니다. 동적 포트 매핑 및 경로 기반 라우팅과 우선순위 규칙 등의 기능을 활용할 수 있도록 Application Load Balancer를 사용하는 것이 좋습니다. CodeDeploy 애플리케이션의 배포 그룹을 만들 때 로드 밸런서를 지정합니다. 자세한 내용은 CodeDeploy Amazon ECS 배포를 위한 로드 밸런서, 대상 그룹 및 리스너 설정 및 Amazon Elastic Container Service 사용 설명서의 로드 밸런서 생성을 참조하세요.

리스너 한 개 또는 두 개

리스너는 로드 밸런서가 대상 그룹으로 트래픽을 보내기 위해 사용합니다. 프로덕션 리스너 한 개는 필수입니다. 확인 테스트를 실행하는 동안 대체 작업 세트로 트래픽을 보내는 두 번째 테스트 리스너(선택 사항)를 지정할 수 있습니다. 배포 그룹을 만들 때 리스너를 한 개 또는 두 개 지정합니다. Amazon ECS 콘솔을 사용하여 Amazon ECS 서비스를 생성하는 경우, 리스너가 자동으로 생성됩니다. 자세한 내용은 Elastic Load Balancing 사용 설명서에서 애플리케이션 로드 밸런서의 리스너 및 Amazon Elastic Container Service 사용 설명서의 서비스 생성을 참조하세요.

2개의 Amazon ECS 대상 그룹

대상 그룹은 등록된 대상으로 트래픽을 라우팅하는 데 사용됩니다. Amazon ECS 배포에는 대상 그룹이 2개 필요합니다. 하나는 Amazon ECS 애플리케이션의 원래 작업 세트용이고 다른 하나는 대체 작업 세트용입니다. 배포 중에 CodeDeploy는 대체 작업 세트를 만들고 원래 작업 세트에서 새 작업 세트로 트래픽을 다시 라우팅합니다. CodeDeploy 애플리케이션의 배포 그룹을 만들 때 대상 그룹을 지정합니다.

배포 중에 CodeDeploy는 상태가 `PRIMARY`인 Amazon ECS 서비스의 작업 세트(원래 작업 세트)와 연결된 대상 그룹을 확인하고 대상 그룹 하나를 이 작업 세트와 연결한 후 다른 대상 그룹을 대체 작업 세트와 연결합니다. 다른 배포를 수행하는 경우, 현재 배포의 원래 작업 세트와 연결된 대상 그룹은 다음 배포의 대체 작업 세트와 연결됩니다. 자세한 내용은 Elastic Load Balancing 사용 설명서의 애플리케이션 로드 밸런서의 대상 그룹을 참조하세요.

Amazon ECS 작업 정의

작업 정의는 Amazon ECS 애플리케이션이 포함된 도커 컨테이너를 실행하는 데 필요합니다. CodeDeploy 애플리케이션의 AppSpec 파일에서 작업 정의의 ARN을 지정합니다. 자세한 내용은 Amazon Elastic Container Service 사용 설명서 및 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션의 Amazon ECS 작업 정의를 참조하세요.

Amazon ECS 애플리케이션의 컨테이너

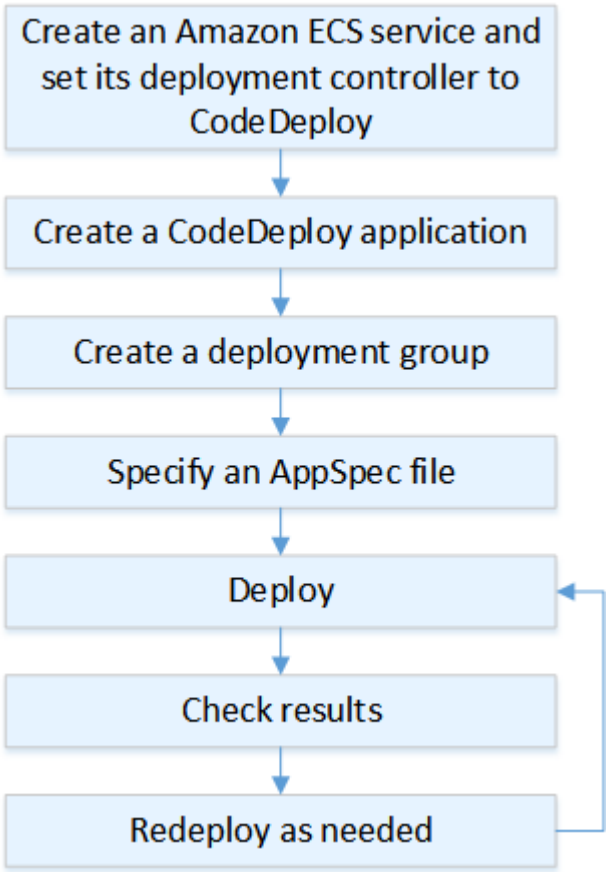
Docker 컨테이너란 애플리케이션이 실행될 수 있도록 코드와 해당 종속성을 패키징하는 소프트웨어 단위를 말합니다. 컨테이너는 애플리케이션을 격리하므로 애플리케이션이 다른 컴퓨팅 환경에서도 실행됩니다. 로드 밸런서는 Amazon ECS 애플리케이션 작업 세트의 컨테이너로 트래픽을 보냅니다. CodeDeploy 애플리케이션의 AppSpec 파일에서 컨테이너의 이름을 지정합니다. AppSpec 파일에 지정된 컨테이너는 Amazon ECS 작업 정의에 지정된 컨테이너 중 하나여야 합니다. 자세한 내용은 Amazon Elastic Container Service 사용 설명서의 Amazon Elastic Container Service란 무엇입니까?와 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션를 참조하세요.

대체 작업 세트의 포트

Amazon ECS 배포 중에 로드 밸런서가 CodeDeploy 애플리케이션의 AppSpec 파일에 지정된 이 포트로 트래픽을 보냅니다. CodeDeploy 애플리케이션의 AppSpec 파일에서 포트의 이름을 지정합니다. 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션 단원을 참조하십시오.

Amazon ECS 컴퓨팅 플랫폼의 배포 워크플로(높은 수준)

다음 다이어그램은 업데이트된 Amazon ECS 서비스 배포의 기본 단계를 보여줍니다.



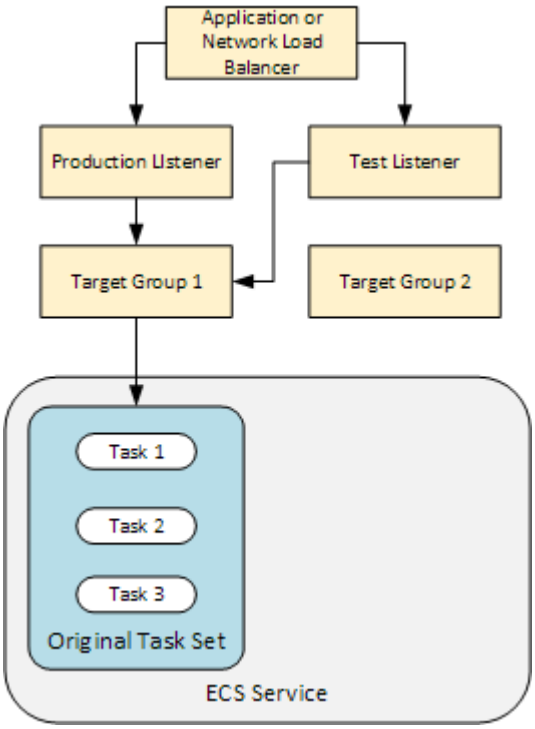
이러한 단계는 다음과 같습니다.

1. 배포하려는 항목을 고유하게 나타내는 이름을 지정하여 AWS CodeDeploy 애플리케이션을 생성합니다. Amazon ECS 애플리케이션을 배포하려면 AWS CodeDeploy 애플리케이션에서 Amazon ECS 컴퓨팅 플랫폼을 선택합니다. CodeDeploy는 배포 중에 애플리케이션을 사용하여 배포 그룹, 대상 그룹, 리스너, 트래픽 다시 라우팅 동작, 애플리케이션 개정 등의 올바른 배포 구성 요소를 참조합니다. 자세한 내용은 CodeDeploy를 사용하여 애플리케이션 생성 단원을 참조하십시오.
2. 다음을 지정하여 배포 그룹을 설정합니다.
   - 배포 그룹 이름.
   - Amazon ECS 클러스터와 서비스 이름. Amazon ECS 서비스의 배포 컨트롤러는 CodeDeploy로 설정되어야 합니다.
   - 배포 중에 사용되는 프로덕션 리스너, 테스트 리스너(선택 사항), 대상 그룹
   - 배포 설정(예: Amazon ECS 서비스의 대체 Amazon ECS 작업 세트로 프로덕션 트래픽을 다시 라우팅할 시기 및 Amazon ECS 서비스의 원래 Amazon ECS 작업 세트를 종료할 시기)
   - 설정(선택 사항) (예: 트리거, 경보, 롤백 동작)
3. 애플리케이션 사양 파일(AppSpec 파일)을 지정합니다. Amazon S3에 업로드하거나, 콘솔에 YAML 또는 JSON 형식으로 입력하거나, AWS CLI 또는 SDK를 사용하여 지정할 수 있습니다. AppSpec 파일은 배포에 사용되는 Amazon ECS 작업 정의, 트래픽을 라우팅하는 데 사용되는 포트 매핑과 컨테이너 이름, 배포 수명 주기 후크 후 실행되는 Lambda 함수를 지정합니다. 컨테이너 이름은 Amazon ECS 작업 정의의 컨테이너여야 합니다. 자세한 내용은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하십시오.
4. 애플리케이션 개정을 배포합니다.는 Amazon ECS 서비스에 있는 작업 세트의 원래 버전에서 새로운 대체 작업 세트로 트래픽을 AWS CodeDeploy 다시 라우팅합니다. 배포 그룹에 지정된 대상 그룹은 원래 및 대체 작업 세트에 트래픽을 제공하는 데 사용됩니다. 배포가 완료되면 원래 작업 세트가 종료됩니다. 트래픽을 다시 라우팅하기 전에 테스트 리스너(선택 사항)를 지정하여 대체 버전에 테스트 트래픽을 제공할 수 있습니다. 자세한 내용은 CodeDeploy에서 배포 만들기 단원을 참조하십시오.
5. 배포 결과를 확인합니다. 자세한 내용은 CodeDeploy에서 배포 모니터링 단원을 참조하십시오.

Amazon ECS 배포 중에 발생하는 일

테스트 리스너를 사용해 Amazon ECS 배포를 시작하려면 먼저 구성 요소를 구성해야 합니다. 자세한 내용은 Amazon ECS 배포를 시작하기 전 단원을 참조하십시오.

다음 다이어그램은 Amazon ECS 배포를 시작할 준비를 마쳤을 때 구성 요소의 관계를 나타낸 것입니다.



배포가 시작되면 배포 수명 주기 이벤트가 한 번에 하나씩 실행되기 시작합니다. 일부 수명 주기 이벤트는 AppSpec 파일에서 지정된 Lambda 함수만 실행하는 후크입니다. 다음 표의 배포 수명 주기 이벤트는 실행 순서대로 나열되어 있습니다. 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션 단원을 참조하십시오.

수명 주기 이벤트	수명 주기 이벤트 작업
`BeforeInstall` (Lambda 함수 후크)	Lambda 함수 실행
설치	대체 작업 세트를 설정합니다.
`AfterInstall` (Lambda 함수 후크)	Lambda 함수 실행
AllowTestTraffic	트래픽을 테스트 리스너에서 대상 그룹 2로 라우팅합니다.
`AfterAllowTestTraffic` (Lambda 함수 후크)	Lambda 함수 실행
`BeforeAllowTraffic` (Lambda 함수 후크)	Lambda 함수 실행

수명 주기 이벤트	수명 주기 이벤트 작업
AllowTraffic	트래픽을 프로덕션 리스너에서 대상 그룹 2로 라우팅합니다.
`AfterAllowTraffic`	Lambda 함수 실행

> **Note**
>
> 후크의 Lambda 함수는 선택 사항입니다.

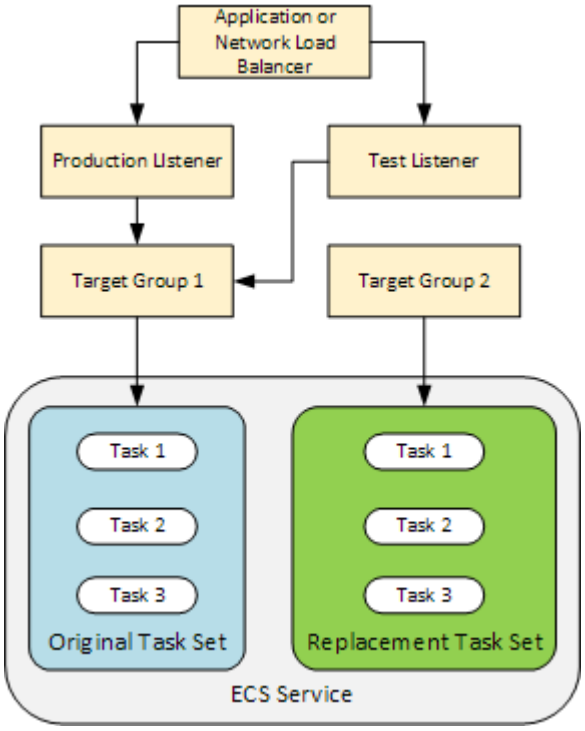
1. AppSpec 파일의 `BeforeInstall` 후크에서 지정한 Lambda 함수를 실행합니다.
2. `Install` 수명 주기 이벤트 도중:

   a. 대체 작업 세트가 Amazon ECS 서비스에 생성됩니다.

   b. 업데이트된 컨테이너화 애플리케이션이 대체 작업 세트에 설치됩니다.

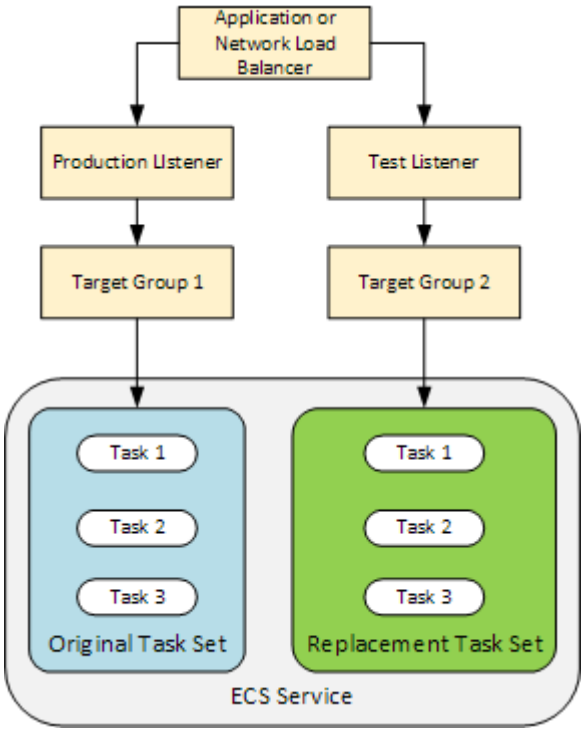
   c. 두 번째 대상 그룹이 대체 작업 세트와 연결됩니다.

   아래 다이어그램은 새로운 대체 작업 세트와 함께 배포 구성 요소를 나타낸 것입니다. 컨테이너화 애플리케이션은 이 작업 세트에 설치되어 있습니다. 작업 세트는 세 가지 작업으로 구성됩니다. 애플리케이션의 작업 수는 무제한입니다. 이제 두 번째 대상 그룹이 대체 작업 세트와 연결됩니다.



3. AppSpec 파일의 `AfterInstall` 후크에서 지정한 Lambda 함수를 실행합니다.

4. `AllowTestTraffic` 이벤트가 호출됩니다. 수명 주기 이벤트 과정에서 테스트 리스너가 트래픽을 업데이트된 컨테이너화 애플리케이션으로 라우팅합니다.



5. AppSpec 파일의 `AfterAllowTestTraffic` 후크에서 지정한 Lambda 함수를 실행합니다. Lambda 함수는 테스트 트래픽을 사용해 배포를 검증할 수 있습니다. 예를 들어 Lambda 함수는 트래픽을 테스트 리스너까지 전송한 후 대체 작업 세트에서 지표를 추적할 수 있습니다. 롤백이 구성된 경우에는 Lambda 함수에서 검증 테스트에 실패할 경우 롤백을 트리거하는 CloudWatch 경보를 구성할 수 있습니다.

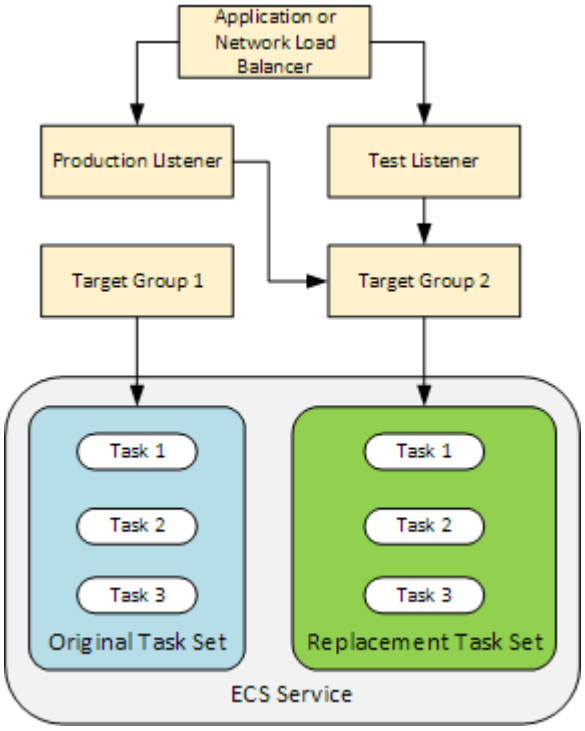
   검증 테스트가 완료되면 다음 중 한 가지가 발생합니다.

   - 검증에 실패하고 롤백이 구성된 경우 배포 상태가 `Failed`로 표시되고 구성 요소는 배포가 시작되었던 상태로 돌아갑니다.
   - 검증에 실패하였지만 롤백이 구성되어 있지 않다면 배포 상태가 `Failed`로 표시되고 구성 요소는 현재 상태를 유지합니다.
   - 검증에 성공한 경우에는 배포가 `BeforeAllowTraffic` 후크까지 이어집니다.

   자세한 내용은 CodeDeploy에서 CloudWatch 경보를 사용하여 배포 모니터링, 자동 롤백, 배포 그룹에 대한 고급 옵션 구성 단원을 참조하세요.

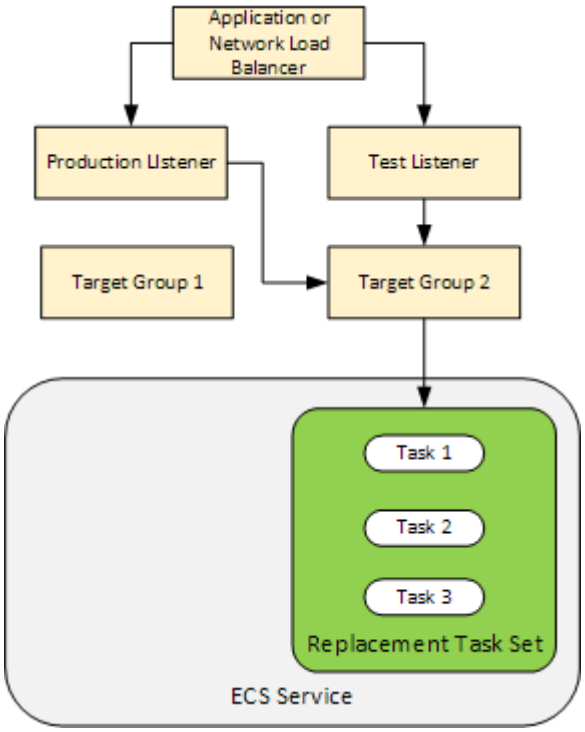
6. AppSpec 파일의 `BeforeAllowTraffic` 후크에서 지정한 Lambda 함수를 실행합니다.

7. `AllowTraffic` 이벤트가 호출됩니다. 프로덕션 트래픽이 원래 작업 세트에서 대체 작업 세트로 다시 라우팅됩니다. 다음 다이어그램은 프로덕션 트래픽을 수신하는 대체 작업 세트를 나타낸 것입니다.



8. AppSpec 파일의 `AfterAllowTraffic` 후크에서 지정한 Lambda 함수를 실행합니다.

9. 모든 이벤트에 성공하면 배포 상태가 `Succeeded`로 설정되고 원래 작업 세트는 제거됩니다.



애플리케이션 개정 업로드

AppSpec 파일을 Amazon S3에 배치하거나, 콘솔 또는 AWS CLI에서 직접 입력합니다. 자세한 내용은 Application Specification Files 단원을 참조하십시오.

애플리케이션 및 배포 그룹 만들기

Amazon ECS 컴퓨팅 플랫폼의 CodeDeploy 배포 그룹은 배포 중에 사용되는 두 대상 그룹과 업데이트된 Amazon ECS 애플리케이션에 트래픽을 제공하는 리스너를 식별합니다. 배포 그룹은 또한 일련의 구성 옵션(경보 및 롤백 구성 등) 세트를 정의합니다.

애플리케이션 개정 배포

이제 배포 그룹에 지정된 업데이트된 Amazon ECS 서비스를 배포할 준비가 되었습니다. CodeDeploy 콘솔 또는 create-deployment 명령을 사용할 수 있습니다. 개정 및 배포 그룹을 비롯하여 배포를 제어하기 위해 지정할 수 있는 파라미터가 있습니다.

애플리케이션 업데이트

애플리케이션을 업데이트한 다음 CodeDeploy 콘솔을 사용하거나 create-deployment 명령을 호출하여 개정 버전을 푸시할 수 있습니다.

중지 및 실패한 배포

CodeDeploy 콘솔 또는 stop-deployment 명령을 사용하여 배포를 중지할 수 있습니다. 배포를 중지하려고 하면 다음 3가지 동작 중 하나가 발생합니다.

- 배포가 중지되고 성공 상태가 반환됩니다. 이 경우 중지된 배포의 배포 그룹에서 더 이상 배포 수명 주기 이벤트가 실행되지 않습니다.
- 배포가 즉시 중지되지 않고, 대기 중 상태가 반환됩니다. 이 경우, 일부 배포 수명 주기 이벤트는 배포 그룹에서 계속 실행 중일 수 있습니다. 대기 중인 작업이 완료되면 배포 중지를 위한 후속 호출에서 성공 상태를 반환합니다.
- 배포를 중지할 수 없고 오류가 반환됩니다. 자세한 내용은 AWS CodeDeploy API 참조의 오류 정보 및 일반 오류를 참조하세요.

다시 배포 및 배포 롤백

CodeDeploy는 대체 작업 세트에서 원래 작업 세트로 트래픽을 다시 라우팅하여 롤백을 구현합니다.

배포에 실패한 경우 또는 경보 모니터링 임계값에 도달한 경우 등 특정 조건이 충족되면 배포를 자동으로 롤백하도록 배포 그룹을 구성할 수 있습니다. 또한 개별 배포에서 배포 그룹에 대해 지정한 롤백 설정을 재정의할 수도 있습니다.

뿐만 아니라 이전에 배포한 개정을 수동으로 다시 배포하여 실패한 배포를 롤백하도록 선택할 수도 있습니다.

어느 경우에도 새 배포나 롤백 배포에 고유의 배포 ID가 할당됩니다. CodeDeploy 콘솔에 자동 배포의 결과인 배포 목록이 표시됩니다.

다시 배포하는 경우 현재 배포의 원래 작업 세트와 연결된 대상 그룹은 재배포의 대체 작업 세트와 연결됩니다.

자세한 내용은 CodeDeploy를 사용하여 재배포 및 배포 롤백 단원을 참조하십시오.

를 통한 Amazon ECS 블루/그린 배포 AWS CloudFormation

AWS CloudFormation 를 사용하여 CodeDeploy를 통해 Amazon ECS 블루/그린 배포를 관리할 수 있습니다. 자세한 내용은 를 통해 Amazon ECS 블루/그린 배포 생성 CloudFormation 단원을 참조하십시오.

> **Note**
>
> 아시아 태평양(오사카) 리전에서는를 사용하여 Amazon ECS 블루/그린 배포를 관리할 수 CloudFormation 없습니다.

EC2/온프레미스 컴퓨팅 플랫폼의 배포

이 항목에서는 EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 CodeDeploy 배포의 워크플로 및 구성 요소에 대해 설명합니다. 블루/그린 배포에 대한 자세한 내용은 블루/그린 배포 개요 단원을 참조하세요.

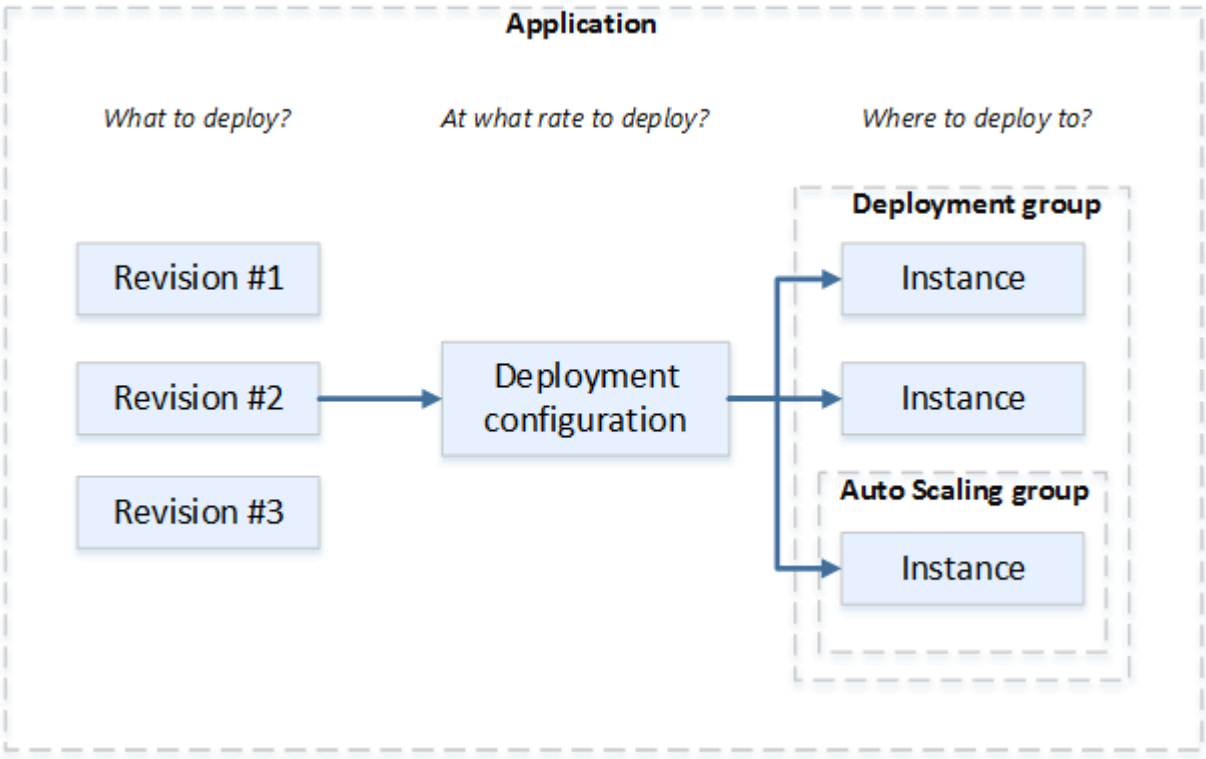
주제

- EC2/온프레미스 컴퓨팅 플랫폼의 배포 구성 요소
- EC2/온프레미스 컴퓨팅 플랫폼의 배포 워크플로
- 인스턴스 설정
- 애플리케이션 개정 업로드

- 애플리케이션 및 배포 그룹 만들기
- 애플리케이션 개정 배포
- 애플리케이션 업데이트
- 중지 및 실패한 배포
- 다시 배포 및 배포 롤백

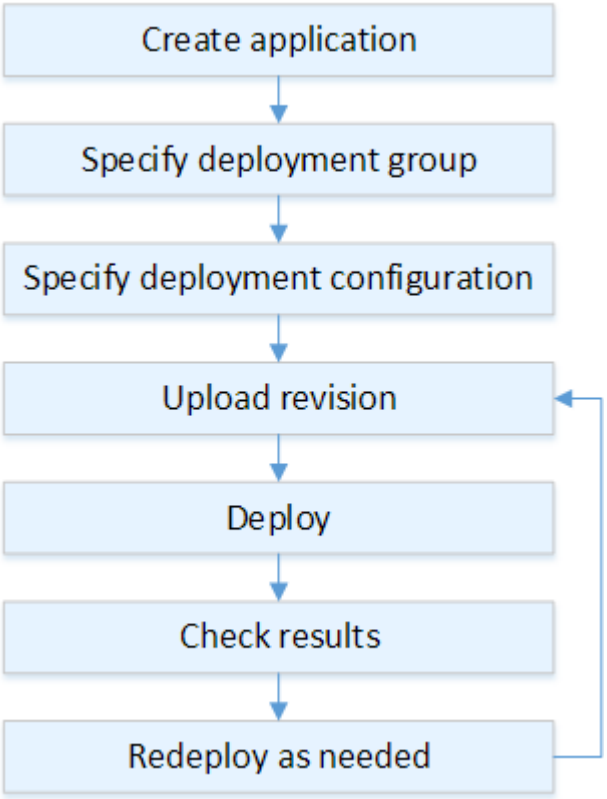
EC2/온프레미스 컴퓨팅 플랫폼의 배포 구성 요소

다음 다이어그램은 EC2/온프레미스 컴퓨팅 플랫폼에서의 CodeDeploy 배포 구성 요소를 보여줍니다.



EC2/온프레미스 컴퓨팅 플랫폼의 배포 워크플로

다음 다이어그램은 애플리케이션 개정 배포의 주요 단계를 보여줍니다.



이러한 단계는 다음과 같습니다.

1. 애플리케이션을 생성하고 배포하려는 애플리케이션 개정 버전과 애플리케이션의 컴퓨팅 플랫폼을 고유하게 식별하는 이름을 지정합니다. CodeDeploy에서는 배포 중 이 이름을 사용하여 해당 배포가 올바른 배포 구성 요소(예: 배포 그룹, 배포 구성 및 애플리케이션 개정)를 참조하도록 합니다. 자세한 내용은 CodeDeploy를 사용하여 애플리케이션 생성 단원을 참조하십시오.
2. 애플리케이션 개정을 배포하려는 인스턴스 및 배포 유형을 지정하여 배포 그룹을 설정합니다. 인 플레이스(in-place) 배포는 최신 애플리케이션 개정으로 인스턴스를 업데이트합니다. 블루/그린 배포는 로드 밸런서에 배포 그룹의 대체 인스턴스 세트를 등록하고 원본 인스턴스의 등록을 취소합니다.

   인스턴스, Amazon EC2 Auto Scaling 그룹 이름 또는 둘 다에 적용되는 태그를 지정할 수 있습니다.

   배포 그룹에 하나의 태그 그룹을 지정한 경우, CodeDeploy에서는 최소 한 개의 지정된 태그가 적용된 인스턴스를 배포합니다. 2개 이상의 태그 그룹을 지정한 경우 CodeDeploy는 각 태그 그룹의 기준에 맞는 인스턴스만 배포합니다. 자세한 내용은 Tagging Instances for Deployments 단원을 참조하십시오.

   어떤 경우든 인스턴스는 배포에 사용하도록 구성되어야 하며(즉, 태그가 지정되거나 Amazon EC2 Auto Scaling 그룹에 속해야 함), CodeDeploy 에이전트를 설치하여 실행 중이어야 합니다.

CloudFormation 템플릿을 사용하여 Amazon Linux 또는 Windows Server를 기반으로 Amazon EC2 인스턴스를 빠르게 설정할 수 있습니다. 또한 Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux(RHEL) 또는 Windows Server 인스턴스에 설치할 수 있는 독립 실행형 CodeDeploy 에이전트도 제공합니다. 자세한 내용은 CodeDeploy에서 배포 그룹 만들기 단원을 참조하십시오.

다음 옵션도 지정할 수 있습니다.

- Amazon SNS 알림. 배포 및 인스턴스에서 지정된 이벤트(예: 성공 또는 실패 이벤트)가 발생하면 Amazon SNS 주제 구독자에게 알림을 보내는 트리거를 만듭니다. 자세한 내용은 Monitoring Deployments with Amazon SNS Event Notifications 단원을 참조하십시오.
- 경보 기반 배포 및 관리. 지표가 CloudWatch에 설정된 임계값을 초과하거나 임계값 미만인 경우 배포를 중지하는 Amazon CloudWatch 경보 모니터링을 구현합니다.
- 자동 배포 롤백. 배포에 실패하거나 경보 임계값에 도달한 경우 이전에 알려진 양호한 상태의 개정으로 자동 롤백되도록 배포를 구성합니다.

3. 애플리케이션 개정을 동시에 배포해야 할 인스턴스 수와 배포 성공 및 실패 조건을 설정하는 배포 구성을 지정합니다. 자세한 내용은 View Deployment Configuration Details 단원을 참조하십시오.
4. 애플리케이션 개정을 Amazon S3 또는 GitHub로 업로드합니다. 배포하려는 파일 및 배포 중 실행하려는 모든 스크립트 이외에 애플리케이션 사양 파일(AppSpec file)을 포함해야 합니다. 이 파일에는 각 인스턴스에 파일을 복사할 위치 및 배포 스크립트를 실행할 시점 등과 같은 배포 명세가 포함되어 있습니다. 자세한 내용은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하십시오.
5. 애플리케이션 개정을 배포 그룹에 배포합니다. 배포 그룹의 각 인스턴스에 설치된 CodeDeploy 에이전트는 애플리케이션 개정을 Amazon S3 또는 GitHub에서 인스턴스로 복사합니다. 그런 다음 CodeDeploy 에이전트는 개정의 번들을 해제하고, AppSpec 파일을 사용하여 지정한 위치로 파일을 복사하고, 배포 스크립트를 실행합니다. 자세한 내용은 CodeDeploy에서 배포 만들기 단원을 참조하십시오.
6. 배포 결과를 확인합니다. 자세한 내용은 CodeDeploy에서 배포 모니터링 단원을 참조하십시오.
7. 개정을 다시 배포합니다. 소스 콘텐츠에서 버그를 수정하거나, 배포 스크립트를 다른 순서로 실행하거나, 실패한 배포를 해결해야 하는 경우 다시 배포하려고 할 수 있습니다. 다시 배포하려면 개정된 소스 콘텐츠, 배포 스크립트, AppSpec 파일을 새 개정으로 다시 번들링한 다음 해당 개정을 Amazon S3 버킷 또는 GitHub 리포지토리로 업로드합니다. 그런 다음 새 개정을 사용하여 동일한 배포 그룹으로 새로 배포를 실행합니다. 자세한 내용은 CodeDeploy에서 배포 만들기 단원을 참조하십시오.

인스턴스 설정

처음으로 애플리케이션 개정을 배포하려면 인스턴스를 설정해야 합니다. 애플리케이션 개정에 프로덕션 서버 3개와 백업 서버 2개가 필요한 경우 인스턴스 5개를 시작 또는 사용합니다.

인스턴스를 수동으로 프로비저닝하려면:

1. 인스턴스에 CodeDeploy 에이전트 설치합니다. Amazon Linux, Ubuntu Server, RHEL, Windows Server 인스턴스에 CodeDeploy 에이전트를 설치할 수 있습니다.
2. 배포 그룹에서 인스턴스를 식별하는 데 태그를 사용하는 경우 태깅을 활성화합니다. CodeDeploy에서는 태그를 사용하여 인스턴스를 식별하고 CodeDeploy 배포 그룹으로 그룹화합니다. 시작하기 자습서를 둘 다 진행했지만 키 또는 값을 사용하여 배포 그룹의 태그를 정의할 수 있습니다.
3. IAM 인스턴스 프로파일로 Amazon EC2 인스턴스를 시작합니다. IAM 인스턴스 프로파일은 CodeDeploy 에이전트가 인스턴스의 자격 증명을 확인하기 위해 시작되므로 Amazon EC2 인스턴스에 연결되어 있어야 합니다.
4. 서비스 역할을 만듭니다. CodeDeploy가 AWS 계정의 태그를 확장할 수 있도록 서비스 액세스 권한을 제공합니다.

초기 배포의 경우 CloudFormation 템플릿이이 모든 작업을 자동으로 수행합니다. 이 템플릿은 이미 설치된 CodeDeploy 에이전트를 사용하여 Amazon Linux 또는 Windows Server에 기반한 단일 Amazon EC2 인스턴스를 새로 만들고 구성합니다. 자세한 내용은 CodeDeploy용 인스턴스 작업 단원을 참조하십시오.

> **Note**
>
> 블루/그린 배포의 경우 대체 환경을 위해 이미 보유하고 있는 인스턴스를 사용하거나, CodeDeploy에서 배포 프로세스의 일부로 새 인스턴스를 프로비저닝하게 하는 두 가지 옵션 중에서 선택할 수 있습니다.

애플리케이션 개정 업로드

AppSpec 파일을 애플리케이션의 소스 콘텐츠 폴더 구조의 루트 폴더에 저장합니다. 자세한 내용은 Application Specification Files 단원을 참조하십시오.

애플리케이션의 소스 콘텐츠 폴더 구조를 아카이브 파일 형식(예: zip, tar 또는 압축 tar)으로 번들링합니다. 아카이브 파일(개정 버전)을 Amazon S3 버킷 또는 GitHub 리포지토리에 업로드합니다.

> **Note**
>
> tar 및 압축된 tar 아카이브 파일 형식(.tar 및 .tar.gz)은 Windows Server 인스턴스에서 지원되지 않습니다.

애플리케이션 및 배포 그룹 만들기

CodeDeploy 배포 그룹은 태그, Amazon EC2 Auto Scaling 그룹 이름 또는 둘 모두를 기반으로 인스턴스 모음을 식별합니다. 애플리케이션 개정 여러 개를 동일한 인스턴스에 배포하거나, 애플리케이션 개정 하나를 여러 인스턴스에 배포할 수 있습니다.

예를 들어, 프로덕션 서버 3개에 태그 "Prod"를 추가하고 백업 서버 2개에 태그 "Backup"을 추가할 수 있습니다. 이러한 두 태그를 사용하여 CodeDeploy 애플리케이션에서 두 가지 배포 그룹을 만들 수 있으며, 배포에 포함시킬 서버 집합(또는 둘 다)을 선택할 수 있습니다.

배포 그룹에 여러 태그 그룹을 사용하여 적은 수의 인스턴스로 배포를 제한할 수 있습니다. 자세한 내용은 Tagging Instances for Deployments 단원을 참조하세요.

애플리케이션 개정 배포

이제 Amazon S3 또는 GitHub에서 배포 그룹으로 애플리케이션 개정을 배포할 준비가 되었습니다. CodeDeploy 콘솔 또는 create-deployment 명령을 사용할 수 있습니다. 개정, 배포 그룹 및 배포 구성을 비롯하여 배포를 제어하기 위해 지정할 수 있는 파라미터가 있습니다.

애플리케이션 업데이트

애플리케이션을 업데이트한 다음 CodeDeploy 콘솔을 사용하거나 create-deployment 명령을 호출하여 개정 버전을 푸시할 수 있습니다.

중지 및 실패한 배포

CodeDeploy 콘솔 또는 stop-deployment 명령을 사용하여 배포를 중지할 수 있습니다. 배포를 중지하려고 하면 다음 3가지 동작 중 하나가 발생합니다.

- 배포가 중지되고 성공 상태가 반환됩니다. 이 경우 중지된 배포의 배포 그룹에서 더 이상 배포 수명 주기 이벤트가 실행되지 않습니다. 배포 그룹의 인스턴스 중 하나 이상에 일부 파일이 이미 복사되었거나 이러한 인스턴스에서 일부 스크립트가 이미 실행되었을 수 있습니다.
- 배포가 즉시 중지되지 않고, 대기 중 상태가 반환됩니다. 이 경우, 일부 배포 수명 주기 이벤트는 배포 그룹에서 계속 실행 중일 수 있습니다. 배포 그룹의 인스턴스 중 하나 이상에 일부 파일이 이미 복사

되었거나 이러한 인스턴스에서 일부 스크립트가 이미 실행되었을 수 있습니다. 대기 중인 작업이 완료되면 배포 중지를 위한 후속 호출에서 성공 상태를 반환합니다.

- 배포를 중지할 수 없고 오류가 반환됩니다. 자세한 내용은 AWS CodeDeploy API 참조의 ErrorInformation 및 Common errors를 참조하세요.

중지된 배포와 마찬가지로 배포에 실패하면 배포 그룹에 있는 하나 이상의 인스턴스에서 일부 배포 수명 주기 이벤트가 이미 실행되었을 수 있습니다. 배포에 실패한 이유를 확인하려면 CodeDeploy 콘솔을 사용하거나, get-deployment-instance 명령을 호출하거나, 실패한 배포에서 로그 파일 데이터를 분석할 수 있습니다. 자세한 내용은 애플리케이션 수정 버전 및 로그 파일 정리 및 CodeDeploy EC2/온프레미스 배포에 대한 로그 데이터 보기 단원을 참조하세요.

다시 배포 및 배포 롤백

CodeDeploy에서는 이전에 배포된 개정 버전을 새 배포로 다시 배포하여 롤백을 구현합니다.

배포에 실패한 경우 또는 경보 모니터링 임계값에 도달한 경우 등 특정 조건이 충족되면 배포를 자동으로 롤백하도록 배포 그룹을 구성할 수 있습니다. 또한 개별 배포에서 배포 그룹에 대해 지정한 롤백 설정을 재정의할 수도 있습니다.

뿐만 아니라 이전에 배포한 개정을 수동으로 다시 배포하여 실패한 배포를 롤백하도록 선택할 수도 있습니다.

어느 경우에도 새 배포나 롤백 배포에 고유의 배포 ID가 할당됩니다. CodeDeploy 콘솔에서 볼 수 있는 배포 목록에는 자동 배포로 배포된 항목이 표시됩니다.

자세한 내용은 CodeDeploy를 사용하여 재배포 및 배포 롤백 단원을 참조하십시오.

CodeDeploy 애플리케이션 사양(AppSpec) 파일

CodeDeploy에 고유한 애플리케이션 사양 파일(AppSpec 파일)은 YAML 형식 또는 JSON 형식 파일입니다. AppSpec 파일은 파일에 정의된 일련의 수명 주기 이벤트 후크로 각 배포를 관리하는 데 사용됩니다.

올바른 형식의 AppSpec 파일을 만드는 방법에 대한 자세한 내용은 CodeDeploy AppSpec 파일 참조 단원을 참조하세요.

주제

- Amazon ECS 컴퓨팅 플랫폼에 대한 AppSpec 파일

- AWS Lambda 컴퓨팅 플랫폼의 AppSpec 파일
- EC2/온프레미스 컴퓨팅 플랫폼에 대한 AppSpec 파일
- CodeDeploy 에이전트에서 AppSpec 파일을 사용하는 방법

Amazon ECS 컴퓨팅 플랫폼에 대한 AppSpec 파일

애플리케이션에서 Amazon ECS 컴퓨팅 플랫폼을 사용하는 경우 AppSpec 파일은 YAML 또는 JSON 형식일 수도 있습니다. 콘솔의 편집기에 직접 입력할 수도 있습니다. AppSpec 파일은 다음을 지정하는 데 사용됩니다.

- Amazon ECS 서비스의 이름과 새 작업 세트로 트래픽을 보내는 데 사용되는 컨테이너 이름 및 포트.
- 검증 테스트로 사용할 함수

배포 수명 주기 이벤트 후에 유효성 검사 Lambda 함수를 실행할 수 있습니다. 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션, Amazon ECS 배포의 AppSpec 파일 구조, Amazon ECS 배포용 AppSpec 파일 예제 단원을 참조하세요.

AWS Lambda 컴퓨팅 플랫폼의 AppSpec 파일

애플리케이션이 AWS Lambda 컴퓨팅 플랫폼을 사용하는 경우 AppSpec 파일의 형식을 YAML 또는 JSON으로 지정할 수 있습니다. 콘솔의 편집기에 직접 입력할 수도 있습니다. AppSpec 파일은 다음을 지정하는 데 사용됩니다.

- 배포할 AWS Lambda 함수 버전입니다.
- 검증 테스트로 사용할 함수

배포 수명 주기 이벤트 후에 유효성 검사 Lambda 함수를 실행할 수 있습니다. 자세한 내용은 AWS Lambda 배포를 위한 AppSpec 'hooks' 섹션 단원을 참조하십시오.

EC2/온프레미스 컴퓨팅 플랫폼에 대한 AppSpec 파일

애플리케이션에서 EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 경우 AppSpec 파일은 항상 YAML 형식입니다. AppSpec 파일은 다음 작업을 수행하는 데 사용됩니다.

- 애플리케이션 개정의 소스 파일을 인스턴스의 대상으로 매핑합니다.
- 배포된 파일에 대한 사용자 지정 권한을 지정합니다.

- 배포 프로세스의 다양한 단계에서 각 인스턴스에 실행할 스크립트를 지정합니다.

여러 개별 배포 수명 주기 이벤트 후에 인스턴스에서 스크립트를 실행할 수 있습니다. CodeDeploy는 파일에 지정된 스크립트만 실행하지만 이러한 스크립트는 인스턴스에서 다른 스크립트를 호출할 수 있습니다. 인스턴스에서 실행 중인 운영 체제에서 지원하는 경우 모든 유형의 스크립트를 실행할 수 있습니다. 자세한 내용은 EC2/온프레미스 배포를 위한 AppSpec 'hooks' 섹션 단원을 참조하십시오.

CodeDeploy 에이전트에서 AppSpec 파일을 사용하는 방법

배포하는 동안 CodeDeploy 에이전트는 AppSpec 파일의 hooks 섹션에서 현재 이벤트의 이름을 조회합니다. 이벤트가 발견되지 않으면 CodeDeploy 에이전트가 다음 단계로 이동합니다. 이벤트가 발견되면 CodeDeploy 에이전트가 실행할 스크립트 목록을 검색합니다. 스크립트는 파일에 나타나는 순서대로 순차적으로 실행됩니다. 각 스크립트의 상태는 인스턴스의 CodeDeploy 에이전트 로그 파일에 기록됩니다.

스크립트가 성공적으로 실행되면 종료 코드 0(영)을 반환합니다.

> **Note**
>
> CodeDeploy 에이전트는 AWS Lambda 또는 Amazon ECS 배포에 사용되지 않습니다.

설치 이벤트 중에 CodeDeploy 에이전트는 AppSpec 파일의 files 섹션에 정의된 매핑을 사용하여 개정에서 인스턴스로 복사할 폴더 또는 파일을 결정합니다.

운영 체제에 설치된 CodeDeploy 에이전트가 AppSpec 파일에 나열된 항목과 일치하지 않으면 배포가 실패합니다.

CodeDeploy 에이전트 로그 파일에 대한 자세한 내용은 CodeDeploy 에이전트 작업 단원을 참조하세요.

CodeDeploy 시작하기

주제

- 1단계: 설정
- 2단계: CodeDeploy에 대한 서비스 역할 생성
- 3단계: CodeDeploy 사용자 권한 제한
- 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기

1단계: 설정

AWS CodeDeploy 를 처음 사용하기 전에 설정 단계를 완료해야 합니다. 이 단계에는 AWS 계정 생성(계정이 아직 없는 경우)과 관리 사용자가 포함됩니다.

이 안내서에서는 관리 사용자를 CodeDeploy 관리 사용자라고 합니다.

에 가입 AWS 계정

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

> **Important**
>
> AWS CLI를 사용하여 CodeDeploy 관리 사용자를 작업 인력 ID(IAM Identity Center에서 관리되는 사용자)로 구성하는 것이 좋습니다. 이 안내서의 많은 절차에서는를 사용하여 구성을 수행하고 있다고 가정 AWS CLI 합니다.

> **Important**
>
> 를 구성하는 경우 AWS 리전을 지정하라는 메시지가 AWS CLI표시될 수 있습니다. AWS 일반 참조의 리전 및 엔드포인트에 나열된 지원되는 리전 중 하나를 선택합니다.

2단계: CodeDeploy에 대한 서비스 역할 생성

에서 AWS서비스 역할은 AWS 리소스에 액세스할 수 있도록 AWS 서비스에 권한을 부여하는 데 사용됩니다. 서비스 역할에 연결하는 정책에 따라 서비스에 액세스할 수 있는 리소스와 서비스가 해당 리소스로 수행할 수 있는 작업이 결정됩니다.

CodeDeploy에 대해 생성하는 서비스 역할에는 컴퓨팅 플랫폼에 필요한 권한을 부여해야 합니다. 둘 이상의 컴퓨팅 플랫폼에 배포하는 경우 플랫폼별로 서비스 역할을 하나씩 생성합니다. 권한을 추가하려면 다음 AWS 제공 정책 중 하나 이상을 연결합니다.

EC2/온프레미스 배포의 경우 **`AWSCodeDeployRole`** 정책을 연결합니다. 이 정책은 서비스 역할에서 다음을 수행하는 데 필요한 권한을 제공합니다.

- Amazon EC2 Auto Scaling 그룹 이름으로 인스턴스의 태그를 읽거나 Amazon EC2 인스턴스를 식별합니다.
- Amazon EC2 Auto Scaling 그룹, 수명 주기 후크 및 크기 조정 정책을 읽고, 생성하고, 업데이트하고, 삭제합니다.
- Amazon SNS 주제로 정보를 게시합니다.
- CloudWatch 경보에 관한 정보를 검색합니다.
- Elastic Load Balancing을 읽고 업데이트합니다.

> **Note**
>
> 시작 템플릿으로 Auto Scaling 그룹을 생성한 경우 다음 권한을 추가해야 합니다.
>
> - `ec2:RunInstances`
> - `ec2:CreateTags`
> - `iam:PassRole`
>
> 자세한 내용은 2단계: 서비스 역할 생성, Auto Scaling 그룹을 위한 시작 템플릿 만들기 및 Amazon EC2 Auto Scaling 사용 설명서의 시작 템플릿 지원을 참조하세요.

Amazon ECS 배포에 대해 지원 서비스에 대한 전체 액세스 권한을 부여하려는 경우 **`AWSCodeDeployRoleForECS`** 정책을 연결합니다. 이 정책은 서비스 역할에서 다음을 수행하는 데 필요한 권한을 제공합니다.

- Amazon ECS 작업 세트를 읽고, 업데이트하고, 삭제합니다.
- Elastic Load Balancing 대상그룹, 리스너 및 규칙을 업데이트합니다.

- AWS Lambda 함수를 호출합니다.
- Amazon S3 버킷에 있는 개정 파일에 액세스합니다.
- CloudWatch 경보에 관한 정보를 검색합니다.
- Amazon SNS 주제로 정보를 게시합니다.

Amazon ECS 배포의 경우 서비스 지원을 위한 액세스를 제한하려면 **AWSCodeDeployRoleForECSLimited** 정책을 사용합니다. 이 정책은 서비스 역할에서 다음을 수행하는 데 필요한 권한을 제공합니다.

- Amazon ECS 작업 세트를 읽고, 업데이트하고, 삭제합니다.
- CloudWatch 경보에 관한 정보를 검색합니다.
- Amazon SNS 주제로 정보를 게시합니다.

AWS Lambda 배포의 경우 Amazon SNS에 게시를 허용하려면 **AWSCodeDeployRoleForLambda** 정책을 연결합니다. 이 정책은 서비스 역할에서 다음을 수행하는 데 필요한 권한을 제공합니다.

- AWS Lambda 함수 및 별칭을 읽고 업데이트하고 호출합니다.
- Amazon S3 버킷에 있는 개정 파일에 액세스합니다.
- CloudWatch 경보에 관한 정보를 검색합니다.
- Amazon SNS 주제로 정보를 게시합니다.

AWS Lambda 배포의 경우 Amazon SNS에 대한 액세스를 제한하려면 **AWSCodeDeployRoleForLambdaLimited** 정책을 연결합니다. 이 정책은 서비스 역할에서 다음을 수행하는 데 필요한 권한을 제공합니다.

- AWS Lambda 함수 및 별칭을 읽고 업데이트하고 호출합니다.
- Amazon S3 버킷에 있는 개정 파일에 액세스합니다.
- CloudWatch 경보에 관한 정보를 검색합니다.

서비스 역할을 설정하는 과정에서 액세스 권한을 부여하려는 엔드포인트를 지정하도록 신뢰 관계를 업데이트합니다.

IAM 콘솔 AWS CLI, 또는 IAM APIs.

주제

- 서비스 역할 생성(콘솔)
- 서비스 역할 생성(CLI)
- 서비스 역할 ARN 확인(콘솔)
- 서비스 역할 ARN 확인(CLI)

서비스 역할 생성(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. AWS 서비스를 선택하고 사용 사례의 드롭다운 목록에서 CodeDeploy를 선택합니다.
4. 사용 사례를 선택합니다.
   - EC2/온프레미스 배포의 경우 CodeDeploy를 선택합니다.
   - AWS Lambda 배포의 경우 Lambda용 CodeDeploy를 선택합니다.
   - Amazon ECS 배포의 경우 CodeDeploy - ECS를 선택합니다.
5. 다음을 선택합니다.
6. 권한 추가 페이지에는 사용 사례에 대한 올바른 권한 정책이 표시됩니다. 다음을 선택합니다.
7. 이름 지정, 검토 및 생성 페이지의 역할 이름에 서비스 역할의 이름(예: **`CodeDeployServiceRole`**)을 입력한 후 역할 생성을 선택합니다.

   이 서비스 역할에 대한 설명을 역할 설명(Role description)에 입력할 수도 있습니다.
8. 이 서비스 역할이 현재 지원되는 모든 엔드포인트에 대한 액세스 권한을 갖도록 하려면 여기서 이 절차를 마칩니다.

   이 서비스 역할이 일부 엔드포인트에 액세스하지 못하도록 제한하려면 이 절차의 나머지 단계를 계속 진행합니다.
9. 역할 목록에서 방금 생성한 역할(`CodeDeployServiceRole`)을 검색하고 선택합니다.
10. 신뢰 관계 탭을 선택합니다.
11. 신뢰 정책 편집을 선택합니다.

    다음 정책은 지원되는 모든 엔드포인트에 액세스할 수 있는 권한을 서비스 역할에 부여합니다.

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

```
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

서비스 역할에 지원되는 일부 엔드포인트에 대한 액세스 권한만 부여하려면 신뢰 정책 텍스트 상자의 콘텐츠를 다음 정책으로 바꿉니다. 액세스를 방지하려는 엔드포인트 행을 제거한 후, 정책 업데이트를 선택합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [

                    "codedeploy.us-east-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
```

```
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
                    "codedeploy.me-south-1.amazonaws.com",
                    "codedeploy.sa-east-1.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

서비스 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 권한을 위임할 역할 생성을 참조하세요.

서비스 역할 생성(CLI)

1. 개발 머신에서 텍스트 파일을 작성합니다(예: `CodeDeployDemo-Trust.json`). 이 파일은 사용자를 대신해 CodeDeploy가 작동하도록 허용하는 데 사용됩니다.

   다음 중 하나를 수행하세요.

   - 지원되는 모든 AWS 리전에 대한 액세스 권한을 부여하려면 파일에 다음 콘텐츠를 저장합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "codedeploy.amazonaws.com"
                ]
            },
```

```
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- 지원되는 일부 리전에만 액세스 권한을 부여하려면 파일에 다음 콘텐츠를 입력하고 액세스를 제외하려는 리전의 줄을 제거합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [

                    "codedeploy.us-east-1.amazonaws.com",
                    "codedeploy.us-east-2.amazonaws.com",
                    "codedeploy.us-west-1.amazonaws.com",
                    "codedeploy.us-west-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.ap-east-1.amazonaws.com",
                    "codedeploy.ap-northeast-1.amazonaws.com",
                    "codedeploy.ap-northeast-2.amazonaws.com",
                    "codedeploy.ap-northeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-1.amazonaws.com",
                    "codedeploy.ap-southeast-2.amazonaws.com",
                    "codedeploy.ap-southeast-3.amazonaws.com",
                    "codedeploy.ap-southeast-4.amazonaws.com",
                    "codedeploy.ap-south-1.amazonaws.com",
                    "codedeploy.ap-south-2.amazonaws.com",
                    "codedeploy.ca-central-1.amazonaws.com",
                    "codedeploy.eu-west-1.amazonaws.com",
                    "codedeploy.eu-west-2.amazonaws.com",
                    "codedeploy.eu-west-3.amazonaws.com",
                    "codedeploy.eu-central-1.amazonaws.com",
                    "codedeploy.eu-central-2.amazonaws.com",
                    "codedeploy.eu-north-1.amazonaws.com",
                    "codedeploy.eu-south-1.amazonaws.com",
                    "codedeploy.eu-south-2.amazonaws.com",
                    "codedeploy.il-central-1.amazonaws.com",
                    "codedeploy.me-central-1.amazonaws.com",
```

```
                    "codedeploy.me-south-1.amazonaws.com",
                    "codedeploy.sa-east-1.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

> **Note**
>
> 목록에서 마지막 엔드포인트 뒤에는 쉼표를 사용하지 마세요.

2. 동일한 디렉터리에서 create-role 명령을 호출하여 방금 만든 텍스트 파일의 정보를 바탕으로 **CodeDeployServiceRole**이라는 서비스 역할을 만듭니다.

```
aws iam create-role --role-name CodeDeployServiceRole --assume-role-policy-document
 file://CodeDeployDemo-Trust.json
```

> **Important**
>
> 파일 이름 앞에 `file://`를 포함해야 합니다. 이 명령에 필수적입니다.

명령의 출력에서 `Arn` 객체 아래의 `Role` 항목 값을 적어 둡니다. 이 값은 나중에 배포 그룹을 만들 때 필요합니다. 값을 잊어버린 경우, 서비스 역할 ARN 확인(CLI)의 지침을 따릅니다.

3. 사용하는 관리형 정책은 컴퓨팅 플랫폼에 따라 다릅니다.

   - EC2/온프레미스 컴퓨팅 플랫폼에 배포하는 경우:

     attach-role-policy 명령을 호출하여 **CodeDeployServiceRole**이라는 서비스 역할에 **AWSCodeDeployRole**이라는 IAM 관리형 정책에 기반한 권한을 부여합니다. 예제:

```
aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSCodeDeployRole
```

   - AWS Lambda 컴퓨팅 플랫폼에 배포하는 경우:

attach-role-policy 명령을 호출하여 **CodeDeployServiceRole**이라는 서비스 역할에 **AWSCodeDeployRoleForLambda** 또는 **AWSCodeDeployRoleForLambdaLimited**라는 IAM 관리형 정책에 기반한 권한을 부여합니다. 예:

```
aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSCodeDeployRoleForLambda
```

- Amazon ECS 컴퓨팅 플랫폼에 배포하는 경우:

  attach-role-policy 명령을 호출하여 **CodeDeployServiceRole**이라는 서비스 역할에 **AWSCodeDeployRoleForECS** 또는 **AWSCodeDeployRoleForECSLimited**라는 IAM 관리형 정책에 기반한 권한을 부여합니다. 예:

```
aws iam attach-role-policy --role-name CodeDeployServiceRole --policy-arn
 arn:aws:iam::aws:policy/AWSCodeDeployRoleForECS
```

서비스 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 역할 생성을 참조하세요.

서비스 역할 ARN 확인(콘솔)

IAM 콘솔을 사용하여 서비스 역할의 ARN을 확인하려면:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 필터 상자에 **CodeDeployServiceRole**을 입력한 후 Enter 키를 누릅니다.
4. CodeDeployServiceRole을 선택합니다.
5. 역할 ARN 필드의 값을 기록해 둡니다.

서비스 역할 ARN 확인(CLI)

AWS CLI 를 사용하여 서비스 역할의 ARN을 가져오려면 라는 서비스 역할에 대해 get-role 명령을 호출합니다**CodeDeployServiceRole**.

```
aws iam get-role --role-name CodeDeployServiceRole --query "Role.Arn" --output text
```

반환되는 값은 서비스 역할의 ARN입니다.

3단계: CodeDeploy 사용자 권한 제한

보안상의 이유로 1단계: 설정에서 생성한 관리 사용자의 권한을 CodeDeploy에서 배포를 생성 및 관리하는 데 필요한 권한으로만 제한하는 것이 좋습니다.

CodeDeploy 관리 사용자의 권한을 제한하려면 다음 일련의 절차를 사용합니다.

시작하기 전 준비 사항

- 1단계: 설정의 지침에 따라 IAM Identity Center에서 CodeDeploy 관리 사용자를 생성했는지 확인합니다.

권한 세트를 생성하려면

이 권한 세트는 나중에 CodeDeploy 관리 사용자에게 할당해야 합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/singlesignon/ AWS IAM Identity Center 콘솔을 엽니다.
2. 탐색 창에서 권한 세트를 선택한 다음 권한 세트 생성을 선택합니다.
3. 사용자 지정 권한 세트를 선택합니다.
4. 다음(Next)을 선택합니다.
5. 인라인정책을 선택합니다.
6. 샘플 코드를 제거합니다.
7. 다음 정책 코드를 추가합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CodeDeployAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "autoscaling:*",
                "codedeploy:*",
```

```
                "ec2:*",
                "lambda:*",
                "ecs:*",
                "elasticloadbalancing:*",
                "iam:AddRoleToInstanceProfile",
                "iam:AttachRolePolicy",
                "iam:CreateInstanceProfile",
                "iam:CreateRole",
                "iam:DeleteInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:GetInstanceProfile",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListInstanceProfilesForRole",
                "iam:ListRolePolicies",
                "iam:ListRoles",
                "iam:PutRolePolicy",
                "iam:RemoveRoleFromInstanceProfile",
                "s3:*",
                "ssm:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CodeDeployRolePolicy",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/
CodeDeployServiceRole"
        }
    ]
}
```

이 정책에서 *arn:aws:iam::account-ID:role/CodeDeployServiceRole*을 2단계: CodeDeploy에 대한 서비스 역할 생성에서 생성한 CodeDeploy 서비스 역할의 ARN 값으로 대체합니다. IAM 콘솔의 서비스 역할 세부 정보 페이지에서 ARN 값을 찾을 수 있습니다.

위의 정책을 통해 AWS Lambda 컴퓨팅 플랫폼, EC2/온프레미스 컴퓨팅 플랫폼 및 Amazon ECS 컴퓨팅 플랫폼에 애플리케이션을 배포할 수 있습니다.

이 설명서에 제공된 CloudFormation 템플릿을 사용하여 CodeDeploy와 호환되는 Amazon EC2 인스턴스를 시작할 수 있습니다. CloudFormation 템플릿을 사용하여 애플리케이션, 배포 그룹 또는 배포 구성을 생성하려면 다음과 같이 CodeDeploy 관리 사용자의 권한 정책에 `cloudformation:*` 권한을 추가하여에 CloudFormation 의존하는 CloudFormation AWS 서비스 및 작업에 대한 액세스를 제공해야 합니다.

JSON

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

8. 다음(Next)을 선택합니다.
9. 권한 세트 이름에 다음을 입력합니다.

```
CodeDeployUserPermissionSet
```

10. 다음(Next)을 선택합니다.
11. 검토 및 생성 페이지에서 정보를 검토하고 생성을 선택합니다.

이 권한 세트를 CodeDeploy 관리 사용자에게 할당하려면

1. 탐색 창에서 AWS 계정를 선택한 다음 현재 로그인 AWS 계정 한 옆의 확인란을 선택합니다.
2. 사용자 또는 그룹 할당 버튼을 선택합니다.
3. 사용자 탭을 선택합니다.
4. CodeDeploy 관리 사용자 옆의 확인란을 선택합니다.
5. 다음(Next)을 선택합니다.
6. `CodeDeployUserPermissionSet` 옆의 확인란을 선택합니다.

7. 다음(Next)을 선택합니다.
8. 정보를 검토하고 제출을 선택합니다.

   이제 CodeDeploy 관리 사용자와를 `CodeDeployUserPermissionSet`에 할당하여 함께 AWS 계정바인딩했습니다.

로그아웃했다가 CodeDeploy 관리 사용자로 다시 로그인하려면

1. 로그아웃하기 전에 CodeDeploy 관리 사용자의 AWS 액세스 포털 URL과 사용자 이름 및 일회용 암호가 있는지 확인합니다.

   > **Note**
   >
   > 이 정보가 없는 경우 IAM Identity Center의 CodeDeploy 관리 사용자 세부 정보 페이지로 이동하여 암호 재설정, 일회용 암호 생성 [...]을 선택한 후 암호 재설정을 다시 선택하면 화면에 해당 정보가 표시됩니다.

2. 로그아웃합니다 AWS.
3. AWS 액세스 포털 URL을 브라우저의 주소 표시줄에 붙여 넣습니다.
4. CodeDeploy 관리 사용자로 로그인합니다.

   화면에 AWS 계정 상자가 나타납니다.
5. AWS 계정을 선택한 다음 CodeDeploy 관리 사용자 및 권한 세트를 할당한 AWS 계정 의 이름을 선택합니다.
6. `CodeDeployUserPermissionSet` 옆에 있는 관리 콘솔을 선택합니다.

   가 AWS Management Console 나타납니다. 이제 제한된 권한을 가진 CodeDeploy 관리 사용자로 로그인되었습니다. 이제 이 사용자로 오직 CodeDeploy 관련 작업만 수행할 수 있습니다.

4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기

> **Note**
>
> Amazon ECS 또는 AWS Lambda 컴퓨팅 플랫폼를 사용하는 경우이 단계를 건너뜁니다.

Amazon EC2 인스턴스에는 애플리케이션이 저장되는 Amazon S3 버킷 또는 GitHub 리포지토리에 액세스할 수 있는 권한이 필요합니다. CodeDeploy와 호환 가능한 Amazon EC2 인스턴스를 시작하려면 추가 IAM 역할인 인스턴스 프로파일을 만들어야 합니다. 다음 지침은 Amazon EC2 인스턴스에 연결할 IAM 인스턴스 프로파일을 만드는 방법을 안내합니다. 이 역할은 애플리케이션이 저장되는 Amazon S3 버킷 또는 GitHub 리포지토리에 액세스할 수 있는 권한을 CodeDeploy 에이전트에 제공합니다.

AWS CLI, IAM 콘솔 또는 IAM APIs.

> **Note**
>
> IAM 인스턴스 프로파일은 Amazon EC2 인스턴스를 시작할 때 해당 인스턴스에 연결하거나 이전에 시작한 인스턴스에 연결할 수 있습니다. 자세한 내용은 인스턴스 프로파일을 참조하세요.

주제

- Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기(CLI)
- Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기(콘솔)

Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기(CLI)

이 단계에서는 CodeDeploy 시작하기의 처음 세 단계에 있는 지침을 이미 완료한 것으로 가정합니다.

1. 개발 머신에 `CodeDeployDemo-EC2-Trust.json`이라는 텍스트 파일을 만듭니다. 다음 콘텐츠를 붙여 넣어 Amazon EC2가 사용자를 대신하여 작업하도록 허용합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ec2.cn-north-1.amazonaws.com",
                    "ec2.cn-northwest-1.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 동일한 디렉터리에 CodeDeployDemo-EC2-Permissions.json이라는 텍스트 파일을 만듭니다. 다음 콘텐츠를 붙여 넣습니다.

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

Note

Amazon EC2 인스턴스에서 액세스해야 하는 Amazon S3 버킷으로만 이 정책을 제한하는 것이 좋습니다. CodeDeploy 에이전트가 포함된 Amazon S3 버킷에 대한 액세스 권한을 제공해야 합니다. 그렇지 않으면 인스턴스에 CodeDeploy 에이전트를 설치 또는 업데이트할 때 오류가 발생할 수 있습니다. Amazon S3의 일부 CodeDeploy 리소스 키트 버킷에만 IAM 인스턴스 프로파일 액세스 권한을 부여하려면 다음 정책을 사용하되 액세스를 차단하려는 버킷 라인을 제거합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
```

```
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

> **Note**
>
> CodeDeploy와 함께 IAM 권한 부여 또는 Amazon Virtual Private Cloud(VPC) 엔드포인트를 사용하는 경우 권한을 추가해야 합니다. 자세한 내용은 Amazon Virtual Private Cloud에서 CodeDeploy 사용을 참조하세요.

3. 동일한 디렉터리에서 create-role 명령을 호출하여 첫 번째 파일의 정보를 바탕으로 **`CodeDeployDemo-EC2-Instance-Profile`**이라는 IAM 역할을 생성하세요.

> **Important**
>
> 파일 이름 앞에 `file://`를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws iam create-role --role-name CodeDeployDemo-EC2-Instance-Profile --assume-role-policy-document file://CodeDeployDemo-EC2-Trust.json
```

4. 동일한 디렉터리에서 put-role-policy 명령을 호출하여 두 번째 파일의 정보를 바탕으로 **`CodeDeployDemo-EC2-Instance-Profile`**이라는 역할을 제공하세요.

> **Important**
>
> 파일 이름 앞에 `file://`를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws iam put-role-policy --role-name CodeDeployDemo-EC2-Instance-Profile --policy-
name CodeDeployDemo-EC2-Permissions --policy-document file://CodeDeployDemo-EC2-
Permissions.json
```

5. attach-role-policy을(를) 호출하여 SSM이 CodeDeploy 에이전트를 설치할 수 있도록 Amazon EC2 Systems Manager 권한을 이 역할에 부여합니다. 명령줄을 사용하여 퍼블릭 Amazon S3 버킷에서 에이전트를 설치하려는 경우에는 이 정책이 필요하지 않습니다. 자세한 내용은 CodeDeploy 에이전트 설치를 참조하세요.

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   AmazonSSMManagedInstanceCore --role-name CodeDeployDemo-EC2-Instance-Profile
   ```

6. create-instance-profile 명령을 호출한 후 add-role-to-instance-profile 명령을 호출하여 **CodeDeployDemo-EC2-Instance-Profile**이라는 IAM 인스턴스 프로파일을 생성하세요. 인스턴스 프로필을 사용하면 인스턴스가 처음 시작될 때 Amazon EC2가 Amazon EC2 인스턴스에 **CodeDeployDemo-EC2-Instance-Profile**이라는 IAM 역할을 전달할 수 있습니다.

   ```
   aws iam create-instance-profile --instance-profile-name CodeDeployDemo-EC2-
   Instance-Profile
   aws iam add-role-to-instance-profile --instance-profile-name CodeDeployDemo-EC2-
   Instance-Profile --role-name CodeDeployDemo-EC2-Instance-Profile
   ```

   IAM 인스턴스 프로파일의 이름을 가져와야 하는 경우 AWS CLI 참조의 list-instance-profiles-for-role을 참조하세요.

이제 Amazon EC2 인스턴스에 연결할 IAM 인스턴스 프로파일을 생성하였습니다. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2의 IAM 역할을 참조하세요.

Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 정책을 선택한 후 정책 생성을 선택하세요.
3. 권한 지정 페이지에서 JSON을 선택합니다.
4. 예제 JSON 코드를 제거합니다.
5. 다음 코드를 붙여넣습니다.

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

Note

Amazon EC2 인스턴스에서 액세스해야 하는 Amazon S3 버킷을 대상으로만 이 정책 적용을 제한하는 것이 좋습니다. CodeDeploy 에이전트가 포함된 Amazon S3 버킷에 대한 액세스 권한을 제공해야 합니다. 그렇지 않으면 인스턴스에 CodeDeploy 에이전트를 설치 또는 업데이트할 때 오류가 발생할 수 있습니다. Amazon S3의 일부 CodeDeploy 리소스 키트 버킷에만 IAM 인스턴스 프로파일 액세스 권한을 부여하려면 다음 정책을 사용하되 액세스를 차단하려는 버킷 라인을 제거합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
```

```
            "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
            "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
            "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
            "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
            "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
            "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
            "arn:aws:s3:::aws-codedeploy-il-central-1/*",
            "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
            "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
            "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
            "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
            "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
            "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
            "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
            "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
            "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
            "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
            "arn:aws:s3:::aws-codedeploy-me-central-1/*",
            "arn:aws:s3:::aws-codedeploy-me-south-1/*",
            "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
        ]
    }
  ]
}
```

> **Note**
>
> CodeDeploy와 함께 IAM 권한 부여 또는 Amazon Virtual Private Cloud(VPC) 엔드포인트를 사용하는 경우 권한을 추가해야 합니다. 자세한 내용은 Amazon Virtual Private Cloud에서 CodeDeploy 사용을 참조하세요.

6. 다음을 선택합니다.
7. 검토 및 생성 페이지에서 정책 이름에 **CodeDeployDemo-EC2-Permissions**를 입력합니다.
8. (선택 사항) 설명에 정책에 대한 설명을 입력합니다.
9. 정책 생성을 선택합니다.
10. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
11. 사용 사례에서 EC2 사용 사례를 선택합니다.
12. 다음을 선택합니다.

13. 정책 목록에서 방금 생성한 정책(CodeDeployDemo-EC2-Permissions) 옆의 확인란을 선택합니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다.
14. Systems Manager를 사용하여 CodeDeploy 에이전트를 설치하거나 구성하려면 AmazonSSMManagedInstanceCore 옆의 확인란을 선택합니다. 이 AWS 관리형 정책을 사용하면 인스턴스가 Systems Manager 서비스 핵심 기능을 사용할 수 있습니다. 필요한 경우 검색 상자를 사용하여 정책을 찾습니다. 명령줄을 사용하여 퍼블릭 Amazon S3 버킷에서 에이전트를 설치하려는 경우에는 이 정책이 필요하지 않습니다. 자세한 내용은 CodeDeploy 에이전트 설치를 참조하세요.
15. 다음을 선택합니다.
16. 이름 지정, 검토 및 생성 페이지의 역할 이름에 서비스 역할의 이름(예: **CodeDeployDemo-EC2-Instance-Profile**)을 입력한 후 역할 생성을 선택합니다.

    이 서비스 역할에 대한 설명을 역할 설명(Role description)에 입력할 수도 있습니다.

이제 Amazon EC2 인스턴스에 연결할 IAM 인스턴스 프로파일을 생성하였습니다. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2의 IAM 역할을 참조하세요.

CodeDeploy와 제품 및 서비스 통합

기본적으로 CodeDeploy는 여러 AWS 서비스 및 파트너 제품 및 서비스와 통합됩니다. 다음 정보를 통해 사용 중인 제품과 서비스를 통합할 CodeDeploy를 구성할 수 있습니다.

- 다른 AWS 서비스와의 통합
- 파트너 제품 및 서비스와의 통합
- 커뮤니티의 통합 예제

다른 AWS 서비스와의 통합

CodeDeploy는 다음 AWS 서비스와 통합됩니다.

Amazon CloudWatch	Amazon CloudWatch는 AWS 클라우드 리소스 및 AWS에서 실행하는 애플리케이션을 모니터링하는 서비스입니다. Amazon CloudWatch를 사용하여 지표를 수집 및 추적하고, 로그 파일을 수집 및 모니터링하고, 경보를 설정할 수 있습니다. CodeDeploy는 다음 CloudWatch 도구를 지원합니다. • CloudWatch 경보는 배포를 모니터링하고, 지정한 모니터링 지표가 CloudWatch 경보 규칙에서 지정한 임계값을 초과하거나 임계값 미만인 경우 해당 배포를 중지시킵니다. 경보 모니터링을 사용하려면 먼저 CloudWatch에서 경보를 설정한 다음 CodeDeploy에서 경보 활성화 시 배포를 중지해야 하는 애플리케이션 또는 배포 그룹에 경보를 추가합니다. 자세히 알아보기: • CloudWatch Logs 경보 생성 • Amazon CloudWatch Events는 CodeDeploy 작업에서 인스턴스 또는 배포의 상태 변화를 감지하고 그에 대응합니다. 그런 다음

CloudWatch Events는 사용자가 만든 규칙에 따라 배포 또는 인스턴스가 규칙에 지정된 상태로 전환되면 하나 이상의 대상 작업을 호출합니다.

자세히 알아보기:

- Amazon CloudWatch Events를 사용하여 배포 모니터링

- Amazon CloudWatch Logs는 인스턴스에 한 번에 하나씩 로그인할 필요 없이 CodeDeploy 에이전트에서 생성한 3가지 로그 유형을 모니터링합니다.

  자세히 알아보기:

  - CloudWatch Logs 콘솔에서 CodeDeploy 로그 보기

Amazon EC2 Auto Scaling	CodeDeploy는 Amazon EC2 Auto Scaling을 지원합니다. 이 AWS 서비스는 지정한 기준에 따라 Amazon EC2 인스턴스를 자동으로 시작할 수 있습니다. 예를 들면 다음과 같습니다. • 지정된 CPU 사용률의 한도 초과 • 디스크 읽기 또는 쓰기 • 지정된 시간 간격 동안의 인바운드 또는 아웃바운드 네트워크 트래픽 필요할 때 언제든지 Amazon EC2 인스턴스 그룹을 확장한 다음 CodeDeploy를 사용하여 애플리케이션 개정 버전을 자동으로 배포할 수 있습니다. Amazon EC2 Auto Scaling은 더 이상 필요하지 않은 경우 해당 Amazon EC2 인스턴스를 종료합니다. 자세히 알아보기: • Amazon EC2 Auto Scaling과 CodeDeploy 통합 • 튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포 • 세부 정보: CodeDeploy 및 Auto Scaling 통합

Amazon Elastic Container Service	CodeDeploy를 사용하여 Amazon ECS 컨테이너화된 애플리케이션을 작업 세트로 배포할 수 있습니다. CodeDeploy는 애플리케이션의 업데이트 버전을 새로운 대체 작업 세트로 설치하여 블루/그린 배포를 수행합니다. CodeDeploy는 프로덕션 트래픽을 원래 애플리케이션 작업 세트에서 대체 작업 세트로 다시 라우팅합니다. 배포가 성공하면 기존 작업 세트는 종료됩니다. Amazon ECS에 대한 자세한 내용은 Amazon Elastic Container Service를 참조하세요. 카나리(Canary), 선형(Linear) 또는 한번에 모두(All-at-once) 구성을 선택하여 배포 중 업데이트된 작업 세트로 트래픽을 전송하는 방식을 관리할 수 있습니다. Amazon ECS 배포에 대한 자세한 내용은 Amazon ECS 컴퓨팅 플랫폼에 배포를 참조하세요.

AWS CloudTrail	CodeDeploy는 AWS CloudTrail과 통합되어 있습니다. 이 서비스는 AWS 계정에서 CodeDeploy에 의해 또는 CodeDeploy를 대신하여 수행된 API 호출을 캡처하고 사용자가 지정하는 Amazon S3 버킷에 로그 파일을 전송합니다. CloudTrail은 CodeDeploy 콘솔, AWS CLI를 통한 CodeDeploy 명령 또는 CodeDeploy API에서 직접 API 호출을 캡처합니다. CloudTrail에서 수집한 정보를 사용하여 다음을 확인할 수 있습니다. • CodeDeploy에 요청한 내용 • 요청을 보낸 소스 IP 주소 • 요청한 사람 • 요청한 시기 자세히 알아보기: • Monitoring Deployments

AWS Cloud9	AWS Cloud9는 인터넷에 연결된 시스템의 브라우저만 사용하여 코드를 작성, 실행, 디버깅 및 배포하는 데 사용할 수 있는 온라인 클라우드 기반 통합 개발 환경(IDE)입니다. 에는 AWS CLI 및 Git과 같은 코드 편집기, 디버거, 터미널 및 필수 도구가 AWS Cloud9 포함되어 있습니다. • IDE를 사용하여 GitHub AWS Cloud9 리포지토리에 있는 코드를 실행, 디버깅 및 빌드할 수 있습니다. IDE 환경 창과 편집기 탭을 이용해 코드를 보고, 변경하고, 저장할 수 있습니다. 준비가 되면 AWS Cloud9 터미널 세션에서 Git을 사용하여 코드 변경 사항을 GitHub 리포지토리로 푸시한 다음 AWS CodeDeploy 를 사용하여 업데이트를 배포할 수 있습니다. GitHub AWS Cloud9 에서를 사용하는 방법에 대한 자세한 내용은 GitHub 샘플을 참조하세요 AWS Cloud9. • AWS Cloud9 IDE를 사용하여 AWS Lambda 함수를 업데이트할 수 있습니다. 그런 다음 AWS CodeDeploy 를 사용하여 트래픽을 AWS Lambda 함수의 새 버전으로 전환하는 배포를 생성할 수 있습니다. 자세한 내용은 AWS Cloud9 통합 개발 환경(IDE)의 AWS Lambda 함수 작업을 참조하세요. 에 대한 자세한 내용은 정의 AWS Cloud9 및 시작하기를 AWS Cloud9참조하세요. AWS Cloud9

AWS CodePipeline	AWS CodePipeline은 지속적인 전송 프로세스에서 소프트웨어 출시에 필요한 단계를 모델링, 시각화 및 자동화하는 데 사용할 수 있는 지속적인 전송 서비스입니다. AWS CodePipeline 을 사용하면 고유한 릴리스 프로세스를 정의할 수 있습니다. 따라서 해당 서비스에서 코드 변경이 발생할 때마다 코드를 빌드, 테스트, 배포합니다. 예를 들어, 하나의 애플리케이션에 대해 Beta, Gamma, Prod의 세 가지 배포 그룹이 있을 수 있습니다. 소스 코드 변경 시마다 업데이트된 내용이 각 배포 그룹에 하나씩 배포되도록 파이프라인을 설정할 수 있습니다. CodeDeploy를 사용하여 배포 AWS CodePipeline 하도록를 구성할 수 있습니다. • Amazon EC2 인스턴스, 온프레미스 인스턴스 또는 둘 다에 대한 코드 • 서버리스 AWS Lambda 함수 버전. 파이프라인을 생성하기 전 단계나 파이프라인 생성(Create Pipeline) 마법사에서 배포 작업에 사용할 CodeDeploy 애플리케이션, 배포 및 배포 그룹을 생성할 수 있습니다. 자세히 알아보기: • DevOps를 위한AWS 시작 가이드 - CodeDeploy와 함께 CodePipeline을 사용하여 CodeCommit 리포지토리의 소스 코드를 Amazon EC2 인스턴스로 지속적으로 전달하고 배포하는 방법을 알아봅니다. • 간단한 파이프라인 시연(Amazon S3 버킷) • 간단한 파이프라인 시연(CodeCommit 리포지토리)

	• 4단계 파이프라인 자습서
AWS 서버리스 애플리케이션 모델	AWS 서버리스 애플리케이션 모델(AWS SAM)은 서버리스 애플리케이션을 정의하는 모델입니다. 서버리스 애플리케이션에 필요한 함수, Amazon API Gateway APIs 및 Amazon DynamoDB 테이블을 정의하는 AWS Lambda 간소화된 방법을 CloudFormation 제공하도록 확장되었습니다. 이미 AWS SAM을 사용하는 경우 배포 기본 설정을 추가하여 CodeDeploy를 사용하여 AWS Lambda 애플리케이션 배포 중에 트래픽이 이동하는 방식을 관리할 수 있습니다. 자세한 내용은 AWS Serverless Application Model을 참조하세요.
Elastic Load Balancing	CodeDeploy는 수신 애플리케이션 트래픽을 여러 Amazon EC2 인스턴스로 분산하는 서비스인 Elastic Load Balancing을 지원합니다. CodeDeploy 배포 시, 로드 밸런서는 준비되지 않았거나, 현재 배포 중이거나, 더 이상 환경의 일부로 필요하지 않은 인스턴스로 트래픽이 라우팅되지 않도록 합니다. 자세히 알아보기: • Integrating CodeDeploy with Elastic Load Balancing

주제

- Amazon EC2 Auto Scaling과 CodeDeploy 통합
- CodeDeploy와 Elastic Load Balancing 통합

Amazon EC2 Auto Scaling과 CodeDeploy 통합

CodeDeploy는 정의한 조건에 따라 Amazon EC2 인스턴스를 자동으로 시작하는 AWS 서비스인 Amazon EC2 Auto Scaling을 지원합니다. 이러한 조건에는 지정된 시간 간격에서 CPU 사용률, 디스크 읽기 또는 쓰기, 인바운드 또는 아웃바운드 네트워크 트래픽에 대해 초과되는 제한이 포함될 수 있습니다. Amazon EC2 Auto Scaling은 더 이상 필요하지 않은 경우 해당 인스턴스를 종료합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon EC2 Auto Scaling이란?을 참조하세요.

새 Amazon EC2 인스턴스가 Amazon EC2 Auto Scaling 그룹의 일부로 시작되면 CodeDeploy에서는 개정을 새 인스턴스로 자동 배포할 수 있습니다. 또한 Elastic Load Balancing 로드 밸런서에 등록된 Amazon EC2 Auto Scaling 인스턴스를 사용하여 CodeDeploy에서 배포를 조정할 수도 있습니다. 자세한 내용은 Integrating CodeDeploy with Elastic Load Balancing 및 Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정 섹션을 참조하세요.

> **Note**
>
> 여러 배포 그룹과 단일 Amazon EC2 Auto Scaling 그룹을 연결하는 경우 문제가 발생할 수 있습니다. 한 개의 배포에 실패하면 예를 들어, 인스턴스가 종료되기 시작하지만 실행 중인 다른 배포는 시간 초과까지 한 시간 가량 걸릴 수 있습니다. 자세한 내용은 단일 Amazon EC2 Auto Scaling 그룹으로 여러 배포 그룹 연결 피하기 및 세부 정보: CodeDeploy 및 Amazon EC2 Auto Scaling 통합을 참조하세요.

주제

- Amazon EC2 Auto Scaling 그룹에 CodeDeploy 애플리케이션 배포
- Auto Scaling 확장 이벤트 중 종료 배포 활성화
- Amazon EC2 Auto Scaling에서 CodeDeploy를 사용하는 방식
- CodeDeploy 및 Amazon EC2 Auto Scaling에서 사용자 지정 AMI 사용

Amazon EC2 Auto Scaling 그룹에 CodeDeploy 애플리케이션 배포

Amazon EC2 Auto Scaling 그룹에 CodeDeploy 애플리케이션 개정 버전을 배포하려면 다음을 수행하세요.

1. Amazon EC2 Auto Scaling 그룹이 Amazon S3와 연동할 수 있도록 허용하는 IAM 인스턴스 프로파일을 만들거나 찾습니다. 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 단원을 참조하십시오.

> **Note**
>
> CodeDeploy를 사용하여 GitHub 리포지토리에서 Amazon EC2 Auto Scaling 그룹으로 개정을 배포할 수도 있습니다. Amazon EC2 인스턴스는 IAM 인스턴스 프로파일을 계속 필요로 하지만, 이 프로파일에는 GitHub 리포지토리에서 배포하기 위한 추가 권한이 필요하지 않습니다.

2. 시작 구성이나 템플릿에서 IAM 인스턴스 프로파일을 지정하여 Amazon EC2 Auto Scaling 그룹을 생성하거나 사용합니다. 자세한 내용은 Amazon EC2 인스턴스에서 실행되는 애플리케이션의 IAM 역할을 참조하세요.
3. CodeDeploy가 Amazon EC2 Auto Scaling 그룹이 포함된 배포 그룹을 생성하도록 허용하는 서비스 역할을 만들거나 찾습니다.
4. Amazon EC2 Auto Scaling 그룹 이름, 서비스 역할 및 기타 몇 가지 옵션을 지정하여 CodeDeploy로 배포 그룹을 생성합니다. 자세한 내용은 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 또는 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)을 참조하세요.
5. CodeDeploy를 사용하여 Amazon EC2 Auto Scaling 그룹이 포함된 배포 그룹에 수정 버전을 배포합니다.

자세한 내용은 튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포 단원을 참조하십시오.

Auto Scaling 확장 이벤트 중 종료 배포 활성화

종료 배포는 Auto Scaling 스케일 인 이벤트가 발생하면 자동으로 활성화되는 CodeDeploy 배포의 한 유형입니다. CodeDeploy는 Auto Scaling 서비스가 인스턴스를 종료하기 직전에 종료 배포를 수행합니다. 종료 배포 중에는 CodeDeploy는 아무것도 배포하지 않습니다. 대신 수명 주기 이벤트를 생성하며, 이를 자체 스크립트에 연결하여 사용자 지정 종료 기능을 활성화할 수 있습니다. 예를 들어, 인스턴스가 종료되기 전에 애플리케이션을 정상적으로 종료하는 스크립트에 `ApplicationStop` 수명 주기 이벤트를 연결할 수 있습니다.

종료 배포 중에 CodeDeploy가 생성하는 수명 주기 이벤트 목록은 수수명 주기 이벤트 후크 가용성을 참조하세요.

종료 배포가 어떤 이유로든 실패하면 CodeDeploy는 인스턴스 종료를 계속 진행하도록 허용합니다. 즉, CodeDeploy가 수명 주기 이벤트의 전체 집합(또는 일부)을 완료까지 실행하지 않았더라도 인스턴스가 종료됩니다.

종료 배포를 활성화하지 않으면 Auto Scaling 서비스는 스케일인 이벤트가 발생할 때 Amazon EC2 인스턴스를 계속 종료하지만 CodeDeploy는 수명 주기 이벤트를 생성하지 않습니다.

> **Note**
>
> 종료 배포를 사용하도록 설정했는지 여부와 관계없이 자동 확장 서비스가 CodeDeploy 배포가 진행 중인 동안 Amazon EC2 인스턴스를 종료하면 자동 확장 및 CodeDeploy 서비스에서 생성된 수명 주기 이벤트 간에 경합 조건이 발생할 수 있습니다. 예를 들어, Auto Scaling 서비스에 의해 생성된 `Terminating` 수명 주기 이벤트가 CodeDeploy 배포에 의해 생성된 `ApplicationStart` 이벤트보다 우선할 수 있습니다. 이 시나리오에서는 Amazon EC2 인스턴스 종료 또는 CodeDeploy 배포에 실패가 발생할 수 있습니다.

CodeDeploy가 종료 배포를 수행하도록 하려면 다음을 수행하세요.

- 배포 그룹을 생성하거나 업데이트할 때 Auto Scaling 그룹에 종료 후크 추가 확인란을 선택합니다. 자세한 내용은 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 또는 EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)을 참조하세요.

  이 확인란을 활성화하면 CodeDeploy 배포 그룹을 만들거나 업데이트할 때 지정한 Auto Scaling 그룹에 자동 확장 수명 주기 후크가 설치됩니다. 이 후크를 종료 후크라고 하며 종료 배포를 활성화합니다.

종료 후크가 설치되면 다음과 같이 스케일 인(종료) 이벤트가 전개됩니다.

1. Auto Scaling 서비스(또는 간단히 자동 확장)는 스케일 인 이벤트가 발생해야 한다고 판단하고 EC2 인스턴스를 종료하기 위해 EC2 서비스에 연락합니다.
2. EC2 서비스가 EC2 인스턴스를 종료하기 시작합니다. 인스턴스는 `Terminating` 상태가 되고 그 다음엔 `Terminating:Wait` 상태가 됩니다.
3. `Terminating:Wait` 동안 Auto Scaling은 CodeDeploy에서 설치된 종료 후크를 포함한 Auto Scaling 그룹에 연결된 모든 수명 주기 후크를 실행합니다.
4. 종료 후크가 알림을 CodeDeploy에 의해 폴링된 Amazon SQS 대기열로 보냅니다.
5. 알림을 받으면 CodeDeploy는 메시지를 구문 분석하고 몇 가지 유효성 검사를 수행한 후 종료 배포를 수행합니다.
6. 종료 배포가 실행되는 동안 CodeDeploy 5분마다 하트비트를 Auto Scaling으로 전송하여 인스턴스가 여전히 작업 중임을 알립니다.

7. 지금까지 EC2 인스턴스는 여전히 `Terminating:Wait` 상태(또는 Auto Scaling group 워밍 풀을 활성화한 경우 `Warmed:Pending:Wait` 상태일 수 있음)에 있습니다.
8. 배포가 완료되면 종료 배포의 성공 또는 실패 여부에 관계없이 Auto Scaling에 EC2 종료 프로세스를 계속 진행하도록 CodeDeploy가 표시합니다.

Amazon EC2 Auto Scaling에서 CodeDeploy를 사용하는 방식

Auto Scaling 그룹을 포함하도록 CodeDeploy 배포 그룹을 생성하거나 업데이트하면 CodeDeploy는 CodeDeploy 서비스 역할을 사용하여 Auto Scaling 그룹에 액세스한 다음 Auto Scaling 수명 주기 후크를 Auto Scaling 그룹에 설치합니다.

> **Note**
>
> Auto Scaling 수명 주기 후크는 CodeDeploy에 의해 생성되고 이 안내서의 AppSpec '후크' 섹션에 설명된 수명 주기 이벤트(수명 주기 이벤트 후크라고도 함)와는 다릅니다.

CodeDeploy가 설치하는 Auto Scaling 수명 주기 훅은 다음과 같습니다.

- 시작 후크 - 이 후크는 Auto Scaling 스케일 아웃 이벤트가 진행 중이며 CodeDeploy가 시작 배포를 시작해야 한다는 것을 CodeDeploy에 알립니다.

  시작 배포 중에 CodeDeploy는 다음을 수행합니다.
  - 애플리케이션의 개정 버전을 스케일 아웃 인스턴스에 배포합니다.
  - 배포 진행 상황을 나타내는 수명 주기 이벤트를 생성합니다. 이러한 수명 주기 이벤트를 자체 스크립트에 연결하여 사용자 지정 시작 기능을 활성화할 수 있습니다. 자세한 내용은 수명 주기 이벤트 후크 가용성 표를 참조하세요.

  시작 후크 및 관련 시작 배포는 항상 활성화되어 있으며 끌 수 없습니다.
- 종료 후크 - 이 선택적 후크는 Auto Scaling 스케일 인 이벤트가 진행 중이며 CodeDeploy가 종료 배포를 시작해야 함을 CodeDeploy에 알립니다.

  종료 배포 중에 CodeDeploy는 수명 주기 이벤트를 생성하여 인스턴스 종료의 진행 상황을 나타냅니다. 자세한 내용은 Auto Scaling 확장 이벤트 중 종료 배포 활성화 단원을 참조하십시오.

주제

- CodeDeploy가 수명 주기 훅을 설치한 후에는 어떻게 사용되나요?

- CodeDeploy가 Amazon EC2 Auto Scaling 그룹에 이름을 지정하는 방법
- 사용자 지정 수명 주기 후크의 실행 순서
- 배포 중 이벤트 확장
- 배포 중에 스케일 인 이벤트
- AWS CloudFormation cfn-init 스크립트의 이벤트 순서

CodeDeploy가 수명 주기 훅을 설치한 후에는 어떻게 사용되나요?

시작 및 종료 수명 주기 후크가 설치되면 Auto Scaling 그룹 스케일 아웃 및 스케일 인 이벤트 중에 각각 CodeDeploy에서 사용됩니다.

스케일 아웃(시작) 이벤트는 다음과 같이 전개됩니다.

1. Auto Scaling 서비스(또는 단순히 Auto Scaling)가 확장 이벤트가 발생해야 한다고 판단하면 EC2 서비스에 연결하여 새 EC2 인스턴스를 시작합니다.
2. EC2 서비스는 새 EC2 인스턴스를 시작합니다. 인스턴스는 `Pending` 상태가 되고 그 다음엔 `Pending:Wait` 상태가 됩니다.
3. `Pending:Wait` 동안 Auto Scaling은 Auto Scaling 그룹에 연결된 모든 수명 주기 후크를 실행하며, 여기에는 CodeDeploy에 의해 설치된 런치 후크가 포함됩니다.
4. 시작 후크는 CodeDeploy가 폴링하는 Amazon SQS 대기열에 알림을 보냅니다.
5. 알림을 받으면 CodeDeploy는 메시지를 구문 분석하고 몇 가지 유효성 검사를 수행한 후 시작 배포를 시작합니다.
6. 시작 배포가 실행되는 동안 CodeDeploy 5분마다 하트비트를 Auto Scaling으로 전송하여 인스턴스가 여전히 작업 중임을 알립니다.
7. 지금까지 EC2 인스턴스는 여전히 `Pending:Wait` 상태입니다.
8. 배포가 완료되면 CodeDeploy는 배포 성공 또는 실패 여부에 따라 Auto Scaling에 `CONTINUE` 또는 `ABANDON` EC2 시작 프로세스를 알립니다.
   - CodeDeploy가 `CONTINUE`를 알리면 Auto Scaling은 시작 프로세스를 이어가서 다른 후크가 완료될 때까지 기다리거나 인스턴스를 `Pending:Proceed` 상태로 설정한 다음 `InService` 상태로 설정합니다.
   - CodeDeploy가 `ABANDON`을 알리면 Auto Scaling은 EC2 인스턴스를 종료하고 Auto Scaling의 원하는 용량(Desired Capacity) 설정에 정의된 인스턴스 수를 충족하기 위해 필요한 경우 시작 절차를 다시 시작합니다.

스케일 인(종료) 이벤트는 다음과 같이 전개됩니다.

Auto Scaling 확장 이벤트 중 종료 배포 활성화을(를) 참조하세요.

CodeDeploy가 Amazon EC2 Auto Scaling 그룹에 이름을 지정하는 방법

EC2/온프레미스 컴퓨팅 플랫폼을 기반으로 한 블루/그린 배포 과정에서는 인스턴스를 대체(그린) 환경에 추가하는 옵션이 두 가지입니다.

- 이미 존재하거나, 혹은 수동으로 생성한 인스턴스를 사용합니다.
- 지정한 Amazon EC2 Auto Scaling 그룹의 설정을 사용하여 새 Amazon EC2 Auto Scaling 그룹에서 인스턴스를 정의하고 생성합니다.

두 번째 옵션을 선택하면 CodeDeploy는 새로운 Amazon EC2 Auto Scaling 그룹을 프로비저닝합니다. 그룹 이름을 지정할 때 사용하는 규칙은 다음과 같습니다.

```
CodeDeploy_deployment_group_name_deployment_id
```

예를 들어 ID가 10인 배포를 통해 alpha-deployments라는 이름의 배포 그룹을 배포하는 경우 프로비저닝되는 Amazon EC2 Auto Scaling 그룹은 CodeDeploy_alpha-deployments_10이라는 이름으로 지정됩니다. 자세한 내용은 EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 및 GreenFleetProvisioningOption 단원을 참조하세요.

사용자 지정 수명 주기 후크의 실행 순서

CodeDeploy에서 배포하는 Amazon EC2 Auto Scaling 그룹에 고유한 수명 주기 후크를 추가할 수 있습니다. 그러나 사용자 지정 수명 주기 후크 이벤트가 실행되는 순서는 CodeDeploy 기본 배포 수명 주기 이벤트를 기준으로 미리 결정할 수 없습니다. 예를 들어 ReadyForSoftwareInstall이라는 사용자 지정 수명 주기 후크를 Amazon EC2 Auto Scaling 그룹에 추가하면 이 후크가 첫 번째 CodeDeploy 기본 배포 수명 주기 이벤트 이전에 실행될지 또는 마지막 이벤트 이후에 실행될지 사전에 알 수 없습니다.

사용자 지정 수명 주기 후크를 Amazon EC2 Auto Scaling 그룹에 추가하는 방법을 알아보려면 Amazon EC2 Auto Scaling 사용 설명서의 수명 주기 후크 추가를 참조하세요.

배포 중 이벤트 확장

배포가 진행되는 동안 Auto Scaling 스케일 아웃 이벤트가 발생하면 새 인스턴스는 최신 애플리케이션 개정 버전이 아닌 이전에 배포된 애플리케이션 개정 버전으로 업데이트됩니다. 배포에 성공하면 이전

인스턴스와 새로 확장된 인스턴스가 다른 애플리케이션 개정을 호스팅합니다. 이전 개정 버전의 인스턴스를 최신 상태로 유지하기 위해 CodeDeploy는 자동으로 후속 배포를 시작하여(첫 번째 배포 직후에) 오래된 인스턴스를 업데이트합니다. 오래된 EC2 인스턴스가 이전 버전으로 유지되도록 이 기본 동작을 변경하려면 Automatic updates to outdated instances 섹션을 참조하세요.

배포가 진행되는 동안 Amazon EC2 Auto Scaling 확장 프로세스를 일시 중단하려면 CodeDeploy를 통해 로드 밸런싱에 사용되는 `common_functions.sh` 스크립트 설정을 통해 수행합니다. `HANDLE_PROCS=true`이면 배포 프로세스 중 다음 Auto Scaling 이벤트가 자동으로 일시 중단됩니다.

- AZRebalance
- AlarmNotification
- ScheduledActions
- ReplaceUnhealthy

> **Important**
>
> CodeDeployDefault.OneAtATime 배포 구성만 이 기능을 지원합니다.

Amazon EC2 Auto Scaling을 사용할 때 `HANDLE_PROCS=true`를 사용하여 배포 문제를 방지하려면 GitHub에 있는 aws-codedeploy-samples에서 AutoScaling 프로세스 처리에 대한 중요 공지를 참조하세요.

배포 중에 스케일 인 이벤트

해당 Auto Scaling 그룹에서 CodeDeploy 배포가 진행 중인 동안 Auto Scaling 그룹이 확장을 시작하면 종료 프로세스(CodeDeploy 종료 배포 수명 주기 이벤트 포함)와 종료 인스턴스의 다른 CodeDeploy 수명 주기 이벤트 간에 경합 조건이 발생할 수 있습니다. 모든 CodeDeploy 수명 주기 이벤트가 완료되기 전에 인스턴스가 종료되면 해당 특정 인스턴스에 대한 배포가 실패할 수 있습니다. 또한 배포 구성에서 최소 정상 호스트 설정을 어떻게 설정했는지에 따라 전체 CodeDeploy 배포가 실패할 수도 있고 실패하지 않을 수도 있습니다.

AWS CloudFormation cfn-init 스크립트의 이벤트 순서

새로 프로비저닝된 Linux 기반 인스턴스에서 `cfn-init`(또는 `cloud-init`)를 사용하여 스크립트를 실행하는 경우 인스턴스 시작 후 발생하는 이벤트 순서를 엄격하게 제어하지 않으면 배포에 실패할 수 있습니다.

순서는 다음과 같아야 합니다.

1. 새로 프로비저닝된 인스턴스가 시작합니다.
2. 모든 `cfn-init` 부트스트래핑 스크립트가 완료될 때까지 실행됩니다.
3. CodeDeploy 에이전트가 시작됩니다.
4. 최신 애플리케이션 개정이 인스턴스에 배포됩니다.

이벤트 순서를 신중하게 제어하지 않으면 모든 스크립트의 실행이 완료되기 전에 CodeDeploy 에이전트가 배포를 시작할 수 있습니다.

이벤트 순서를 제어하려면 다음 모범 사례 중 하나를 사용합니다.

- `cfn-init` 스크립트를 통해 CodeDeploy 에이전트를 설치합니다. 이때 이 스크립트를 다른 모든 스크립트 뒤에 배치합니다.
- 사용자 정의 AMI에 CodeDeploy 에이전트를 포함하고 `cfn-init` 스크립트를 사용해 시작합니다. 이때 이 스크립트를 다른 모든 스크립트 뒤에 배치합니다.

`cfn-init` 사용에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 cfn-init를 참조하세요.

CodeDeploy 및 Amazon EC2 Auto Scaling에서 사용자 지정 AMI 사용

새 Amazon EC2 인스턴스를 Amazon EC2 Auto Scaling 그룹에서 시작하는 경우 사용할 기본 AMI를 지정하는 데 다음 두 가지 옵션이 있습니다.

- CodeDeploy 에이전트가 이미 설치된 기본 사용자 지정 AMI를 지정할 수 있습니다. 에이전트가 이미 설치되어 있으므로 이 옵션은 다른 옵션보다 훨씬 빨리 새 Amazon EC2 인스턴스를 시작합니다. 그러나 이 옵션은 특히 CodeDeploy 에이전트가 오래된 경우 Amazon EC2 인스턴스의 초기 배포에 실패할 가능성이 더 큽니다. 이 옵션을 선택하면 기본 사용자 지정 AMI에서 CodeDeploy 에이전트를 정기적으로 업데이트하는 것이 좋습니다.
- CodeDeploy 에이전트가 설치되지 않은 기본 AMI를 지정하고 각각의 새 인스턴스가 Amazon EC2 Auto Scaling 그룹에서 시작될 때 이 에이전트가 설치되도록 할 수 있습니다. 이 옵션은 다른 옵션보다 새 Amazon EC2 인스턴스의 시작 속도가 훨씬 느리긴 하지만 초기 인스턴스 배포에 성공할 가능성이 더 큽니다. 이 옵션은 최신 버전의 CodeDeploy 에이전트를 사용합니다.

CodeDeploy와 Elastic Load Balancing 통합

CodeDeploy 배포 시, 로드 밸런서는 준비되지 않았거나, 현재 배포 중이거나, 더 이상 환경의 일부로 필요하지 않은 인스턴스로 인터넷 트래픽이 라우팅되지 않도록 합니다. 그러나 로드 밸런서의 정확한

역할은 블루/그린 배포에 사용되는지 아니면 인 플레이스(in-place) 배포에 사용되는지에 따라 다릅니다.

> **Note**
>
> 블루/그린(Blue/Green) 배포에서는 Elastic Load Balancing 로드 밸런서를 반드시 사용해야 하고, 인 플레이스(In-place) 배포에서는 선택 사항입니다.

Elastic Load Balancing 유형

Elastic Load Balancing은 CodeDeploy 배포에서 사용할 수 있는 Classic Load Balancer, Application Load Balancer, Network Load Balancer의 세 가지 로드 밸런서 유형을 제공합니다.

Classic Load Balancer

전송 계층(TCP/SSL) 또는 애플리케이션 계층(HTTP/HTTPS)에서 라우팅 및 로드 밸런싱합니다. VPC를 지원합니다.

> **Note**
>
> Classic Load Balancer는 Amazon ECS 배포에서 지원되지 않습니다.

Application Load Balancer

애플리케이션 계층(HTTP/HTTPS)에서 라우팅 및 로드 밸런싱하며 경로 기반 라우팅을 지원합니다. Virtual Private Cloud(VPC)에서 각 EC2 인스턴스 또는 컨테이너 인스턴스에서 포트로 요청을 라우팅할 수 있습니다.

> **Note**
>
> Application Load Balancer 대상 그룹의 대상 유형은 EC2 인스턴스에서 배포 시 `instance`이고, Fargate에서 배포 시 `IP`여야 합니다. 자세한 내용은 대상 유형을 참조하세요.

Network Load Balancer

패킷 콘텐츠가 아닌 TCP 패킷 헤더에서 추출된 주소 정보를 기반으로 한 전송 계층(TCP/UDP Layer-4)에서 라우팅 및 로드 밸런싱합니다. Network Load Balancer는 트래픽 급증을 처리하고, 클라이언트의 소스 IP를 유지하며, 로드 밸런서의 수명 동안 고정 IP를 사용할 수 있도록 합니다.

Elastic Load Balancing 로드 밸런서에 대해 자세히 알아보려면 다음 주제를 참조하세요.

- Elastic Load Balancing이란 무엇인가요?
- Classic Load Balancer란 무엇인가요?
- Application Load Balancer란 무엇인가요?
- Network Load Balancer란 무엇인가요?

블루/그린 배포

Elastic Load Balancing 로드 밸런서 뒤에서 인스턴스 트래픽을 다시 라우팅하는 것은 CodeDeploy 블루/그린 배포의 기본 사항입니다.

블루/그린 배포 중 로드 밸런서는 사용자가 지정한 규칙에 따라 최신 애플리케이션 개정이 배포된 배포 그룹(대체 환경)에서 새 인스턴스로 트래픽이 라우팅되도록 한 다음 이전 애플리케이션 개정이 실행 중인 이전 인스턴스(원본 환경)에서의 트래픽은 차단합니다.

대체 환경의 인스턴스가 하나 이상의 로드 밸런서에 등록되면 원래 환경의 인스턴스는 등록 취소되고 선택한 경우 종료됩니다.

블루/그린 배포의 경우 배포 그룹에서 하나 이상의 Classic Load Balancer, Application Load Balancer 대상 그룹 또는 Network Load Balancer 대상 그룹을 지정할 수 있습니다. CodeDeploy 콘솔 또는를 사용하여 배포 그룹에 로드 밸런서를 AWS CLI 추가합니다.

블루/그린 배포에서 로드 밸런서에 대한 자세한 내용은 다음 항목을 참조하세요.

- Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정
- Blue/Green 배포를 위한 애플리케이션 생성(콘솔)
- EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)

인 플레이스(in-place) 배포

인 플레이스(in-place) 배포 시 로드 밸런서는 배포 대상 인스턴스로 인터넷 트래픽이 라우팅되지 않도록 하고 해당 인스턴스에 대한 배포가 완료되면 인스턴스가 다시 트래픽을 처리할 수 있도록 합니다.

인 플레이스(in-place) 배포에서 로드 밸런서를 사용하지 않으면, 배포 프로세스 중 인터넷 트래픽이 인스턴스로 계속 보내질 수 있습니다. 그로 인해 고객에게 웹 애플리케이션이 끊기거나, 완료되지 않거나, 이전 상태로 표시되는 문제가 생길 수 있습니다. 인플레이스 배포에서 Elastic Load Balancing 로드 밸런서를 사용할 경우, 배포 그룹의 인스턴스가 로드 밸런서에서 등록 취소되고 최신 애플리케이션 버전으로 업데이트된 다음 배포가 성공하고 나면 동일한 배포 그룹의 일부로 로드 밸런서에 다시 등록됩니다. CodeDeploy는 인스턴스가 로드 밸런서 뒤에서 정상 상태가 될 때까지 최대 1시간을 기다립니다. 인스턴스가 대기 기간 동안 로드 밸런서에 의해 정상 상태로 표시되지 않으면 CodeDeploy는 배포 구성에 따라 다음 인스턴스로 이동하거나 배포에 실패합니다.

인플레이스 배포의 경우 하나 이상의 Classic Load Balancer, Application Load Balancer 대상 그룹 또는 Network Load Balancer 대상 그룹을 지정할 수 있습니다. 로드 밸런서를 배포 그룹 구성의 일부로 지정하거나, CodeDeploy에서 제공하는 스크립트를 사용하여 로드 밸런서를 구현할 수 있습니다.

배포 그룹을 사용한 인 플레이스(In-place) 배포 로드 밸런서 지정

배포 그룹에 로드 밸런서를 추가하려면 CodeDeploy 콘솔 또는를 사용합니다 AWS CLI. 인 플레이스(in-place) 배포를 위해 배포 그룹에 로드 밸런서를 지정하는 자세한 내용은 다음 항목을 참조하세요.

- 현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔)
- 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)
- Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정

스크립트를 사용한 인 플레이스(In-place) 배포 로드 밸런서 지정

배포 수명 주기 스크립트를 사용하여 인 플레이스(in-place) 배포를 위한 로드 밸런싱을 설정하려면 다음 절차의 단계를 수행합니다.

> **Note**
>
> 인 플레이스 배포에 대한 로드 밸런서를 설정하는 스크립트를 사용 중일 때에만 CodeDeployDefault.OneAtATime 배포 구성을 사용해야 합니다. 동시 실행은 지원되지 않으며, CodeDeployDefault.OneAtATime 설정은 스크립트의 직렬 실행을 보장합니다. 배포 구성에 대한 자세한 내용은 CodeDeploy에서 배포 구성 작업 단원을 참조하세요.

GitHub의 CodeDeploy 샘플 리포지토리에 CodeDeploy Elastic Load Balancing 로드 밸런서를 사용할 때 적용할 수 있는 지침과 샘플이 있습니다. 이 리포지토리에는 진행에 필요한 모든 코드를 제공하는 3가지 샘플 스크립트(`register_with_elb.sh`, `deregister_from_elb.sh`, `common_functions.sh`)가 포함되어 있습니다. 이러한 3개 스크립트에서 자리 표시자를 편집한 후 `appspec.yml` 파일에서 해당 스크립트를 참조하면 됩니다.

CodeDeploy에서 Elastic Load Balancing 로드 밸런서에 등록된 Amazon EC2 인스턴스에 대해 인 플레이스(In-place) 배포를 설정하려면 다음을 수행하세요.

1. 인 플레이스(in-place) 배포에 사용할 로드 밸런서 유형의 샘플을 다운로드합니다.

   - Classic Load Balancer
   - Application Load Balancer 또는 Network Load Balancer(두 유형에 동일한 스크립트를 사용할 수 있음)
2. 각 대상 Amazon EC2 인스턴스에가 AWS CLI 설치되어 있는지 확인합니다.
3. 대상 Amazon EC2 인스턴스 각각에 최소한 elasticloadbalancing:* 및 autoscaling:* 권한을 가진 IAM 인스턴스 프로파일이 연결되어 있는지 확인합니다.
4. 애플리케이션의 소스 코드 디렉터리에 배포 수명 주기 이벤트 스크립트(`register_with_elb.sh`, `deregister_from_elb.sh`, `common_functions.sh`)를 포함합니다.
5. 애플리케이션 수정 버전의 `appspec.yml`에서 ApplicationStart 이벤트 중에 `register_with_elb.sh` 스크립트, ApplicationStop 이벤트 중에 `deregister_from_elb.sh` 스크립트를 실행할 수 있도록 CodeDeploy에 대한 지침을 제공합니다.
6. 인스턴스가 Amazon EC2 Auto Scaling 그룹에 포함된 경우 이 단계를 건너뛸 수 있습니다.

   `common_functions.sh` 스크립트:

   - Classic Load Balancer를 사용하는 경우, `ELB_LIST=""`에 Elastic Load Balancing 로드 밸런서의 이름을 지정하고 파일에서 다른 배포 설정에 필요한 사항을 수정합니다.
   - Application Load Balancer 또는 Network Load Balancer를 사용하는 경우, `TARGET_GROUP_LIST=""`에 Elastic Load Balancing 대상 그룹의 이름을 지정하고 파일에서 다른 배포 설정에 필요한 사항을 수정합니다.
7. 애플리케이션의 소스 코드, `appspec.yml`, 배포 수명 주기 이벤트 스크립트를 하나의 애플리케이션 개정으로 번들링한 후 이 개정을 업로드합니다. 개정을 Amazon EC2 인스턴스로 배포합니다. 배포 진행 중에 배포 수명 주기 이벤트 스크립트는 로드 밸런서에서 Amazon EC2 인스턴스 등록을 취소하고 연결이 해제될 때까지 기다린 후, 배포가 완료되면 로드 밸런서에 Amazon EC2 인스턴스를 다시 등록합니다.

파트너 제품 및 서비스와의 통합

CodeDeploy는 다음 파트너 제품 및 서비스와 기본적으로 통합되어 있습니다.

Ansible	Ansible 플레이북 세트가 이미 있는데 어딘가에서 실행해야 하는 경우, Ansible 및 CodeDeploy용 템플릿은 단순한 배포 후크 2개가 로컬 배포 인스턴스에서 Ansible의 사용 및 플레이북 실행을 보장할 수 있는 방법을 보여줍니다. 인벤토리 구축 및 유지 관리를 위한 프로세스가 이미 마련되어 있으면 CodeDeploy 에이전트를 설치 및 실행하는 데 사용할 수 있는 Ansible 모듈도 있습니다. 자세히 알아보기: • Ansible 및 CodeDeploy
Atlassian – Bamboo 및 Bitbucket	Bamboo에 대한 CodeDeploy 작업은 AppSpec 파일이 포함된 디렉터리를 .zip 파일로 압축하고 파일을 Amazon S3에 업로드한 다음 CodeDeploy 애플리케이션에서 제공하는 구성에 따라 배포를 시작합니다. CodeDeploy에 대한 Atlassian Bitbucket 지원은 요청 시 Amazon EC2 인스턴스에 대한 코드를 Bitbucket UI에서 배포 그룹으로 직접 푸시할 수 있도록 합니다. 즉, Bitbucket 리포지토리에서 코드를 업데이트한 후에는 배포 프로세스를 수동으로 실행하기 위해 지속적 통합(CI) 플랫폼 또는 Amazon EC2 인스턴스에 로그인할 필요가 없습니다. 자세히 알아보기: • Bamboo용 CodeDeploy 작업 사용 • CodeDeploy에 대한 Atlassian Bitbucket 지원 발표

Chef	AWS 는 Chef와 CodeDeploy를 통합하기 위한 두 가지 템플릿 샘플을 제공합니다. 첫 번째 샘플 템플릿은 CodeDeploy 에이전트를 설치 및 시작하는 Chef 쿡북입니다. 이 템플릿은 CodeDeploy를 사용하는 동안에도 계속해서 호스트 인프라를 관리할 수 있도록 합니다. 두 번째 샘플 템플릿은 CodeDeploy를 사용하여 각 노드에서 chef-solo로 쿡북 및 레시피 실행을 오케스트레이션하는 방법을 보여줍니다. 자세히 알아보기: • Chef 및 CodeDeploy
CircleCI	CircleCI는 자동화된 테스트, 지속적 통합 및 배포 도구 세트를 제공합니다. 에서 CircleCI와 함께 AWS 사용할 IAM 역할을 생성하고 circle.yml 파일에서 배포 파라미터를 구성한 후, CircleCI를 CodeDeploy와 함께 사용하여 애플리케이션 개정을 생성하고 Amazon S3 버킷에 업로드한 다음 배포를 시작하고 모니터링할 수 있습니다. 자세히 알아보기: • CircleCI Orb를 사용하여 애플리케이션을 AWS CodeDeploy에 배포
CloudBees	CloudBees DEV@cloud에서 이용할 수 있는 CodeDeploy Jenkins 플러그인을 구축 후 작업으로 사용할 수 있습니다. 예를 들어, 지속적인 배포 파이프라인 종료 시 이 플러그인을 사용하여 서버 집합에 애플리케이션 개정을 배포할 수 있습니다. 자세히 알아보기: • DEV@cloud에서 CodeDeploy Jenkins 플러그인 제공

Codeship	Codeship을 사용하여 CodeDeploy를 통해 애플리케이션 수정 버전을 배포할 수 있습니다. Codeship UI를 사용하여 브랜치의 배포 파이프라인에 CodeDeploy를 추가할 수 있습니다. 자세히 알아보기: • CodeDeploy로 배포 • Codeship에서 CodeDeploy 통합
GitHub	CodeDeploy를 사용하여 GitHub 리포지토리에서 애플리케이션 수정 버전을 배포할 수 있습니다. 또한 GitHub 리포지토리의 소스 코드가 변경될 때마다 해당 리포지토리에서 배포를 트리거할 수도 있습니다. 자세히 알아보기: • GitHub와 CodeDeploy 통합 • 튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포 • CodeDeploy를 사용하여 GitHub에서 자동 배포
HashiCorp Consul	CodeDeploy에서 애플리케이션을 배포하는 경우 오픈 소스 HashiCorp Consul 도구를 사용하여 애플리케이션 환경의 상태 및 안정성을 보장할 수 있습니다. Consul을 사용하여 배포 중 검색할 수 있도록 애플리케이션을 등록하고, 배포에서 제외하기 위해 애플리케이션 및 노드를 유지 관리 모드로 전환하고, 대상 인스턴스가 비정상 상태가 되면 배포를 중지할 수 있습니다. 자세히 알아보기: • HashiCorp Consul을 사용하여 CodeDeploy 배포

Jenkins	CodeDeploy Jenkins 플러그 인은 Jenkins 프로젝트에 구축 후 단계를 제공합니다. 구축에 성공하면 작업 영역을 압축해 Amazon S3로 업로드하고 새 배포를 시작합니다. 자세히 알아보기: • CodeDeploy Jenkins 플러그 인 • CodeDeploy용 Jenkins 플러그 인 설정
Puppet Labs	AWS 는 Puppet 및 CodeDeploy용 샘플 템플릿을 제공합니다. 첫 번째 샘플 템플릿은 CodeDeploy 에이전트를 설치 및 시작하는 Puppet 모듈입니다. 이 템플릿은 CodeDeploy를 사용하는 동안에도 Puppet으로 계속해서 호스트 인프라를 관리할 수 있도록 합니다. 두 번째 샘플 템플릿은 CodeDeploy를 사용하여 각 노드에서 마스터 없는 puppet으로 모듈 및 매니페스트 실행을 오케스트레이션하는 방법을 보여줍니다. 자세히 알아보기: • Puppet 및 CodeDeploy
SaltStack	SaltStack 인프라를 CodeDeploy와 통합할 수 있습니다. CodeDeploy 모듈을 사용해 미니언에 CodeDeploy 에이전트를 설치 및 실행하거나 단순한 배포 후크 2개와 함께 CodeDeploy를 사용해 Salt States 실행을 오케스트레이션할 수 있습니다. 자세히 알아보기: • SaltStack 및 CodeDeploy

TeamCity	CodeDeploy Runner 플러그 인을 사용하여 TeamCity에서 직접 애플리케이션을 배포할 수 있습니다. 이 플러그 인은 애플리케이션 개정을 준비해 Amazon S3 버킷으로 업로드하는 TeamCity 구축 단계를 추가하고 해당 개정을 CodeDeploy 애플리케이션에 등록하며, CodeDeploy 배포를 만들고 선택한 경우 배포가 완료될 때까지 대기합니다. 자세히 알아보기: • CodeDeploy Runner(다운로드) • CodeDeploy Runner 플러그 인(문서)
Travis CI	성공적인 구축 후 CodeDeploy에서 배포를 트리거하도록 Travis CI를 구성할 수 있습니다. 자세히 알아보기: • Travis CI 및 CodeDeploy 배포

주제

• GitHub와 CodeDeploy 통합

GitHub와 CodeDeploy 통합

CodeDeploy는 웹 기반 호스팅 및 공유 서비스인 GitHub를 지원합니다. CodeDeploy는 GitHub 리포지토리 또는 Amazon S3 버킷에 저장된 애플리케이션 수정 버전을 인스턴스에 배포할 수 있습니다. CodeDeploy는 EC2 및 온프레미스 배포에 대해서만 GitHub를 지원합니다.

주제

• GitHub에서 CodeDeploy 수정 버전 배포
• CodeDeploy를 사용한 GitHub 동작

GitHub에서 CodeDeploy 수정 버전 배포

GitHub 리포지토리에서 인스턴스로 애플리케이션 개정을 배포하려면:

1. 배포할 Amazon EC2 인스턴스 유형 및 CodeDeploy와 호환되는 개정을 만듭니다.

   호환되는 계정을 만들려면 CodeDeploy의 개정 계획 및 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가의 지침을 따르십시오.
2. GitHub 계정을 사용하여 GitHub 리포지토리에 개정을 추가합니다.

   GitHub 계정을 만들려면 GitHub 조인을 참조하세요. GitHub 리포지토리를 만들려면 리포지토리 생성을 참조하세요.
3. CodeDeploy 콘솔의 배포 생성 페이지 또는 명령을 사용하여 GitHub AWS CLI create-deployment리포지토리에서 CodeDeploy 배포에 사용하도록 구성된 대상 인스턴스로 개정을 배포합니다.

   create-deployment 명령을 호출하려면 먼저 콘솔의 배포 생성(Create deployment) 페이지를 사용하여 지정된 애플리케이션에 대해 해당 GitHub 계정을 대신하여 GitHub와 상호 작용할 권한을 CodeDeploy에 부여해야 합니다. 애플리케이션당 한번만 이 작업을 수행하면 됩니다.

   [Create deployment] 페이지를 사용하여 GitHub 리포지토리에서 배포하는 방법에 대한 자세한 내용은 CodeDeploy에서 배포 만들기을 참조하세요.

   create-deployment 명령을 호출하여 GitHub 리포지토리에서 배포하는 방법은 EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(CLI) 단원을 참조하세요.

   CodeDeploy 배포용으로 인스턴스를 준비하는 방법은 CodeDeploy용 인스턴스 작업 섹션을 참조하세요.

자세한 내용은 튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포 단원을 참조하십시오.

CodeDeploy를 사용한 GitHub 동작

주제

- CodeDeploy에서 애플리케이션으로 GitHub 인증
- 프라이빗 및 퍼블릭 GitHub 리포지토리와 CodeDeploy 통합
- 조직에서 관리하는 GitHub 리포지토리와 CodeDeploy 통합

- CodeDeploy를 사용하여 CodePipeline에서 자동으로 배포

CodeDeploy에서 애플리케이션으로 GitHub 인증

GitHub와의 상호 작용 권한을 CodeDeploy에 부여한 후 GitHub 계정과 애플리케이션 간의 연결은 CodeDeploy에 저장됩니다. 애플리케이션을 다른 GitHub 계정에 연결할 수 있습니다. 또한 CodeDeploy가 GitHub와 상호 작용할 수 있는 권한을 취소할 수도 있습니다.

CodeDeploy에서 GitHub 계정을 애플리케이션에 연결하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포(Deploy)를 확장하고 애플리케이션(Applications)을 선택합니다.
3. 다른 GitHub 계정에 연결할 애플리케이션을 선택합니다.
4. 애플리케이션에 배포 그룹이 없는 경우, 배포 그룹 생성(Create deployment group)을 선택하여 하나를 생성합니다. 자세한 내용은 CodeDeploy에서 배포 그룹 만들기 단원을 참조하십시오. 다음 단계에서 배포 만들기를 선택하려면 배포 그룹이 필요합니다.
5. 배포에서 배포 만들기를 선택합니다.

   > **Note**
   >
   > 새 배포를 만들 필요는 없습니다. 현재로서는 이 방법이 다른 GitHub 계정을 애플리케이션에 연결하는 유일한 방법입니다.

6. 배포 설정의 개정 유형에서 애플리케이션을 GitHub에 저장을 선택합니다.
7. 다음 중 하나를 수행하세요.

   - AWS CodeDeploy 애플리케이션을 GitHub 계정에 연결하려면 별도의 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. GitHub 토큰 이름에 이 연결을 식별하는 이름을 입력한 후 GitHub에 연결을 선택합니다. 웹 페이지에 애플리케이션에 대해 GitHub와 상호 작용할 권한을 CodeDeploy에 부여하라는 메시지가 표시됩니다. 계속해서 10단계를 진행합니다.
   - 이미 만든 연결을 사용하려면 GitHub 토큰 이름에서 이름을 선택한 후 GitHub에 연결을 선택합니다. 계속해서 8단계를 진행합니다.

- 다른 GitHub 계정에 대한 연결을 만들려면 별도의 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. GitHub 토큰 이름에 연결을 식별하는 이름을 입력한 후 GitHub에 연결을 선택합니다. 웹 페이지에 애플리케이션에 대해 GitHub와 상호 작용할 권한을 CodeDeploy에 부여하라는 메시지가 표시됩니다. 계속해서 10단계를 진행합니다.

8. GitHub에 아직 로그인하기 전이라면 [Sign in] 페이지의 지침을 따라 애플리케이션을 연결하려는 GitHub 계정에 로그인합니다.
9. [Authorize application]을 선택합니다. GitHub는 선택한 애플리케이션에 대해 로그인한 GitHub 계정을 대신하여 GitHub와 상호 작용할 수 있는 권한을 CodeDeploy에 부여합니다.
10. 배포를 만들지 않으려면 [Cancel]를 선택합니다.

CodeDeploy가 GitHub와 상호 작용할 수 있는 권한을 취소하려면

1. AWS CodeDeploy 권한을 취소하려는 GitHub 계정의 자격 증명을 사용하여 GitHub에 로그인합니다.
2. GitHub 애플리케이션 페이지를 열어 권한이 있는 애플리케이션 목록에서 CodeDeploy를 찾은 다음 GitHub 절차에 따라 애플리케이션의 권한 부여를 취소합니다.

프라이빗 및 퍼블릭 GitHub 리포지토리와 CodeDeploy 통합

CodeDeploy는 프라이빗 및 퍼블릭 GitHub 리포지토리에서 애플리케이션 배포를 지원합니다. 사용자 대신 GitHub에 액세스할 수 있도록 CodeDeploy에 권한을 부여하면 CodeDeploy에는 사용자의 GitHub 계정에서 액세스할 수 있는 모든 프라이빗 GitHub 리포지토리에 대한 읽기/쓰기 액세스 권한이 생깁니다. 그러나 CodeDeploy에서는 GitHub 리포지토리 읽기만 할 수 있습니다. 사용자의 프라이빗 GitHub 리포지토리에는 쓸 수 없습니다.

조직에서 관리하는 GitHub 리포지토리와 CodeDeploy 통합

기본적으로, 사용자 계정의 프라이빗 또는 퍼블릭 리포지토리와 달리 조직에서 관리하는 GitHub 리포지토리는 CodeDeploy를 비롯하여 서드 파티 애플리케이션에 대한 액세스 권한을 부여하지 않습니다. 조직의 타사 애플리케이션 제약 조건이 GitHub에서 활성화되어 있는데 GitHub 리포지토리에서 코드를 배포하려고 하면 배포에 실패합니다. 이 문제는 두 가지 방법으로 해결할 수 있습니다.

- 조직의 멤버는 조직 소유자에게 CodeDeploy에 대한 액세스를 승인하도록 요청할 수 있습니다. 이러한 액세스 요청 단계는 개별 계정에 대한 CodeDeploy의 액세스 권한이 이미 부여되었는지 여부에 따라 달라집니다.
  - 계정에서 CodeDeploy에 대한 액세스를 승인한 경우 승인된 애플리케이션에 대한 조직 승인 요청을 참조하세요.

- 계정에서 CodeDeploy에 대한 액세스를 아직 승인하지 않은 경우 서드 파티 애플리케이션에 대한 조직 승인 요청을 참조하세요.
- 조직 소유자는 조직에 대한 모든 타사 애플리케이션 제한을 비활성화할 수 있습니다. 자세한 내용은 조직에 대한 서드 파티 애플리케이션 제한 비활성화를 참조하세요.

자세한 내용은 서드 파티 애플리케이션 제한 정보를 참조하세요.

CodeDeploy를 사용하여 CodePipeline에서 자동으로 배포

소스 코드가 변경될 때마다 CodePipeline에서 배포를 트리거할 수 있습니다. 자세한 내용은 CodePipeline을 참조하세요.

커뮤니티의 통합 예제

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

> **Note**
>
> 링크는 정보 제공 목적으로만 제공되며 포괄적인 목록이나 예제 내용의 보증으로 간주해서는 안됩니다. AWS 는 내용이나 외부 콘텐츠의 정확성에 대해서 책임을 지지 않습니다.

블로그 게시물

- 에서 CodeDeploy 프로비저닝 자동화 CloudFormation

  CloudFormation을(를) 사용하여 CodeDeploy에서 애플리케이션 배포를 프로비저닝하는 방법을 알아봅니다.

  2016년 1월 게시

- AWS Toolkit for Eclipse CodeDeploy와 통합(1부)

  AWS Toolkit for Eclipse CodeDeploy와 통합(2부)

  AWS Toolkit for Eclipse CodeDeploy와 통합(3부)

  Java 개발자가 Eclipse용 CodeDeploy 플러그인을 사용하여 Eclipse 개발 환경에서 AWS 직접에 웹 애플리케이션을 배포하는 방법을 알아봅니다.

2015년 2월 게시

- CodeDeploy를 사용하여 GitHub에서 자동 배포

  GitHub에서 CodeDeploy로의 자동 배포를 사용하여 소스 제어에서 테스트 또는 프로덕션 환경으로 엔드 투 엔드 파이프라인을 만드는 방법을 알아봅니다.

  2014년 12월 게시

CodeDeploy 자습서

이 단원에는 CodeDeploy 사용 방법에 대해 알아보는 데 유용하게 활용할 수 있는 몇 가지 자습서가 포함되어 있습니다.

이러한 튜토리얼의 절차에서는 파일을 저장할 위치(예: c:\temp) 및 버킷, 하위 폴더 또는 파일에 지정할 이름(예: 각각 amzn-s3-demo-bucket, HelloWorldApp 및 CodeDeployDemo-EC2-Trust.json)을 제안하지만 이러한 제안 사항을 사용할 필요는 없습니다. 절차를 수행하면서 파일 위치 및 이름만 대체하면 됩니다.

주제

- 튜토리얼: Amazon EC2 인스턴스(Amazon Linux 또는 Red Hat Enterprise Linux 및 Linux, macOS 또는 Unix)에 WordPress 배포
- 튜토리얼: CodeDeploy를 사용하여 "hello, world!" 애플리케이션 배포(Windows Server)
- 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)
- 튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포
- 튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포
- 튜토리얼: Amazon ECS에 애플리케이션 배포
- 튜토리얼: Amazon ECS 서비스 배포 및 확인 테스트
- 자습서: CodeDeploy 및 AWS 서버리스 애플리케이션 모델을 사용하여 업데이트된 Lambda 함수 배포

튜토리얼: Amazon EC2 인스턴스(Amazon Linux 또는 Red Hat Enterprise Linux 및 Linux, macOS 또는 Unix)에 WordPress 배포

이 튜토리얼에서는 PHP와 MySQL을 기반으로 하는 오픈 소스 블로그 도구 및 콘텐츠 관리 시스템인 WordPress를 Amazon Linux 또는 Red Hat Enterprise Linux(RHEL)를 실행하는 단일 Amazon EC2 인스턴스에 배포합니다.

찾는 항목이 보이지 않습니까?

- 대신 Windows Server를 실행하는 Amazon EC2 인스턴스에 배포하는 방법을 연습하려면 튜토리얼: CodeDeploy를 사용하여 "hello, world!" 애플리케이션 배포(Windows Server) 단원을 참조하세요.

- Amazon EC2 인스턴스 대신 온프레미스 인스턴스에 배포하는 방법을 연습하려면 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux) 단원을 참조하세요.

이 튜토리얼의 단계는 Linux, macOS, 또는 Unix를 실행하는 로컬 개발 시스템의 관점에서 제시됩니다. Windows를 실행하는 로컬 컴퓨터에서 이러한 단계를 대부분 완료할 수 있지만 명령(예: chmod, wget), 애플리케이션(예: sed), 디렉터리 경로(예: `/tmp`)를 다루는 단계를 적용해야 합니다.

이 자습서를 시작하기 전에 CodeDeploy 시작하기의 사전 조건을 완료해야 합니다. 여기에는 사용자 구성, 설치 또는 업그레이드 AWS CLI, IAM 인스턴스 프로파일 및 서비스 역할 생성이 포함됩니다.

주제

- 1단계: Amazon Linux 또는 Red Hat Enterprise Linux Amazon EC2 인스턴스 시작 및 구성
- 2단계: Amazon Linux 또는 Red Hat Enterprise Linux Amazon EC2 인스턴스에 배포할 원본 콘텐츠 구성
- 3단계: WordPress 애플리케이션을 Amazon S3에 업로드
- 4단계: WordPress 애플리케이션 배포
- 5단계: WordPress 애플리케이션 업데이트 및 재배포
- 6단계: WordPress 애플리케이션 및 관련 리소스 정리

1단계: Amazon Linux 또는 Red Hat Enterprise Linux Amazon EC2 인스턴스 시작 및 구성

CodeDeploy를 사용하여 WordPress 애플리케이션을 배포하려면 Amazon Linux 또는 Red Hat Enterprise Linux(RHEL)를 실행하는 Amazon EC2 인스턴스가 필요합니다. Amazon EC2 인스턴스에는 HTTP 연결을 허용하는 새로운 인바운드 보안 규칙이 필요합니다. 이 규칙은 애플리케이션이 성공적으로 배포된 후 브라우저에서 WordPress 페이지를 보기 위해 필요합니다.

CodeDeploy를 위한 Amazon EC2 인스턴스 생성의 지침을 따르세요. 인스턴스에 Amazon EC2 인스턴스 태그 할당 지침을 수행한 경우 태그 키로 **`Name`**, 태그 값으로 **`CodeDeployDemo`**가 지정되어 있어야 합니다. 태그 키 또는 태그 값을 다르게 지정한 경우 4단계: WordPress 애플리케이션 배포의 지침을 따르면 예기치 않은 결과가 발생할 수 있습니다.

지침에 따라 Amazon EC2 인스턴스를 시작한 후에는 이 페이지로 돌아와 계속해서 다음 단원으로 진행합니다. 다음 단계로 CodeDeploy를 사용하여 애플리케이션 생성 단원을 진행하지 마세요.

Amazon Linux 또는 RHEL Amazon EC2 인스턴스에 연결

새 Amazon EC2 인스턴스 시작 후 다음 지침에 따라 해당 인스턴스에 연결하는 연습을 합니다.

1. ssh 명령(또는 PuTTY와 같이 SSH를 지원하는 터미널 에뮬레이터)을 사용하여 Amazon Linux 또는 RHEL Amazon EC2 인스턴스에 연결하세요. Amazon EC2 인스턴스를 시작할 때 사용한 키 페어의 프라이빗 키와 인스턴스의 퍼블릭 DNS 주소가 필요합니다. 자세한 내용은 인스턴스에 연결을 참조하세요.

   예를 들어 퍼블릭 DNS 주소가 **ec2-01-234-567-890.compute-1.amazonaws.com**이고, SSH 액세스를 위한 Amazon EC2 인스턴스 키 페어 이름이 **codedeploydemo.pem**인 경우, 다음을 입력하세요.

   ```
   ssh -i /path/to/codedeploydemo.pem ec2-
   user@ec2-01-234-567-890.compute-1.amazonaws.com
   ```

   */path/to/*codedeploydemo.pem을(를) .pem 파일의 경로로 바꾸고, 예제 DNS 주소를 Amazon Linux 또는 RHEL Amazon EC2 인스턴스에 대한 주소로 바꾸세요.

   > **Note**
   >
   > 키 파일의 권한이 너무 개방되었다는 오류가 표시되면, 현재 사용자에게만 액세스를 부여하도록 권한을 제한해야 합니다. 예를 들어, Linux, macOS 또는 Unix에서 chmod 명령을 사용하는 경우 다음을 입력합니다.

   ```
   chmod 400 /path/to/codedeploydemo.pem
   ```

2. 로그인하면 Amazon EC2 인스턴스에 대한 AMI 배너가 표시됩니다. Amazon Linux의 경우 다음과 같아야 합니다.

   ```
        __|  __|_  )
        _|  (     /   Amazon Linux AMI
       ___|\___|___|
   ```

3. 이제 실행 중인 Amazon EC2 인스턴스에서 로그아웃할 수 있습니다.

> **Warning**
>
> Amazon EC2 인스턴스를 중지하거나 종료하지 마세요. 인스턴스를 중지하거나 종료할 경우 CodeDeploy가 인스턴스에 배포할 수 없습니다.

Amazon Linux 또는 RHEL Amazon EC2 인스턴스에 HTTP 트래픽을 허용하는 인바운드 규칙 추가

다음 단계에서는 배포된 WordPress 애플리케이션의 홈 페이지를 브라우저에서 볼 수 있도록 Amazon EC2 인스턴스에 열린 HTTP 포트가 있는지 확인합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 인스턴스를 선택한 후 인스턴스를 선택합니다.
3. 설명 탭의 보안 그룹에서 인바운드 규칙 보기를 선택합니다.

   보안 그룹에 다음과 같은 규칙 목록이 있어야 합니다.

   ```
   Security Groups associated with i-1234567890abcdef0
    Ports     Protocol     Source     launch-wizard-N
    22        tcp          0.0.0.0/0          #
   ```

4. 보안 그룹에서 Amazon EC2 인스턴스를 위한 보안 그룹을 선택합니다. 이름은 **launch-wizard-*N***이 될 수 있습니다. 이름의 ***N***은 인스턴스가 생성될 때 보안 그룹에 할당된 번호입니다.

   인바운드 탭을 선택합니다. 인스턴스의 보안 그룹이 올바르게 구성되어 있으면 다음 값이 있는 규칙이 표시되어야 합니다.

   - 유형: HTTP
   - 프로토콜: TCP
   - 포트 범위: 80
   - 소스: 0.0.0.0/0

5. 이러한 값이 있는 규칙이 없는 경우 보안 그룹에 규칙 추가의 절차를 사용하여 해당 값을 새 보안 규칙에 추가하세요.

2단계: Amazon Linux 또는 Red Hat Enterprise Linux Amazon EC2 인스턴스에 배포할 원본 콘텐츠 구성

이제 인스턴스에 배포할 항목이 있도록 애플리케이션의 원본 콘텐츠를 구성해야 합니다.

주제

- 소스 코드 가져오기
- 애플리케이션을 실행하기 위한 스크립트 만들기
- 애플리케이션 사양 파일 추가

소스 코드 가져오기

이 튜토리얼에서는 개발 시스템에서 대상 Amazon EC2 인스턴스로 WordPress 콘텐츠 게시 플랫폼을 배포합니다. WordPress 소스 코드를 얻으려면 기본 제공 명령줄 호출을 사용할 수 있습니다. 또는 Git이 개발 컴퓨터에 설치되어있는 경우 대신 사용할 수 있습니다.

이 단계를 위해, WordPress 소스 코드의 복사본을 개발 머신의 `/tmp` 디렉터리로 다운로드할 수 있다고 가정합니다. (원하는 디렉터리를 선택할 수 있지만 이 단계에서 지정되는 모든 곳에서 위치를 `/tmp`로 바꿔야 합니다.)

WordPress 소스 파일을 개발 컴퓨터에 복사하려면 다음 두 가지 옵션 중 하나를 선택합니다. 첫 번째 옵션은 기본 제공 명령줄 호출을 사용합니다. 두 번째 옵션은 Git를 사용합니다.

주제

- WordPress 소스 코드의 복사본을 얻으려면(기본 제공 명령줄 호출)
- WordPress 소스 코드의 복사본을 얻으려면(Git)

WordPress 소스 코드의 복사본을 얻으려면(기본 제공 명령줄 호출)

1. wget 명령을 호출하여 WordPress 소스 코드의 복사본을.zip 파일로 현재 디렉터리에 다운로드합니다.

   ```
   wget https://github.com/WordPress/WordPress/archive/master.zip
   ```

2. unzip, mkdir, cp, 및 rm 명령으르 호출하여 다음을 수행합니다.

   - `master.zip` 파일을 `/tmp/WordPress_Temp` 디렉터리(폴더)에 압축을 풉니다.

- 압축이 풀린 내용을 /tmp/WordPress 대상 폴더에 복사합니다.
- 임시 /tmp/WordPress_Temp 폴더 및 master 파일을 삭제합니다.

명령을 한 번에 하나씩 실행합니다.

```
unzip master -d /tmp/WordPress_Temp
```

```
mkdir -p /tmp/WordPress
```

```
cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
```

```
rm -rf /tmp/WordPress_Temp
```

```
rm -f master
```

이렇게 하면 /tmp/WordPress 폴더에 깨끗한 WordPress 소스 코드 파일 세트가 남게 됩니다.

WordPress 소스 코드의 복사본을 얻으려면(Git)

1. 개발 머신에 Git를 다운로드하고 설치합니다.
2. /tmp/WordPress 폴더에서 git init 명령을 호출합니다.
3. git clone 명령을 호출하여 공용 WordPress 리포지토리를 복제하고 /tmp/WordPress 대상 폴더에 자신만의 복사본을 만듭니다.

```
git clone https://github.com/WordPress/WordPress.git /tmp/WordPress
```

이렇게 하면 /tmp/WordPress 폴더에 깨끗한 WordPress 소스 코드 파일 세트가 남게 됩니다.

애플리케이션을 실행하기 위한 스크립트 만들기

그런 다음 디렉터리에 폴더와 스크립트를 만듭니다. CodeDeploy는 이러한 스크립트를 사용하여 대상 Amazon EC2 인스턴스에 애플리케이션 수정 버전을 설정하고 배포합니다. 어떤 텍스트 편집기든 사용하여 스크립트를 생성할 수 있습니다.

1. WordPress 소스 코드의 복사본에 스크립트 디렉터리를 만듭니다.

```
mkdir -p /tmp/WordPress/scripts
```

2. /tmp/WordPress/scripts에 install_dependencies.sh 파일을 생성합니다. 파일에 다음 줄을 추가합니다. 이 install_dependencies.sh 스크립트는 Apache, MySQL, PHP를 설치합니다. 또한 PHP에 MySQL 지원을 추가합니다.

```
#!/bin/bash
sudo amazon-linux-extras install php7.4
sudo yum install -y httpd mariadb-server php
```

3. /tmp/WordPress/scripts에 start_server.sh 파일을 생성합니다. 파일에 다음 줄을 추가합니다. 이 start_server.sh 스크립트는 Apache와 MySQL을 시작합니다.

```
#!/bin/bash
systemctl start mariadb.service
systemctl start httpd.service
systemctl start php-fpm.service
```

4. /tmp/WordPress/scripts에 stop_server.sh 파일을 생성합니다. 파일에 다음 줄을 추가합니다. 이 stop_server.sh 스크립트는 Apache와 MySQL을 중지합니다.

```
#!/bin/bash
isExistApp="pgrep httpd"
if [[ -n $isExistApp ]]; then
systemctl stop httpd.service
fi
isExistApp=pgrep mysqld
if [[ -n $isExistApp ]]; then
systemctl stop mariadb.service
fi
isExistApp=pgrep php-fpm
if [[ -n $isExistApp ]]; then
systemctl stop php-fpm.service

fi
```

5. /tmp/WordPress/scripts에 create_test_db.sh 파일을 생성합니다. 파일에 다음 줄을 추가합니다. 이 create_test_db.sh스크립트는 MySQL을 사용하여 WordPress가 사용할 **test** 데이터베이스를 만듭니다.

```
#!/bin/bash
mysql -uroot <<CREATE_TEST_DB
CREATE DATABASE IF NOT EXISTS test;
CREATE_TEST_DB
```

6. 마지막으로, /tmp/WordPress/scripts에서 change_permissions.sh 스크립트를 만듭니다. 이것은 Apache에서 폴더 권한을 변경하는 데 사용됩니다.

> **Important**
>
> 이 스크립트는 /tmp/WordPress 폴더에서 사용 권한을 업데이트하여 누구나 쓸 수 있습니다. 이것은 5단계: WordPress 애플리케이션 업데이트 및 재배포 중에 WordPress가 데이터베이스에 쓸 수 있도록 하는 데 필요합니다 WordPress 애플리케이션을 설정한 후 다음 명령을 실행하여 사용 권한을 보다 안전한 설정으로 업데이트합니다.
>
> ```
> chmod -R 755 /var/www/html/WordPress
> ```

```
#!/bin/bash
chmod -R 777 /var/www/html/WordPress
```

7. 모든 스크립트에 실행 권한을 부여합니다. 명령줄에 다음을 입력합니다.

```
chmod +x /tmp/WordPress/scripts/*
```

애플리케이션 사양 파일 추가

다음으로, CodeDeploy 에서 사용하는 YAML 형식 파일인 애플리케이션 사양 파일(AppSpec 파일)을 추가하여 다음을 수행합니다.

- 애플리케이션 수정 버전의 소스 파일을 Amazon EC2 인스턴스의 대상으로 매핑합니다.

- 배포된 파일에 대한 사용자 정의 권한을 지정합니다.
- 배포 중에 대상 Amazon EC2 인스턴스에서 실행할 스크립트를 지정합니다.

AppSpec 파일의 이름은 `appspec.yml`이어야 합니다. 애플리케이션 소스 코드의 루트 폴더에 있어야 합니다. 이 튜토리얼에서 루트 디렉토리는 `/tmp/WordPress`입니다.

텍스트 편집기에서 `appspec.yml`(이)라는 파일을 만듭니다. 파일에 다음 줄을 추가합니다.

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

CodeDeploy는 이 AppSpec 파일을 사용하여 개발 시스템에 있는 `/tmp/WordPress` 폴더의 모든 파일을 대상 Amazon EC2 인스턴스의 `/var/www/html/WordPress` 폴더 복사합니다. 배포하는 동안 CodeDeploy는 지정된 스크립트를 배포 수명 주기 동안 지정된 이벤트(예:**`BeforeInstall`** 및 **`AfterInstall`**)에서 대상 Amazon EC2 인스턴스에 있는 `/var/www/html/WordPress/scripts` 폴더에서 `root`(으)로 실행합니다. 스크립트를 실행하는 데 300초(5분) 이상 걸리면 CodeDeploy는 배포를 중지하고 배포를 실패로 표시합니다.

이러한 설정에 대한 자세한 정보는 CodeDeploy AppSpec 파일 참조 단원을 참조하세요.

> **Important**
>
> 이 파일에 있는 각 항목 사이의 공백 위치와 개수는 중요합니다. 공백이 올바르지 않으면 CodeDeploy는 디버깅하기 어려운 오류를 발생시킵니다. 자세한 내용은 AppSpec 파일 간격 단원을 참조하십시오.

3단계: WordPress 애플리케이션을 Amazon S3에 업로드

이제 CodeDeploy가 배포할 수 있는 위치에 소스 콘텐츠를 준비하고 업로드합니다. 다음 지침은 Amazon S3 버킷을 프로비저닝하고, 버킷에 대한 애플리케이션 수정 버전 파일을 준비하고, 수정 버전의 파일을 번들로 묶은 다음, 버전을 버킷에 푸시하는 방법을 보여 줍니다.

> **Note**
>
> 이 튜토리얼에서 다루지 않지만 CodeDeploy를 사용하여 애플리케이션을 GitHub 리포지토리에서 인스턴스로 배포할 수 있습니다. 자세한 내용은 GitHub와 CodeDeploy 통합 단원을 참조하세요.

주제

- Amazon S3 버킷 프로비저닝
- 버킷에 대한 애플리케이션 파일 준비
- 애플리케이션의 파일을 단일 아카이브 파일로 묶고 아카이브 파일을 푸시합니다.

Amazon S3 버킷 프로비저닝

스토리지 컨테이너 또는 Amazon S3의 버킷을 만듭니다(또는 기존 버킷 사용). 버킷에 수정 버전을 업로드할 수 있는지, 배포에 사용된 Amazon EC2 인스턴스가 버킷에서 수정 버전을 다운로드할 수 있는지 확인합니다.

AWS CLI, Amazon S3 콘솔 또는 Amazon S3 APIs를 사용하여 Amazon S3 버킷을 생성할 수 있습니다. 버킷을 생성한 후에는 버킷과 AWS 계정에 액세스 권한을 부여해야 합니다.

> **Note**
>
> 버킷 이름은 모든 AWS 계정의 Amazon S3에서 고유해야 합니다. **amzn-s3-demo-bucket**을(를) 사용할 수 없는 경우, 다른 버킷 이름(예: **amzn-s3-demo-bucket** 뒤에 대시와 이니셜 또

는 다른 고유 식별자가 옴)을 사용하세요. 그런 다음 이 자습서 전체에서 버킷 이름을 **amzn-s3-demo-bucket**(으)로 대체해야 합니다.
대상 Amazon EC2 인스턴스가 시작되는 리전과 동일한 리전에서 Amazon S3 버킷을 생성해야 합니다. AWS Amazon EC2 예를 들어 버킷을 미국 동부(버지니아) 리전에 생성한 경우 대상 Amazon EC2 인스턴스를 미국 동부(버지니아 북부) 리전에서 시작해야 합니다.

주제

- Amazon S3 버킷을 생성하려면(CLI)
- Amazon S3 버킷을 생성하려면(콘솔)
- Amazon S3 버킷 및 AWS 계정에 권한 부여

Amazon S3 버킷을 생성하려면(CLI)

mb 명령을 호출하여 이름이 **amzn-s3-demo-bucket**인 Amazon S3 버킷을 생성합니다.

```
aws s3 mb s3://amzn-s3-demo-bucket --region region
```

Amazon S3 버킷을 생성하려면(콘솔)

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. Amazon S3; 콘솔에서 버킷 생성을 선택합니다.
3. 버킷 이름 상자에서 버킷의 이름을 입력합니다.
4. 리전 목록에서 대상 리전을 선택한 다음 생성을 선택합니다.

Amazon S3 버킷 및 AWS 계정에 권한 부여

또한 Amazon S3 버킷으로 업로드할 수 있는 권한이 있어야 합니다. Amazon S3 버킷 정책을 통해 이러한 권한을 지정할 수 있습니다. 예를 들어 다음 Amazon S3 버킷 정책에서 와일드카드 문자(*)를 사용하면 AWS 계정이 라는 Amazon S3 버킷의 모든 디렉터리에 파일을 111122223333 업로드할 수 있습니다amzn-s3-demo-bucket.

```
{
    "Statement": [
        {
            "Action": [
```

```
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

AWS 계정 ID를 보려면 AWS 계정 ID 찾기를 참조하세요.

이제 Amazon S3 버킷이 참여하는 각 Amazon EC2 인스턴스에서 다운로드 요청을 허용하는지 확인하는 것이 좋습니다. Amazon S3 버킷 정책을 통해 이를 지정할 수 있습니다. 예를 들어, 다음 Amazon S3 버킷 정책에서 와일드카드 문자(*)를 사용하면 `arn:aws:iam::444455556666:role/CodeDeployDemo` ARN이 포함된 IAM 인스턴스 프로파일이 연결된 Amazon EC2 인스턴스가 이름이 `amzn-s3-demo-bucket`인 Amazon S3 버킷의 디렉터리에서 파일을 다운로드할 수 있습니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

Amazon S3 버킷 정책 생성 및 연결에 대한 자세한 내용은 버킷 정책 예제를 참조하세요.

IAM 정책을 만들고 연결하는 방법은 정책 작업을 참조하세요.

버킷에 대한 애플리케이션 파일 준비

WordPress 애플리케이션 파일, AppSpec 파일 및 스크립트가 개발 컴퓨터에서 다음과 같이 구성되어 있는지 확인하세요.

```
/tmp/
  |--WordPress/
       |-- appspec.yml
       |-- scripts/
       |    |-- change_permissions.sh
       |    |-- create_test_db.sh
       |    |-- install_dependencies.sh
       |    |-- start_server.sh
       |    |-- stop_server.sh
       |-- wp-admin/
       |    |-- (various files...)
       |-- wp-content/
       |    |-- (various files...)
       |-- wp-includes/
       |    |-- (various files...)
       |-- index.php
       |-- license.txt
       |-- readme.html
       |-- (various files ending with .php...)
```

애플리케이션의 파일을 단일 아카이브 파일로 묶고 아카이브 파일을 푸시합니다.

WordPress 애플리케이션 파일과 AppSpec 파일을 아카이브 파일(애플리케이션 수정 버전이라고도 함)로 번들링합니다.

> **Note**
>
> 객체를 버킷에 저장하거나 애플리케이션 수정 버전을 버킷으로 전송한 경우 요금이 부과될 수 있습니다. 자세한 내용은 Amazon S3 요금을 참조하세요.

1. 개발 컴퓨터에서 파일이 저장된 폴더로 전환하세요.

   ```
   cd /tmp/WordPress
   ```

> **Note**
>
> 이 폴더로 전환하지 않으면 현재 폴더에서 파일 번들링이 시작됩니다. 예를 들어 현재 폴더가 `/tmp/WordPress` 대신 `/tmp`이면 `tmp` 폴더의 파일 및 하위 폴더로 번들링이 시작되어 `WordPress` 하위 폴더보다 더 많은 폴더를 포함할 수 있습니다.

2. create-application 명령을 호출하여 이름이 **`WordPress_App`**인 새 애플리케이션을 등록합니다.

```
aws deploy create-application --application-name WordPress_App
```

3. CodeDeploy push 명령을 호출하여 파일을 묶고, 수정 버전을 Amazon S3로 업로드하고, 업로드된 수정 버전에 대한 정보를 CodeDeploy에 등록할 수 있습니다. 이 모든 작업을 한 번에 수행할 수 있습니다.

```
aws deploy push \
  --application-name WordPress_App \
  --s3-location s3://amzn-s3-demo-bucket/WordPressApp.zip \
  --ignore-hidden-files
```

이 명령은 현재 디렉터리의 파일(숨겨진 파일 제외)을 단일 아카이브 파일(이름: **`WordPressApp.zip`**)로 번들링하고 수정 버전을 **`amzn-s3-demo-bucket`** 버킷에 업로드하고 업로드된 수정 버전에 대한 정보를 CodeDeploy에 등록합니다.

4단계: WordPress 애플리케이션 배포

이제 Amazon S3에 업로드한 샘플 WordPress 애플리케이션 수정 버전을 배포합니다. AWS CLI 또는 CodeDeploy 콘솔을 사용하여 개정을 배포하고 배포 진행 상황을 모니터링할 수 있습니다. 애플리케이션 수정 배포에 성공한 후에는 결과를 확인합니다.

주제

- CodeDeploy로 애플리케이션 수정 버전 배포
- 배포 모니터링 및 문제 해결
- 배포 확인

CodeDeploy로 애플리케이션 수정 버전 배포

AWS CLI 또는 콘솔을 사용하여 애플리케이션 개정을 배포합니다.

주제

- 애플리케이션 수정을 배포하려면(CLI)
- 애플리케이션 수정을 배포하려면(콘솔)

애플리케이션 수정을 배포하려면(CLI)

1. 배포에는 배포 그룹이 필요합니다. 그러나 배포 그룹을 만들기 전에 서비스 역할 ARN이 필요합니다. 서비스 역할은 서비스에 사용자를 대신할 수 있는 권한을 부여하는 IAM 역할입니다. 이 경우에 서비스 역할은 CodeDeploy에 Amazon EC2 인스턴스에 액세스하여 Amazon EC2 인스턴스 태그를 확장할(읽을) 수 있는 권한을 부여합니다.

   서비스 역할 생성(CLI)의 지침에 따라 앞에서 서비스 역할을 생성했습니다. 서비스 역할의 ARN을 확인하려면 서비스 역할 ARN 확인(CLI) 단원을 참조하세요.

2. 이제 ARN이 있으므로 create-deployment-group 명령을 호출하고, **CodeDeployDemo**(이)라는 Amazon EC2 인스턴스와 **CodeDeployDefault.OneAtATime**(이)라는 배포 구성을 사용하여 **WordPress_App**(이)라는 애플리케이션과 연결된 **WordPress_DepGroup**(이)라는 배포 그룹을 만듭니다.

```
aws deploy create-deployment-group \
  --application-name WordPress_App \
  --deployment-group-name WordPress_DepGroup \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --ec2-tag-filters Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE \
  --service-role-arn serviceRoleARN
```

> **Note**
>
> create-deployment-group 명령은 배포 및 인스턴스의 지정된 이벤트에 대해 주제의 구독자에게 Amazon SNS 알림을 전송하게 하는 트리거를 생성합니다. 또한 해당 명령은 Amazon CloudWatch 경보 모니터링 임계값에 도달한 경우 자동으로 배포를 롤백하고 배포를 중지하도록 경보를 설정하는 옵션을 지원합니다. 이 작업에 대한 명령은 이 자습서에 포함되지 않습니다.

3. 배포를 만들기 전에 배포 그룹의 인스턴스에 CodeDeploy 에이전트가 설치되어 있어야 합니다. AWS Systems Manager 를 사용하여 명령줄에서 다음 명령으로 에이전트를 설치할 수 있습니다.

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \
  --parameters action=Install,name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

   이 명령은 Systems Manager 상태 관리자에 연결을 생성하여 CodeDeploy 에이전트를 설치하고 매주 일요일 오전 2시에 업데이트를 시도합니다. CodeDeploy 에이전트에 대한 자세한 내용은 CodeDeploy 에이전트 작업을 참조하세요. Systems Manager에 대한 자세한 내용은 AWS Systems Manager란 무엇입니까를 참조하세요.

4. 이제 create-deployment 명령을 호출하고 **WordPress_App**이라는 버킷에 있는 **CodeDeployDefault.OneAtATime**이라는 애플리케이션 수정을 사용하여 **WordPress_DepGroup**이라는 애플리케이션, **WordPressApp.zip**이라는 배포 구성 및 **amzn-s3-demo-bucket**이라는 배포 그룹과 연결된 배포를 만듭니다.

```
aws deploy create-deployment \
  --application-name WordPress_App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name WordPress_DepGroup \
  --s3-location bucket=amzn-s3-demo-bucket,bundleType=zip,key=WordPressApp.zip
```

애플리케이션 수정을 배포하려면(콘솔)

1. CodeDeploy 콘솔을 사용하여 애플리케이션 수정을 배포하기 전에 서비스 역할 ARN이 필요합니다. 서비스 역할은 서비스에 사용자를 대신할 수 있는 권한을 부여하는 IAM 역할입니다. 이 경우에 서비스 역할은 CodeDeploy에 Amazon EC2 인스턴스에 액세스하여 Amazon EC2 인스턴스 태그를 확장할(읽을) 수 있는 권한을 부여합니다.

   서비스 역할 생성(콘솔)의 지침에 따라 앞에서 서비스 역할을 생성했습니다. 서비스 역할의 ARN을 확인하려면 서비스 역할 ARN 확인(콘솔) 단원을 참조하세요.

2. 이제 ARN이 있으므로 CodeDeploy 콘솔을 사용하여 애플리케이션 수정 버전을 배포할 수 있습니다.

   에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

3. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
4. 애플리케이션 목록에서 WordPress_App을 선택합니다.
5. 배포 그룹 탭에서 Create deployment group(배포 그룹 생성)을 선택합니다.
6. Deployment group name(배포 그룹 이름)에 **WordPress_DepGroup**을 입력합니다.
7. 배포 유형 아래에서 인 플레이스(In-place) 배포를 선택합니다.
8. 환경 구성에서 Amazon EC2 인스턴스를 선택합니다.
9. 를 사용한 에이전트 구성 AWS Systems Manager에서 기본값을 유지합니다.
10. 키에 **Name**을(를) 입력합니다.
11. 값에 **CodeDeployDemo**을(를) 입력합니다.

> **Note**
>
> **CodeDeployDemo**을(를) 입력한 후 CodeDeploy가 일치하는 Amazon EC2 인스턴스를 하나 찾았는지 확인할 수 있도록 1이 일치하는 인스턴스 아래에 표시되어야 합니다.

12. 배포 구성에서 CodeDeployDefault.OneAtATime을 선택합니다.
13. 서비스 역할 ARN에서 서비스 역할 ARN을 선택한 후배포 그룹 생성을 선택합니다.
14. 배포 만들기를 선택합니다.
15. 배포 그룹에서 **WordPress_DepGroup**을(를) 선택합니다.
16. 리포지토리 유형 옆의 내 애플리케이션이 Amazon S3에 저장되어 있음을 선택합니다. 수정 버전 위치에 이전에 Amazon S3에 업로드한 샘플 WordPress 애플리케이션 수정 버전의 위치를 입력합니다. 위치를 가져오려면:

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

    b. 버킷 목록에서 amzn-s3-demo-bucket(또는 애플리케이션 수정 버전을 업로드한 버킷 이름)을 선택합니다.

    c. 객체 목록에서 WordPressApp.zip을 선택합니다.

    d. 개요 탭에서 링크 필드의 값을 클립보드에 복사합니다.

    값이 다음과 같을 것입니다.

https://s3.amazonaws.com/amzn-s3-demo-bucket/WordPressApp.zip

e. CodeDeploy 콘솔로 돌아가서 수정 버전 위치에 링크 필드 값을 붙여 넣습니다.

17. 파일 형식에서 파일 형식을 찾을 수 없다는 메시지가 표시되는 경우, .Zip을 선택합니다.
18. (선택 사항) 배포 설명 상자에 설명을 입력합니다.
19. 배포 그룹 재정의를 확장하고 배포 구성에서 CodeDeployDefault.OneAtATime을 선택합니다.
20. Start deployment(배포 시작)를 선택합니다. 새로 만든 배포에 대한 정보가 [Deployments] 페이지에 표시됩니다.

배포 모니터링 및 문제 해결

AWS CLI 또는 콘솔을 사용하여 배포를 모니터링하고 문제를 해결합니다.

주제

- 배포를 모니터링하고 문제를 해결하려면(CLI)
- 배포를 모니터링하고 문제를 해결하려면(콘솔)

배포를 모니터링하고 문제를 해결하려면(CLI)

1. **WordPress_App**이라는 애플리케이션 및 **WordPress_DepGroup**이라는 배포 그룹에 대해 list-deployments 명령을 호출하여 배포 ID를 가져옵니다.

```
aws deploy list-deployments --application-name WordPress_App --deployment-group-
name WordPress_DepGroup --query 'deployments' --output text
```

2. 배포 ID로 get-deployment 명령을 호출합니다.

```
aws deploy get-deployment --deployment-id deploymentID --query
 'deploymentInfo.status' --output text
```

3. 이 명령으로 배포의 전체 상태가 반환됩니다. 성공하면 `Succeeded` 값이 반환됩니다.

   전체 상태가 `Failed`이면 list-deployment-instances 및 get-deployment-instance와 같은 명령을 호출하여 문제를 해결할 수 있습니다. 문제 해결 옵션을 더 보려면 인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석을 참조하세요.

배포를 모니터링하고 문제를 해결하려면(콘솔)

CodeDeploy 콘솔의 배포 페이지에 있는 상태 열에서 배포 상태를 모니터링할 수 있습니다.

배포에 대한 자세한 정보를 확인하려면(특히, [Status] 열 값이 [Succeeded]가 아닌 경우):

1. 배포 테이블에서 배포 이름을 선택합니다. 배포에 실패하면 실패 원인을 설명하는 메시지가 표시됩니다.
2. 인스턴스 활동에서 배포에 대한 자세한 정보가 표시됩니다. 배포에 실패한 후 배포에 실패한 Amazon EC2 인스턴스와 해당 단계를 확인할 수 있습니다.
3. 문제 해결을 더 수행해야 할 경우 View Instance Details에 설명된 것과 같은 기능을 사용할 수 있습니다. 또한 Amazon EC2 인스턴스의 배포 로그 파일을 분석할 수도 있습니다. 자세한 내용은 인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석 단원을 참조하세요.

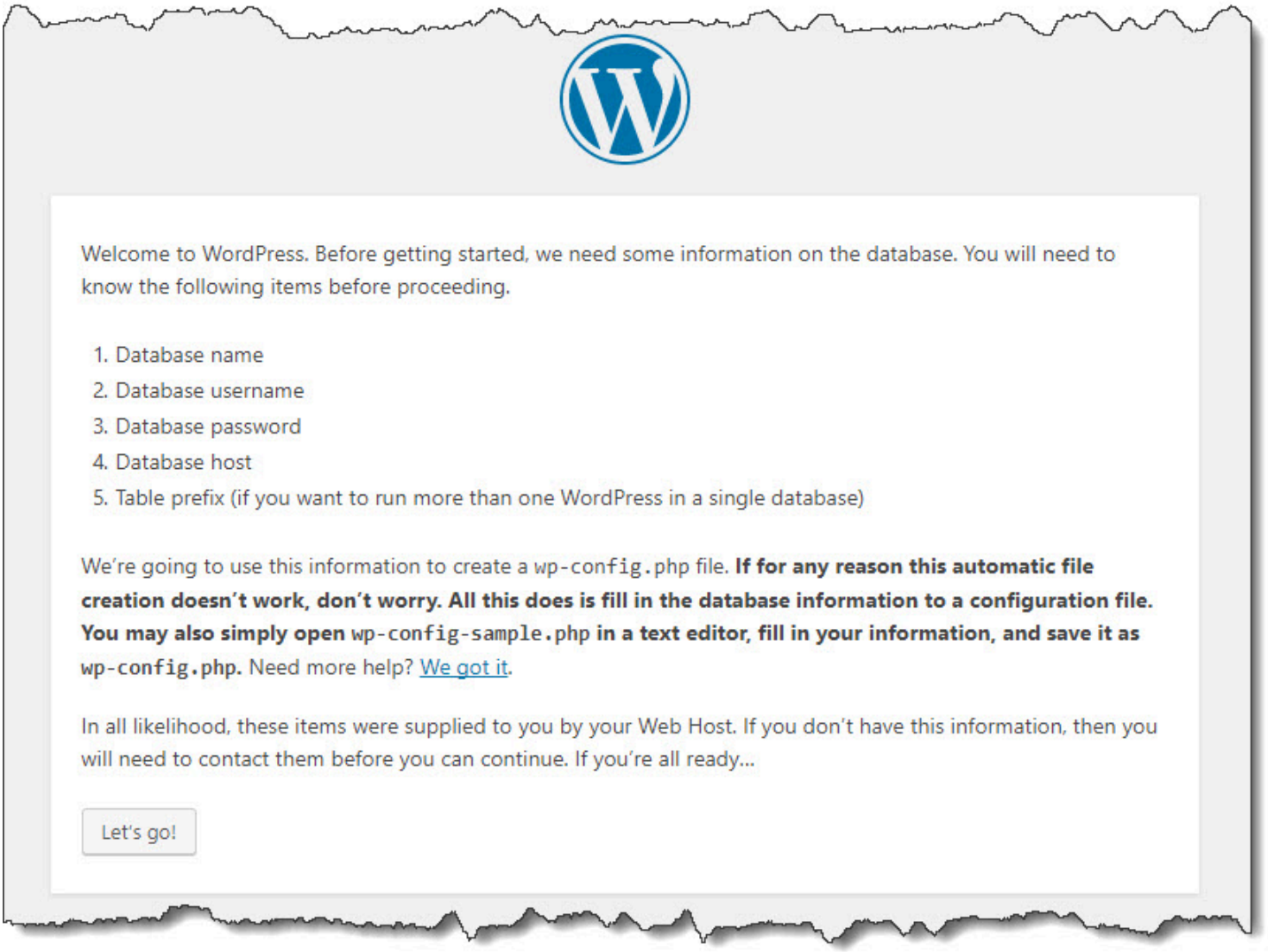
배포 확인

배포에 성공한 후 WordPress 설치가 작동 중인지 확인합니다. Amazon EC2 인스턴스의 퍼블릭 DNS 주소(뒤에 `/WordPress`이 옴)를 사용해 웹 브라우저에서 사이트를 확인합니다. (퍼블릭 DNS 값을 확인하려면 Amazon EC2 콘솔에서 Amazon EC2 인스턴스를 선택하고 설명 탭에서 퍼블릭 DNS의 값을 찾습니다.)

예를 들어, Amazon EC2 인스턴스의 퍼블릭 DNS 주소가 **`ec2-01-234-567-890.compute-1.amazonaws.com`**이면 다음 URL을 사용합니다.

```
http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress
```

브라우저에서 사이트를 볼 때 다음과 유사한 WordPress 시작 페이지가 표시됩니다.



Amazon EC2 인스턴스에 HTTP 인바운드 규칙이 보안 그룹에 추가되지 않은 경우 WordPress 시작 페이지가 나타나지 않습니다. 원격 서버가 응답하지 않는다는 메시지가 표시되면 Amazon EC2 인스턴스의 보안 그룹에 인바운드 규칙이 있는지 확인합니다. 자세한 내용은 Amazon Linux 또는 RHEL Amazon EC2 인스턴스에 HTTP 트래픽을 허용하는 인바운드 규칙 추가 단원을 참조하십시오.

5단계: WordPress 애플리케이션 업데이트 및 재배포

이제 애플리케이션 수정 버전을 배포했으므로 개발 컴퓨터에서 WordPress의 코드를 업데이트한 다음 CodeDeploy를 사용하여 사이트를 다시 배포합니다. 그런 다음 Amazon EC2 인스턴스에서 코드 변경 사항을 확인해야 합니다.

주제

- WordPress 사이트 설정

- 사이트 수정
- 사이트 재배포

WordPress 사이트 설정

코드 변경의 효과를 보려면 완전한 기능을 설치할 수 있도록 WordPress 사이트 설정을 완료하세요.

1. 웹 브라우저에 사이트 URL을 입력합니다. URL은 Amazon EC2 인스턴스의 퍼블릭 DNS 주소에 `/WordPress` 확장자를 더한 것입니다. 이 예제 WordPress 사이트(예: Amazon EC2 인스턴스 퍼블릭 DNS 주소)의 경우 URL은 **`http://ec2-01-234-567-890.compute-1.amazonaws.com/WordPress`**입니다.
2. 사이트를 아직 설정하지 않은 경우 WordPress 기본 시작 페이지가 나타납니다. Let's go!를 선택합니다.
3. 기본 MySQL 데이터베이스를 사용하려면 데이터베이스 구성 페이지에서 다음 값을 입력합니다.

   - 데이터베이스 이름: **`test`**
   - 사용자 이름: **`root`**
   - 암호: 비워 둠
   - 데이터베이스 호스트: **`localhost`**
   - 테이블 접두사: **`wp_`**

   제출을 선택하여 데이터베이스를 설정합니다.
4. 사이트 설정을 계속합니다. 시작 페이지에서 원하는 값을 입력하고 WordPress 설치를 선택합니다. 설치가 완료되면 대시보드에 로그인할 수 있습니다.

> **Important**
>
> WordPress 애플리케이션을 배포하는 동안 **`change_permissions.sh`** 스크립트가 `/tmp/WordPress` 폴더의 사용 권한을 업데이트하여 누구나 쓸 수 있습니다. 이제 소유자만 쓸 수 있도록 다음 명령을 실행하여 사용 권한을 제한하는 것이 좋습니다.
>
> ```
> chmod -R 755 /var/www/html/WordPress
> ```

사이트 수정

WordPress 사이트를 수정하려면 개발 컴퓨터의 애플리케이션 폴더로 이동하세요.

```
cd /tmp/WordPress
```

사이트의 색상 중 일부를 수정하려면 `wp-content/themes/twentyfifteen/style.css` 파일에서 텍스트 편집기 또는 sed를 사용하여 `#fff`을(를) #768331(으)로 변경합니다.

Linux 또는 GNU가 sed인 다른 시스템에서에서는 다음을 사용합니다.

```
sed -i 's/#fff/#768331/g' wp-content/themes/twentyfifteen/style.css
```

MacOS, Unix 또는 BSD가 sed인 다른 시스템에서에서는 다음을 사용합니다.

```
sed -i '' 's/#fff/#768331/g' wp-content/themes/twentyfifteen/style.css
```

사이트 재배포

사이트의 코드를 수정했으므로 Amazon S3 및 CodeDeploy를 사용하여 사이트를 다시 배포합니다.

애플리케이션의 파일을 단일 아카이브 파일로 묶고 아카이브 파일을 푸시합니다.에 설명된 대로 Amazon S3 변경 사항을 번들로 묶어 업로드합니다. (이러한 지침을 따를 때 애플리케이션을 만들 필요가 없다는 것을 기억하세요.) 수정 버전에 이전과 동일한 키(**`WordPressApp.zip`**)를 지정하세요. 이전에 생성한 것과 동일한 Amazon S3 버킷(예:**`amzn-s3-demo-bucket`**)에 업로드합니다.

AWS CLI, CodeDeploy 콘솔 또는 CodeDeploy APIs를 사용하여 사이트를 재배포합니다.

주제

- 사이트를 재배포하려면(CLI)
- 사이트를 재배포하려면(콘솔)

사이트를 재배포하려면(CLI)

create-deployment 명령을 호출하여 새로 업로드한 수정 버전을 기반으로 배포를 만듭니다. **`WordPress_App`**(이)라는 애플리케이션, **`CodeDeployDefault.OneAtATime`**(이)라는 배포

구성, **WordPress_DepGroup**(이)라는 배포 그룹, **amzn-s3-demo-bucket**(이)라는 버킷의 **WordPressApp.zip**(이)라는 수정 버전을 사용합니다.

```
aws deploy create-deployment \
 --application-name WordPress_App \
 --deployment-config-name CodeDeployDefault.OneAtATime \
 --deployment-group-name WordPress_DepGroup \
 --s3-location bucket=amzn-s3-demo-bucket,bundleType=zip,key=WordPressApp.zip
```

배포 모니터링 및 문제 해결에 설명된 대로 배포의 상태를 확인할 수 있습니다.

CodeDeploy가 사이트를 재배포한 경우 웹 브라우저에서 사이트를 다시 방문하여 색상이 변경되었는지 확인합니다. (브라우저를 새로 고쳐야 할 수 있습니다.) 색상이 변경되었으면 제대로 수행된 것입니다. 사이트를 성공적으로 수정하고 재배포했습니다!

사이트를 재배포하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. 애플리케이션 목록에서 WordPress_App을 선택합니다.
4. 배포 그룹 탭에서 **WordPress_DepGroup**을(를) 선택합니다.
5. 배포 만들기를 선택합니다.
6. 배포 만들기 페이지에서,

   a. 배포 그룹에서 **WordPress_DepGroup**을(를) 선택합니다.

   b. 리포지토리 유형 영역에서 내 애플리케이션이 Amazon S3에 저장됨을 선택한 다음 수정 버전의 Amazon S3 링크를 수정 버전 위치 상자에 복사합니다. 링크 값을 찾으려면:

      i. 별도의 브라우저 탭에서:

         에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.

amzn-s3-demo-bucket을 찾아서 열고 수정 버전 **WordPressApp.zip**을 선택합니다.

ii. Amazon S3 콘솔에서 속성 창이 보이지 않으면, 속성 버튼을 선택합니다.

iii. 속성 패널에서 링크 필드의 값을 CodeDeploy 콘솔의 수정 버전 위치 상자로 복사합니다.

c. 파일 형식을 찾을 수 없다는 메시지가 표시되는 경우, .zip을 선택합니다.

d. 배포 설명 상자는 비워 둡니다.

e. 배포 그룹 재정의를 확장하고 배포 구성에서 CodeDeployDefault.OneAtATime을 선택합니다.

f. Start deployment(배포 시작)를 선택합니다. 새로 만든 배포에 대한 정보가 [Deployments] 페이지에 표시됩니다.

g. 배포 모니터링 및 문제 해결에 설명된 대로 배포의 상태를 확인할 수 있습니다.

CodeDeploy가 사이트를 재배포한 경우 웹 브라우저에서 사이트를 다시 방문하여 색상이 변경되었는지 확인합니다. (브라우저를 새로 고쳐야 할 수 있습니다.) 색상이 변경되었으면 제대로 수행된 것입니다. 사이트를 성공적으로 수정하고 재배포했습니다!

6단계: WordPress 애플리케이션 및 관련 리소스 정리

이제 WordPress 코드를 성공적으로 업데이트하고 사이트를 다시 배포했습니다. 이 튜토리얼에서 만든 리소스에 계속해서 비용이 부과되지 않도록 하려면 다음 항목을 삭제해야 합니다.

- 모든 CloudFormation 스택(또는 외부에서 생성한 경우 Amazon EC2 인스턴스 종료 CloudFormation).
- 모든 Amazon S3 버킷.
- CodeDeploy의 `WordPress_App` 애플리케이션.
- CodeDeploy 에이전트에 대한 AWS Systems Manager 상태 관리자 연결입니다.

AWS CLI, CloudFormation, Amazon S3, Amazon EC2 및 CodeDeploy 콘솔 또는 AWS APIs를 수행할 수 있습니다.

주제

- 리소스를 정리하려면(CLI)
- 리소스를 정리하려면(콘솔)
- 다음 단계

리소스를 정리하려면(CLI)

1. 이 자습서에서 CloudFormation 템플릿을 사용한 경우 라는 스택에 대해 delete-stack 명령을 호출합니다**CodeDeployDemoStack**. 그러면 함께 제공되는 모든 Amazon EC2 인스턴스가 종료되고 스택에서 생성한 모든 동반 IAM 역할이 삭제됩니다.

   ```
   aws cloudformation delete-stack --stack-name CodeDeployDemoStack
   ```

2. Amazon S3 버킷을 삭제하려면 **amzn-s3-demo-bucket**(이)라는 버킷에 대해 --recursive 스위치를 사용하여 rm 명령을 호출합니다. 버킷을 비롯해 버킷에 있는 모든 객체가 삭제됩니다.

   ```
   aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region
   ```

3. WordPress_App 애플리케이션을 삭제하려면 delete-application 명령을 호출합니다. 이렇게 하면 애플리케이션에 대해 연결된 배포 그룹 레코드와 배포 레코드가 모두 삭제됩니다.

   ```
   aws deploy delete-application --application-name WordPress_App
   ```

4. Systems Manager 상태 관리자 연결을 삭제하려면 delete-association 명령을 호출합니다.

   ```
   aws ssm delete-association --assocation-id association-id
   ```

   describe-association 명령을 호출하여 *association-id*를 얻을 수 있습니다.

   ```
   aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
    Key=tag:Name,Values=CodeDeployDemo
   ```

이 자습서에서 CloudFormation 스택을 사용하지 않은 경우 terminate-instances 명령을 호출하여 수동으로 생성한 Amazon EC2 인스턴스를 종료합니다. 종료할 Amazon EC2 인스턴스의 ID를 입력합니다.

```
aws ec2 terminate-instances --instance-ids instanceId
```

리소스를 정리하려면(콘솔)

이 자습서에서 CloudFormation 템플릿을 사용한 경우 연결된 CloudFormation 스택을 삭제합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.

2. 필터 상자에 이전에 생성한 CloudFormation 스택 이름(예:)을 입력합니다**CodeDeployDemoStack**.
3. 스택 이름 옆의 상자를 선택합니다. 작업 메뉴에서 스택 삭제를 선택합니다.

   CloudFormation 는 스택을 삭제하고, 함께 제공되는 모든 Amazon EC2 인스턴스를 종료하고, 함께 제공되는 모든 IAM 역할을 삭제합니다.

CloudFormation 스택 외부에서 생성한 Amazon EC2 인스턴스를 종료하려면:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 인스턴스 목록에서 인스턴스를 선택합니다.
3. 검색 상자에 종료할 Amazon EC2 인스턴스의 이름(예: **CodeDeployDemo**)을 입력한 후 Enter를 누릅니다.
4. Amazon EC2 인스턴스의 이름을 선택합니다.
5. [Actions] 메뉴에서 [Instance State]를 가리킨 다음 [Terminate]를 선택합니다. 메시지가 나타나면 [Yes, Terminate]를 선택합니다.

인스턴스마다 이들 단계를 반복합니다.

Amazon S3 버킷을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 이전에 생성한 Amazon S3 버킷의 이름(예: **amzn-s3-demo-bucket**)을 찾아서 선택합니다.
3. 버킷을 삭제하려면 먼저 버킷의 콘텐츠를 삭제해야 합니다. 버킷에 있는 모든 파일(예: **WordPressApp.zip**)을 선택합니다. 작업 메뉴에서 삭제를 선택합니다. 삭제 확인 메시지가 표시되면 확인을 선택합니다.
4. 버킷을 비운 후 버킷을 삭제할 수 있습니다. 버킷 목록에서 버킷 행을 선택합니다(버킷 이름은 아님). 버킷 삭제를 선택하고 확인하라는 메시지가 나타나면확인을 선택합니다.

CodeDeploy에서 `WordPress_App` 애플리케이션을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. 애플리케이션 목록에서 WordPress_App을 선택합니다.
4. 애플리케이션 세부 정보 페이지에서 애플리케이션 삭제를 선택합니다.
5. 메시지가 표시되면 삭제를 확인할 애플리케이션의 이름을 입력한 후 삭제를 선택합니다.

Systems Manager 상태 관리자 연결을 삭제하려면

1. https://console.aws.amazon.com/systems-manager AWS Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 상태 관리자를 선택합니다.
3. 생성한 연결을 선택하고 삭제를 선택합니다.

다음 단계

축하합니다! CodeDeploy 배포를 성공적으로 완료했고 사이트의 코드를 업데이트하고 다시 배포했습니다.

튜토리얼: CodeDeploy를 사용하여 "hello, world!" 애플리케이션 배포(Windows Server)

이 튜토리얼에서는 IIS(인터넷 정보 서비스)를 웹 서버로 실행하는 단일 Windows Server Amazon EC2 인스턴스에 단일 웹 페이지를 배포합니다. 이 웹 페이지는 간단한 “Hello, World!” 메시지를 표시합니다.

찾는 항목이 보이지 않습니까?

- Amazon Linux 또는 Red Hat Enterprise Linux(RHEL) Amazon EC2 인스턴스에 배포하는 방법을 연습하려면 튜토리얼: Amazon EC2 인스턴스(Amazon Linux 또는 Red Hat Enterprise Linux 및 Linux, macOS 또는 Unix)에 WordPress 배포 단원을 참조하세요.
- 온-프레미스 인스턴스에 배포하는 방법을 연습하려면 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux) 단원을 참조하세요.

이 튜토리얼의 단계는 Windows 관점에서 제공됩니다. Linux, macOS 또는 Unix를 실행하는 로컬 컴퓨터에서 이러한 단계를 대부분 완료할 수 있지만 `c:\temp`와(과) 같은 Windows 기반 디렉터리 경로를 포함하는 경로를 조정해야 합니다. 또한 Amazon EC2 인스턴스에 연결하려면 RDP(원격 데스크톱 프로토콜)를 통해 Windows Server를 실행하는 Amazon EC2 인스턴스에 연결할 수 있는 클라이언트 애플리케이션이 필요합니다. Windows에는 기본적으로 RDP 연결 클라이언트 애플리케이션이 포함되어 있습니다.

이 자습서를 시작하기 전에 사용자 구성, 설치 또는 업그레이드, IAM 인스턴스 프로파일 및 서비스 역할 AWS CLI생성 CodeDeploy 시작하기등의 사전 조건을 완료해야 합니다.

주제

- 1단계: Windows Server Amazon EC2 인스턴스 시작
- 2단계: Windows Server Amazon EC2 인스턴스에 배포하도록 원본 콘텐츠 구성
- 3단계: "hello, world!" 애플리케이션을 Amazon S3에 업로드
- 4단계: Hello World 애플리케이션 배포
- 5 단계: "hello, world!" 애플리케이션 업데이트 및 재배포
- 6단계: "hello, world!" 애플리케이션 및 관련 리소스 정리

1단계: Windows Server Amazon EC2 인스턴스 시작

Hello, World 애플리케이션을 CodeDeploy로 배포하려면 Windows Server를 실행하는 Amazon EC2 인스턴스가 필요합니다.

CodeDeploy를 위한 Amazon EC2 인스턴스 생성의 지침을 따르세요. Amazon EC2 인스턴스 태그를 인스턴스에 할당할 준비가 되면 태그 키 **`Name`**와(과) 태그 값 **`CodeDeployDemo`**을(를) 지정해야 합니다. 태그 키 또는 태그 값을 다르게 지정한 경우 4단계: Hello World 애플리케이션 배포의 지침을 따르면 예기치 않은 결과가 발생할 수 있습니다.

Amazon EC2 인스턴스를 시작한 후에는 이 페이지로 돌아와 계속해서 다음 섹션으로 진행합니다. 다음 단계로 CodeDeploy를 사용하여 애플리케이션 생성 단원을 진행하지 마십시오.

Amazon EC2 인스턴스에 연결

Amazon EC2 인스턴스 시작 후 다음 지침에 따라 해당 인스턴스에 연결하는 연습을 합니다.

> **Note**
>
> 다음 지침에서는 Windows와 Windows Desktop Connection 클라이언트 애플리케이션을 실행한다고 가정합니다. 자세한 내용은 RDP를 사용하여 Windows 인스턴스에 연결 단원을 참조하세요. 다른 운영 체제 또는 기타 RDP 연결 클라이언트 애플리케이션의 경우 다음 지침을 조정해야 할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 인스턴스에서 인스턴스를 선택합니다.
3. 목록에서 Windows Server 인스턴스를 찾아서 선택하세요.
4. 연결을 선택합니다.
5. 암호 가져오기를 선택한 다음 파일 선택을 선택합니다.
6. Windows Server Amazon EC2 인스턴스와 연결된 Amazon EC2 인스턴스 키 페어 파일을 찾아서 선택한 후 열기를 선택하세요.
7. 암호 해독을 선택합니다. 표시되는 암호를 기록해 둡니다. 10단계에서 이 이름이 필요합니다.
8. [Download Remote Desktop File]을 선택한 후 파일을 엽니다.
9. 원격 연결 게시자를 식별할 수 없더라도 연결하라는 메시지가 표시되면 계속 진행합니다.
10. 7단계 메모해 둔 암호를 입력한 다음, 계속 진행합니다. (RDP 연결 클라이언트 애플리케이션에 사용자 이름을 입력하라는 메시지가 표시되면 **`Administrator`**를 입력하세요.)
11. 원격 컴퓨터의 자격 증명을 확인할 수 없더라도 연결하라는 메시지가 표시되면 계속 진행합니다.
12. 연결된 후에는 Windows Server를 실행하는 Amazon EC2 인스턴스의 데스크톱이 표시됩니다.
13. 이제 Amazon EC2 인스턴스와의 연결을 해제할 수 있습니다.

> **Warning**
>
> 인스턴스를 중지하거나 종료하지 마십시오. 인스턴스를 중지하거나 종료할 경우 CodeDeploy가 인스턴스에 배포할 수 없습니다.

Windows Server Amazon EC2 인스턴스에 HTTP 트래픽을 허용하는 인바운드 규칙 추가

다음 단계에서는 브라우저의 Windows Server Amazon EC2 인스턴스에서 배포된 웹페이지를 볼 수 있도록 Amazon EC2 인스턴스에 열린 HTTP 포트가 있는지 확인합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 인스턴스를 선택한 후 인스턴스를 선택합니다.
3. 설명 탭의 보안 그룹에서 인바운드 규칙 보기를 선택합니다.

   보안 그룹에 다음과 같은 규칙 목록이 있어야 합니다.

   ```
   Security Groups associated with i-1234567890abcdef0
    Ports     Protocol     Source     launch-wizard-N
    22        tcp          0.0.0.0/0          #
   ```

4. 보안 그룹에서 Amazon EC2 인스턴스를 위한 보안 그룹을 선택합니다. 이름은 **launch-wizard-*N***이 될 수 있습니다. 이름의 ***N***은 인스턴스가 생성될 때 보안 그룹에 할당된 번호입니다.

   인바운드 탭을 선택합니다. 인스턴스의 보안 그룹이 올바르게 구성되어 있으면 다음 값이 있는 규칙이 표시되어야 합니다.

   - 유형: HTTP
   - 프로토콜: TCP
   - 포트 범위: 80
   - 소스: 0.0.0.0/0
5. 이러한 값이 있는 규칙이 없는 경우 보안 그룹에 규칙 추가의 절차를 사용하여 해당 값을 새 보안 규칙에 추가하세요.

2단계: Windows Server Amazon EC2 인스턴스에 배포하도록 원본 콘텐츠 구성

이제 Amazon EC2 인스턴스에 배포할 수 있도록 애플리케이션의 소스 콘텐츠를 구성해야 합니다. 이 튜토리얼에서는 Windows Server를 실행하는 Amazon EC2 인스턴스에 단일 웹 페이지를 배포합니다. 이 인스턴스는 IIS(인터넷 정보 서비스)를 웹 서버로 실행합니다. 이 웹 페이지는 간단한 “Hello, World!” 메시지를 표시합니다.

주제

- 웹 페이지 만들기
- 애플리케이션을 실행하는 스크립트 만들기
- 애플리케이션 사양 파일 추가

웹 페이지 만들기

1. c:\temp 폴더에 HelloWorldApp(이)라는 하위 디렉터리(하위 폴더)를 만든 다음 다음 해당 폴더로 전환합니다.

```
mkdir c:\temp\HelloWorldApp
cd c:\temp\HelloWorldApp
```

> **Note**
>
> c:\temp 또는 HelloWorldApp(이)라는 하위 폴더의 위치를 사용할 필요가 없습니다. 다른 위치 또는 하위 폴더를 사용하는 경우 이 자습서 전체에서 이를 사용해야 합니다.

2. 텍스트 편집기를 사용하여 폴더 내에 파일을 생성합니다. 파일 이름을 index.html으로 지정합니다.

```
notepad index.html
```

3. 파일에 다음 코드를 추가하고 파일을 저장합니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
<head>
  <title>Hello, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #0188cc;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
  </style>
</head>
```

```
<body>
  <div align="center"><h1>Hello, World!</h1></div>
  <div align="center"><h2>You have successfully deployed an application using
 CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

애플리케이션을 실행하는 스크립트 만들기

다음으로 CodeDeploy가 대상 Amazon EC2 인스턴스에 웹 서버를 설정하는 데 사용할 스크립트를 생성합니다.

1. 동일한 하위 폴더에서 `index.html` 파일이 저장되면 텍스트 편집기를 사용하여 다른 파일을 생성합니다. 파일 이름을 `before-install.bat`으로 지정합니다.

   ```
   notepad before-install.bat
   ```

2. 파일에 다음 배치 스크립트 코드를 추가하고 파일을 저장합니다.

   ```
   REM Install Internet Information Server (IIS).
   c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Import-Module -
   Name ServerManager
   c:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe -Command Install-
   WindowsFeature Web-Server
   ```

애플리케이션 사양 파일 추가

그런 다음 웹 페이지 및 배치 스크립트 파일 외에 애플리케이션 사양 파일(AppSpec file)을 추가합니다. AppSpec 파일은 CodeDeploy에서 다음을 위해 사용하는 YAML 형식 파일입니다.

- 애플리케이션 수정 버전의 소스 파일을 인스턴스의 대상으로 매핑합니다.
- 배포 중에 인스턴스에서 실행할 스크립트를 지정합니다.

AppSpec 파일의 이름은 `appspec.yml`이어야 합니다. 애플리케이션 소스 코드의 루트 폴더에 있어야 합니다.

1. 동일한 하위 폴더에서 `index.html` 및 `before-install.bat` 파일이 저장되면 텍스트 편집기를 사용하여 다른 파일을 생성합니다. 파일 이름을 `appspec.yml`으로 지정합니다.

```
notepad appspec.yml
```

2. 파일에 다음 YAML 코드를 추가하고 파일을 저장합니다.

```
version: 0.0
os: windows
files:
  - source: \index.html
    destination: c:\inetpub\wwwroot
hooks:
  BeforeInstall:
    - location: \before-install.bat
      timeout: 900
```

CodeDeploy는 이 AppSpec 파일을 사용하여 `index.html` 파일을 애플리케이션 소스 코드의 루트 폴더에서 대상 Amazon EC2 인스턴스의 `c:\inetpub\wwwroot` 폴더로 복사합니다. 배포하는 동안 CodeDeploy는 **`BeforeInstall`** 배포 수명 주기 이벤트 중에 `before-install.bat` 배치 스크립트를 대상 Amazon EC2 인스턴스에서 실행합니다. 이 스크립트를 실행하는 데 900초(15분) 이상 걸리면 CodeDeploy는 배포를 중지하고 Amazon EC2 인스턴스에 대한 배포를 실패로 표시합니다.

이러한 설정에 대한 자세한 정보는 CodeDeploy AppSpec 파일 참조 단원을 참조하세요.

> **Important**
>
> 이 파일에 있는 각 항목 사이의 공백 위치와 개수는 중요합니다. 공백이 올바르지 않으면 CodeDeploy는 디버깅하기 어려운 오류를 발생시킵니다. 자세한 내용은 AppSpec 파일 간격 단원을 참조하십시오.

3단계: "hello, world!" 애플리케이션을 Amazon S3에 업로드

이제 CodeDeploy가 배포할 수 있는 위치에 소스 콘텐츠를 준비하고 업로드합니다. 다음 지침은 Amazon S3 버킷을 프로비저닝하고, 버킷에 대한 애플리케이션 수정 버전 파일을 준비하고, 수정 버전의 파일을 번들로 묶은 다음, 버전을 버킷에 푸시하는 방법을 보여 줍니다.

> **Note**
>
> 이 튜토리얼에서 다루지 않지만 CodeDeploy를 사용하여 애플리케이션을 GitHub 리포지토리에서 인스턴스로 배포할 수 있습니다. 자세한 내용은 GitHub와 CodeDeploy 통합 단원을 참조하세요.

주제

- Amazon S3 버킷 프로비저닝
- 버킷에 대한 애플리케이션 파일 준비
- 애플리케이션의 파일을 단일 아카이브 파일로 묶고 아카이브 파일을 푸시합니다.

Amazon S3 버킷 프로비저닝

스토리지 컨테이너 또는 Amazon S3의 버킷을 만듭니다(또는 기존 버킷 사용). 버킷에 수정 버전을 업로드할 수 있는지, 배포에 사용된 Amazon EC2 인스턴스가 버킷에서 수정 버전을 다운로드할 수 있는지 확인합니다.

AWS CLI, Amazon S3 콘솔 또는 Amazon S3 APIs를 사용하여 Amazon S3 버킷을 생성할 수 있습니다. 버킷을 생성한 후에는 버킷과 CodeDeploy 사용자에게 액세스 권한을 부여해야 합니다.

> **Note**
>
> 버킷 이름은 모든 AWS 계정의 Amazon S3에서 고유해야 합니다. **`amzn-s3-demo-bucket`**을(를) 사용할 수 없는 경우, 다른 버킷 이름(예: **`amzn-s3-demo-bucket`** 뒤에 대시와 이니셜 또는 다른 고유 식별자가 옴)을 사용하세요. 그런 다음 이 자습서 전체에서 버킷 이름을 **`amzn-s3-demo-bucket`**(으)로 대체해야 합니다.
>
> Amazon S3 버킷은 대상 Amazon EC2 인스턴스가 시작되는 리전과 동일한 AWS 리전에서 생성해야 합니다. 예를 들어 버킷을 미국 동부(버지니아) 리전에 생성한 경우 대상 Amazon EC2 인스턴스를 미국 동부(버지니아 북부) 리전에서 시작해야 합니다.

주제

- Amazon S3 버킷을 생성하려면(CLI)
- Amazon S3 버킷을 생성하려면(콘솔)
- Amazon S3 버킷 및 AWS 계정에 권한 부여

Amazon S3 버킷을 생성하려면(CLI)

mb 명령을 호출하여 이름이 **amzn-s3-demo-bucket**인 Amazon S3 버킷을 생성합니다.

```
aws s3 mb s3://amzn-s3-demo-bucket --region region
```

Amazon S3 버킷을 생성하려면(콘솔)

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. Amazon S3; 콘솔에서 버킷 생성을 선택합니다.
3. 버킷 이름 상자에서 버킷의 이름을 입력합니다.
4. 리전 목록에서 대상 리전을 선택한 다음 생성을 선택합니다.

Amazon S3 버킷 및 AWS 계정에 권한 부여

또한 Amazon S3 버킷으로 업로드할 수 있는 권한이 있어야 합니다. Amazon S3 버킷 정책을 통해 이러한 권한을 지정할 수 있습니다. 예를 들어 다음 Amazon S3 버킷 정책에서 와일드카드 문자(*)를 사용하면 AWS 계정이 라는 Amazon S3 버킷의 모든 디렉터리에 파일을 111122223333 업로드할 수 있습니다amzn-s3-demo-bucket.

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

AWS 계정 ID를 보려면 AWS 계정 ID 찾기를 참조하세요.

이제 Amazon S3 버킷이 참여하는 각 Amazon EC2 인스턴스에서 다운로드 요청을 허용하는지 확인하는 것이 좋습니다. Amazon S3 버킷 정책을 통해 이를 지정할 수 있습니다. 예를 들어, 다음 Amazon S3 버킷 정책에서 와일드카드 문자(*)를 사용하면 `arn:aws:iam::444455556666:role/CodeDeployDemo` ARN이 포함된 IAM 인스턴스 프로파일이 연결된 Amazon EC2 인스턴스가 이름이 `amzn-s3-demo-bucket`인 Amazon S3 버킷의 디렉터리에서 파일을 다운로드할 수 있습니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

Amazon S3 버킷 정책 생성 및 연결에 대한 자세한 내용은 버킷 정책 예제를 참조하세요.

1단계: 설정에서 만든 CodeDeploy 관리 사용자에게도 Amazon S3 버킷으로 개정 버전을 업로드할 수 있는 권한이 있어야 합니다. 이를 지정하는 한 가지 방법은 사용자의 권한 세트나 IAM 역할(사용자에게 수임을 허용한 역할)에 추가하는 IAM 정책을 사용하는 것입니다. 다음 IAM 정책은 사용자가 `amzn-s3-demo-bucket`이라는 Amazon S3 버킷의 어디에나 개정 버전을 업로드할 수 있도록 허용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":["s3:PutObject"],
      "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
```

```
  ]
}
```

IAM 정책을 생성하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성을 참조하세요. 권한 세트에 정책을 추가하는 방법에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 Create a permission set(권한 세트 생성)를 참조하세요.

버킷에 대한 애플리케이션 파일 준비

웹 페이지, AppSpec 파일 및 스크립트가 개발 컴퓨터에서 다음과 같이 구성되어 있는지 확인하세요.

```
c:\
  |-- temp\
        |--HelloWorldApp\
             |-- appspec.yml
             |-- before-install.bat
             |-- index.html
```

애플리케이션의 파일을 단일 아카이브 파일로 묶고 아카이브 파일을 푸시합니다.

파일을 아카이브 파일(애플리케이션 수정 버전이라고도 함)로 번들링합니다.

> **Note**
>
> 객체를 버킷에 저장하거나 애플리케이션 수정 버전을 버킷으로 전송한 경우 요금이 부과될 수 있습니다. 자세한 내용은 Amazon S3 요금을 참조하세요.

1. 개발 컴퓨터에서 파일이 저장된 폴더로 전환하세요.

   ```
   cd c:\temp\HelloWorldApp
   ```

   > **Note**
   >
   > 이 폴더로 전환하지 않으면 현재 폴더에서 파일 번들링이 시작됩니다. 예를 들어 현재 폴더가 `c:\temp\HelloWorldApp` 대신 `c:\temp`이면 `c:\temp` 폴더의 파일 및 하위 폴더로 번들링이 시작되어 `HelloWorldApp` 하위 폴더보다 더 많은 폴더를 포함할 수 있습니다.

2. create-application 명령을 호출하여 CodeDeploy에서 이름이 **HelloWorld_App**인 새 애플리케이션을 등록합니다.

```
aws deploy create-application --application-name HelloWorld_App
```

3. CodeDeploy push 명령을 호출하여 파일을 묶고, 수정 버전을 Amazon S3로 업로드하고, 업로드된 수정 버전에 대한 정보를 CodeDeploy에 등록할 수 있습니다. 이 모든 작업을 한 번에 수행할 수 있습니다.

```
aws deploy push --application-name HelloWorld_App --s3-location s3://amzn-s3-demo-
bucket/HelloWorld_App.zip --ignore-hidden-files
```

이 명령은 현재 디렉터리의 파일(숨겨진 파일 제외)을 단일 아카이브 파일(이름: `HelloWorld_App.zip`)로 번들링하고 수정 버전을 **amzn-s3-demo-bucket** 버킷에 업로드하고 업로드된 수정 버전에 대한 정보를 CodeDeploy에 등록합니다.

4단계: Hello World 애플리케이션 배포

이제 Amazon S3에 업로드한 샘플 Hello, World 애플리케이션 수정 버전을 배포합니다. AWS CLI 또는 CodeDeploy 콘솔을 사용하여 수정 버전을 배포하고 배포 진행 상황을 모니터링합니다. 애플리케이션 수정 배포에 성공한 후에는 결과를 확인합니다.

주제

- CodeDeploy로 애플리케이션 수정 버전 배포
- 배포 모니터링 및 문제 해결
- 배포 확인

CodeDeploy로 애플리케이션 수정 버전 배포

CLI 또는 콘솔을 사용하여 애플리케이션을 배포할 수 있습니다.

주제

- 애플리케이션 수정을 배포하려면(CLI)
- 애플리케이션 수정을 배포하려면(콘솔)

애플리케이션 수정을 배포하려면(CLI)

1. 먼저 배포에는 배포 그룹이 필요합니다. 그러나 배포 그룹을 만들기 전에 서비스 역할 ARN이 필요합니다. 서비스 역할은 서비스에 사용자를 대신할 수 있는 권한을 부여하는 IAM 역할입니다. 이 경우에 서비스 역할은 CodeDeploy에 Amazon EC2 인스턴스에 액세스하여 Amazon EC2 인스턴스 태그를 확장할(읽을) 수 있는 권한을 부여합니다.

   서비스 역할 생성(CLI)의 지침에 따라 앞에서 서비스 역할을 생성했습니다. 서비스 역할의 ARN을 확인하려면 서비스 역할 ARN 확인(CLI) 단원을 참조하세요.

2. 이제 ARN이 있으므로 해당 서비스 역할 ARN으로 create-deployment-group 명령을 호출하고, **CodeDeployDemo**(이)라는 Amazon EC2 인스턴스 태그와 **CodeDeployDefault.OneAtATime**(이)라는 배포 구성을 사용하여 **HelloWorld_App**(이)라는 애플리케이션과 연결된 **HelloWorld_DepGroup**(이)라는 배포 그룹을 만듭니다.

   ```
   aws deploy create-deployment-group --application-name HelloWorld_App
    --deployment-group-name HelloWorld_DepGroup --deployment-
   config-name CodeDeployDefault.OneAtATime --ec2-tag-filters
    Key=Name,Value=CodeDeployDemo,Type=KEY_AND_VALUE --service-role-arn serviceRoleARN
   ```

   > **Note**
   >
   > create-deployment-group 명령은 배포 및 인스턴스의 지정된 이벤트에 대해 주제의 구독자에게 Amazon SNS 알림을 전송하게 하는 트리거를 생성합니다. 또한 해당 명령은 Amazon CloudWatch 경보 모니터링 임계값에 도달한 경우 자동으로 배포를 롤백하고 배포를 중지하도록 경보를 설정하는 옵션을 지원합니다. 이 작업에 대한 명령은 이 자습서에 포함되지 않습니다.

3. 배포를 만들기 전에 배포 그룹의 인스턴스에 CodeDeploy 에이전트가 설치되어 있어야 합니다. AWS Systems Manager 를 사용하여 명령줄에서 다음 명령으로 에이전트를 설치할 수 있습니다.

   ```
   aws ssm create-association --name AWS-ConfigureAWSPackage
    --targets Key=tag:Name,Values=CodeDeployDemo --parameters
    action=Install,name=AWSCodeDeployAgent --schedule-expression "cron(0 2 ? * SUN
    *)"
   ```

   이 명령은 Systems Manager 상태 관리자에 연결을 생성하여 CodeDeploy 에이전트를 설치하고 매주 일요일 오전 2시에 업데이트를 시도합니다. CodeDeploy 에이전트에 대한 자세한 내용

은 CodeDeploy 에이전트 작업을 참조하세요. Systems Manager에 대한 자세한 내용은 AWS Systems Manager란 무엇입니까를 참조하세요.

4. 이제 create-deployment 명령을 호출하고 **HelloWorld_App**이라는 버킷에 있는 **CodeDeployDefault.OneAtATime**이라는 애플리케이션 수정을 사용하여 **HelloWorld_DepGroup**이라는 애플리케이션, **HelloWorld_App.zip**이라는 배포 구성 및 **amzn-s3-demo-bucket**이라는 배포 그룹과 연결된 배포를 만듭니다.

```
aws deploy create-deployment --application-name HelloWorld_App --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-
location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip
```

애플리케이션 수정을 배포하려면(콘솔)

1. CodeDeploy 콘솔을 사용하여 애플리케이션 수정을 배포하기 전에 서비스 역할 ARN이 필요합니다. 서비스 역할은 서비스에 사용자를 대신할 수 있는 권한을 부여하는 IAM 역할입니다. 이 경우에 서비스 역할은 CodeDeploy에 Amazon EC2 인스턴스에 액세스하여 Amazon EC2 인스턴스 태그를 확장할(읽을) 수 있는 권한을 부여합니다.

   서비스 역할 생성(콘솔)의 지침에 따라 앞에서 서비스 역할을 생성했습니다. 서비스 역할의 ARN을 확인하려면 서비스 역할 ARN 확인(콘솔) 단원을 참조하세요.
2. 이제 ARN이 있으므로 CodeDeploy 콘솔을 사용하여 애플리케이션 수정을 배포할 수 있습니다.

   에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

3. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
4. HelloWorld_App을 선택합니다.
5. 배포 그룹 탭에서 Create deployment group(배포 그룹 생성)을 선택합니다.
6. Deployment group name(배포 그룹 이름)에 **HelloWorld_DepGroup**을 입력합니다.
7. 서비스 역할에서 서비스 역할의 이름을 선택합니다.
8. 배포 유형에서 In-place(현재 위치)를 선택합니다.
9. 환경 구성에서 Amazon EC2 인스턴스를 선택합니다.

10. 를 사용한 에이전트 구성 AWS Systems Manager에서 기본값을 유지합니다.
11. 키에 **Name**을(를) 입력합니다.
12. 값에 **CodeDeployDemo**을(를) 입력합니다.
13. 배포 구성에서 CodeDeployDefault.OneAtATime을 선택합니다.
14. 로드 밸런서에서 Enable load balancing(로드 밸런싱 활성화)을 선택 해제합니다.
15. [Create deployment group]을 선택합니다.
16. 배포 만들기를 선택합니다.
17. 배포 그룹에서 HelloWorld_DepGroup을 선택합니다.
18. 수정 유형에서 애플리케이션을 Amazon S3에 저장을 선택한 후 수정 버전 위치에 이전에 Amazon S3에 업로드한 샘플 Hello, World 애플리케이션 수정 버전의 위치를 입력합니다. 위치를 가져오려면:

    a. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
    b. 버킷 목록에서 amzn-s3-demo-bucket(또는 애플리케이션 수정 버전을 업로드한 버킷 이름)을 선택합니다.
    c. 객체 목록에서 [HelloWorld_App.zip]을 선택합니다.
    d. 개요 탭에서 경로 복사를 선택합니다.
    e. CodeDeploy 콘솔로 돌아가서 수정 위치에 링크 필드 값을 붙여 넣습니다.
19. 수정 파일 형식으로 .zip을 선택합니다.
20. (선택 사항) 배포 설명에 설명을 입력합니다.
21. 배포 만들기를 선택합니다. 새로 만든 배포에 대한 정보가 [Deployments] 페이지에 표시됩니다.

배포 모니터링 및 문제 해결

AWS CLI 또는 콘솔을 사용하여 배포를 모니터링하고 문제를 해결합니다.

주제

- 배포를 모니터링하고 문제를 해결하려면(CLI)
- 배포를 모니터링하고 문제를 해결하려면(콘솔)

배포를 모니터링하고 문제를 해결하려면(CLI)

1. **HelloWorld_App**이라는 애플리케이션 및 **HelloWorld_DepGroup**이라는 배포 그룹에 대해 list-deployments 명령을 호출하여 배포 ID를 가져옵니다.

```
aws deploy list-deployments --application-name HelloWorld_App --deployment-group-
name HelloWorld_DepGroup --query "deployments" --output text
```

2. 배포 ID로 get-deployment 명령을 호출합니다.

```
aws deploy get-deployment --deployment-id deploymentID --query
 "deploymentInfo.status" --output text
```

3. 이 명령으로 배포의 전체 상태가 반환됩니다. 성공하면 `Succeeded` 값이 반환됩니다.

   전체 상태가 `Failed`이면 list-deployment-instances 및 get-deployment-instance와 같은 명령을 호출하여 문제를 해결할 수 있습니다. 문제 해결 옵션을 더 보려면 인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석을 참조하세요.

배포를 모니터링하고 문제를 해결하려면(콘솔)

CodeDeploy 콘솔의 배포 페이지에 있는 상태 열에서 배포 상태를 모니터링할 수 있습니다.

배포에 대한 자세한 정보를 확인하려면(특히, [Status] 열 값이 [Succeeded]가 아닌 경우):

1. 배포 테이블에서 배포 ID를 선택합니다. 배포에 실패하면 실패 원인을 설명하는 메시지가 배포 세부 정보 페이지에 표시됩니다.
2. 배포의 인스턴스에 대한 자세한 정보가 표시됩니다. 배포에 실패한 후 배포에 실패한 Amazon EC2 인스턴스와 해당 단계를 확인할 수 있습니다.
3. 문제 해결을 더 수행해야 할 경우 View Instance Details와 같은 기능을 사용할 수 있습니다. 또한 Amazon EC2 인스턴스의 배포 로그 파일을 분석할 수도 있습니다. 자세한 내용은 인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석 단원을 참조하세요.

배포 확인

배포에 성공한 후 설치가 작동 중인지 확인합니다. Amazon EC2 인스턴스의 퍼블릭 DNS 주소를 사용해 웹 브라우저에서 웹 페이지를 확인합니다. (퍼블릭 DNS 값을 확인하려면 Amazon EC2 콘솔에서 Amazon EC2 인스턴스를 선택하고 설명 탭에서 퍼블릭 DNS의 값을 찾습니다.)

예를 들어, Amazon EC2 인스턴스의 퍼블릭 DNS 주소가 **`ec2-01-234-567-890.compute-1.amazonaws.com`**이면 다음 URL을 사용합니다.

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

성공하면 Hello, World 웹 페이지가 표시되어야 합니다.

5 단계: "hello, world!" 애플리케이션 업데이트 및 재배포

이제 애플리케이션 수정 버전을 배포했으므로 개발 컴퓨터에서 웹 페이지의 코드를 업데이트한 다음 CodeDeploy를 사용하여 사이트를 다시 배포합니다. 재배포한 후에는 Amazon EC2 인스턴스에서 변경 내용을 확인할 수 있어야 합니다.

주제

- 웹 페이지 수정
- 사이트 재배포

웹 페이지 수정

1. `c:\temp\HelloWorldApp` 하위 폴더로 이동하고 텍스트 편집기를 사용하여 `index.html` 파일을 수정합니다.

```
cd c:\temp\HelloWorldApp
notepad index.html
```

2. `index.html` 파일의 콘텐츠를 수정하여 웹 페이지의 배경 색과 텍스트 일부를 변경한 다음, 파일을 저장합니다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
<head>
  <title>Hello Again, World!</title>
  <style>
    body {
      color: #ffffff;
      background-color: #66cc00;
      font-family: Arial, sans-serif;
      font-size:14px;
    }
  </style>
</head>
<body>
  <div align="center"><h1>Hello Again, World!</h1></div>
```

```
  <div align="center"><h2>You have successfully deployed a revision of an
 application using CodeDeploy</h2></div>
  <div align="center">
    <p>What to do next? Take a look through the <a href="https://aws.amazon.com/
codedeploy">CodeDeploy Documentation</a>.</p>
  </div>
</body>
</html>
```

사이트 재배포

코드를 수정했으므로 Amazon S3 및 CodeDeploy를 사용하여 웹 페이지를 다시 배포합니다.

애플리케이션의 파일을 단일 아카이브 파일로 묶고 아카이브 파일을 푸시합니다.에 설명된 대로 Amazon S3 변경 사항을 번들로 묶어 업로드합니다. 이러한 지침을 따르면 새 애플리케이션을 만들지 않아도 됩니다. 수정 버전에 이전과 동일한 키(**HelloWorld_App.zip**)를 지정하세요. 이전에 생성한 것과 동일한 Amazon S3 버킷(예:**amzn-s3-demo-bucket**)에 업로드합니다.

AWS CLI 또는 CodeDeploy 콘솔을 사용하여 사이트를 재배포합니다.

주제

- 사이트를 재배포하려면(CLI)
- 사이트를 재배포하려면(콘솔)

사이트를 재배포하려면(CLI)

create-deployment 명령을 호출하고 다시 **HelloWorld_App**(이)라는 애플리케이션, **CodeDeployDefault.OneAtATime**(이)라는 배포 구성, **HelloWorld_DepGroup**(이)라는 배포 그룹, **amzn-s3-demo-bucket**(이)라는 버킷에 있는 **HelloWorld_App.zip**(이)라는 애플리케이션 수정 버전을사용하여 업로드된 수정 버전을 기반으로 배포를 생성합니다.

```
 aws deploy create-deployment --application-name HelloWorld_App --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name HelloWorld_DepGroup --s3-
location bucket=amzn-s3-demo-bucket,bundleType=zip,key=HelloWorld_App.zip
```

배포 모니터링 및 문제 해결에 설명된 대로 새 배포의 상태를 확인할 수 있습니다.

CodeDeploy가 사이트를 재배포한 경우 웹 브라우저에서 사이트를 다시 방문하여 웹 페이지의 배경색과 텍스트가 변경되었는지 확인합니다. (브라우저를 새로 고쳐야 할 수 있습니다.) 배경색과 텍스트가 변경되었으면 제대로 수행된 것입니다. 사이트를 수정하고 재배포했습니다!

사이트를 재배포하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 애플리케이션을 선택합니다.
3. 애플리케이션 목록에서 HelloWorld_App을 선택합니다.
4. 배포 탭에서 배포 만들기를 선택합니다.

   a. 배포 그룹 목록에서 HelloWorld_DepGroup을 선택합니다.

   b. 수정 버전 위치에 수정 버전에 대한 Amazon S3 링크를 입력합니다.

      링크 값을 찾으려면:

      i. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.

         amzn-s3-demo-bucket을 찾아서 열고 Amazon S3 콘솔에서 수정 버전 **`HelloWorld_App.zip`**을 선택합니다.

      ii. Amazon S3 콘솔에서 속성 창이 보이지 않으면, 속성 버튼을 선택합니다.

      iii. 속성 창에서 링크 필드의 값을 복사합니다.

      iv. CodeDeploy 콘솔로 돌아가서 수정 버전 위치에 링크를 붙여 넣습니다.

      v.

   c. 수정 버전 파일 형식에서 파일 형식을 찾을 수 없다는 메시지가 표시되는 경우, .zip을 선택합니다.

   d. 배포 설명은 비워 둡니다.

   e. 배포 구성 목록의 배포 그룹 재정의를 확장하고 CodeDeployDefault.OneAtATime를 선택한 다음 배포 만들기를 선택합니다.

배포 모니터링 및 문제 해결에 설명된 대로 배포의 상태를 확인할 수 있습니다.

CodeDeploy가 사이트를 재배포한 경우 웹 브라우저에서 사이트를 다시 방문하여 웹 페이지의 배경색과 텍스트가 변경되었는지 확인합니다. (브라우저를 새로 고쳐야 할 수 있습니다.) 배경색과 텍스트가 변경되었으면 제대로 수행된 것입니다. 사이트를 수정하고 재배포했습니다!

6단계: "hello, world!" 애플리케이션 및 관련 리소스 정리

이제 “Hello, World!” 코드를 업데이트했고 사이트를 다시 배포했습니다. 이 자습서를 완료하기 위해 만든 리소스에 계속해서 비용이 부과되지 않도록 하려면 다음 항목을 삭제해야 합니다.

- 모든 CloudFormation 스택(또는 외부에서 생성한 경우 Amazon EC2 인스턴스 종료 CloudFormation).
- 모든 Amazon S3 버킷.
- CodeDeploy의 `HelloWorld_App` 애플리케이션.
- CodeDeploy 에이전트에 대한 AWS Systems Manager 상태 관리자 연결입니다.

AWS CLI, CloudFormation, Amazon S3, Amazon EC2 및 CodeDeploy 콘솔 또는 AWS APIs를 수행할 수 있습니다.

주제

- 정리 리소스를 사용하려면(CLI)
- 리소스를 정리하려면(콘솔)
- 다음 단계

정리 리소스를 사용하려면(CLI)

1. 이 자습서에서 CloudFormation 스택을 사용한 경우 라는 스택에 대해 delete-stack 명령을 호출하여 스택을 삭제합니다**`CodeDeployDemoStack`**. 이렇게 하면 수반되는 모든 Amazon EC2 인스턴스가 종료되고 스택에서 원래 생성된 모든 동반 IAM 역할이 삭제됩니다.

```
aws cloudformation delete-stack --stack-name CodeDeployDemoStack
```

2. Amazon S3 버킷을 삭제하려면 **amzn-s3-demo-bucket**(이)라는 버킷에 대해 --recursive 스위치를 사용하여 rm 명령을 호출합니다. 버킷을 비롯해 버킷에 있는 모든 객체가 삭제됩니다.

```
aws s3 rm s3://amzn-s3-demo-bucket --recursive --region region
```

3. CodeDeploy 에서 HelloWorld_App 애플리케이션을 삭제하려면 delete-application 명령을 호출합니다. 이렇게 하면 애플리케이션에 대해 연결된 배포 그룹 레코드와 배포 레코드가 모두 삭제됩니다.

```
aws deploy delete-application --application-name HelloWorld_App
```

4. Systems Manager 상태 관리자 연결을 삭제하려면 delete-association 명령을 호출합니다.

```
aws ssm delete-association --assocation-id association-id
```

describe-association 명령을 호출하여 *association-id*를 얻을 수 있습니다.

```
aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
 Key=tag:Name,Values=CodeDeployDemo
```

5. 이 자습서에서 CloudFormation 스택을 사용하지 않은 경우 terminate-instances 명령을 호출하여 수동으로 생성한 Amazon EC2 인스턴스를 종료합니다. 종료할 Amazon EC2 인스턴스의 ID를 입력합니다.

```
aws ec2 terminate-instances --instance-ids instanceId
```

리소스를 정리하려면(콘솔)

이 자습서에서 CloudFormation 템플릿을 사용한 경우 연결된 CloudFormation 스택을 삭제합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.
2. 검색 상자에 CloudFormation 스택 이름(예: **CodeDeployDemoStack**)을 입력합니다.
3. 스택 이름 옆의 상자를 선택합니다.
4. 작업 메뉴에서 스택 삭제를 선택합니다. 이렇게 하면 스택이 삭제되고, 수반되는 모든 Amazon EC2 인스턴스가 종료되며, 수반되는 모든 IAM 역할이 삭제됩니다.

CloudFormation 스택 외부에서 생성한 Amazon EC2 인스턴스를 종료하려면:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 인스턴스 영역에서 인스턴스를 선택합니다.
3. 검색 상자에 종료할 Amazon EC2 인스턴스의 이름을 입력한 후 Enter를 누릅니다.
4. Amazon EC2 인스턴스를 선택합니다.
5. 작업을 선택하고 인스턴스 상태를 가리킨 다음 종료를 선택합니다. 메시지가 나타나면 [Yes, Terminate]를 선택합니다. 추가 Amazon EC2 인스턴스에 대해 이 단계를 반복합니다.

Amazon S3 버킷을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 버킷 목록에서 Amazon S3 버킷의 이름(예: **amzn-s3-demo-bucket**)을 찾아서 선택합니다.
3. 버킷을 삭제하려면 먼저 버킷의 콘텐츠를 삭제해야 합니다. 버킷에 있는 모든 파일(예: **HelloWorld_App.zip**)을 선택합니다. 작업 메뉴에서 삭제를 선택합니다. 삭제 확인 메시지가 표시되면 확인을 선택합니다.
4. 버킷을 비운 후 버킷을 삭제할 수 있습니다. 버킷 목록에서 버킷 행을 선택합니다(버킷 이름은 아님). 버킷 삭제를 선택하고 확인하라는 메시지가 나타나면확인을 선택합니다.

CodeDeploy에서 `HelloWorld_App` 애플리케이션을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. **HelloWorld_App**을 선택합니다.
4. 애플리케이션 삭제를 선택합니다.
5. 메시지가 표시되면 **Delete**를 입력한 후 삭제를 선택합니다.

Systems Manager 상태 관리자 연결을 삭제하려면

1. https://console.aws.amazon.com/systems-manager AWS Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 상태 관리자를 선택합니다.
3. 생성한 연결을 선택하고 삭제를 선택합니다.

다음 단계

여기까지 진행했으면 CodeDeploy를 사용하여 배포를 성공적으로 완료한 것입니다. 축하합니다!

튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)

이 튜토리얼에서는 Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux(RHEL)를 실행하는 단일 온프레미스 인스턴스(Amazon EC2 온프레미스 인스턴스가 아닌 물리적 디바이스)에 샘플 애플리케이션 수정 버전을 배포하는 과정을 안내하여 CodeDeploy를 경험해 볼 수 있습니다. 온프레미스 인스턴스와 이러한 인스턴스가 CodeDeploy와 작동하는 방식에 대한 자세한 내용은 Working with On-Premises Instances 단원을 참조하세요.

찾는 항목이 보이지 않습니까?

- Amazon Linux 또는 RHEL을 실행하는 Amazon EC2 인스턴스에 배포하려면 튜토리얼: Amazon EC2 인스턴스(Amazon Linux 또는 Red Hat Enterprise Linux 및 Linux, macOS 또는 Unix)에 WordPress 배포 단원을 참조하세요.
- Windows Server를 실행하는 Amazon EC2 인스턴스에 배포하려면 튜토리얼: CodeDeploy를 사용하여 "hello, world!" 애플리케이션 배포(Windows Server) 단원을 참조하세요.

주제

- 사전 조건
- 1단계: 온프레미스 인스턴스 구성
- 2단계: 샘플 애플리케이션 수정 버전 만들기
- 3단계: 애플리케이션 수정 버전을 번들링하여 Amazon S3에 업로드
- 4단계: 애플리케이션 수정 버전 배포
- 5단계: 배포 확인

- 6단계: 리소스 정리

사전 조건

이 자습서를 시작하기 전에 사용자 구성, 설치 또는 업그레이드, 서비스 역할 AWS CLI생성 CodeDeploy 시작하기등의 사전 조건을 완료해야 합니다. 사전 요구 사항에서 설명하는 것처럼 IAM 인스턴스 프로파일을 만들 필요는 없습니다. 온프레미스 인스턴스에서는 IAM 인스턴스 프로파일을 사용하지 않습니다.

온프레미스 인스턴스로 구성할 물리적 디바이스에서는 CodeDeploy 에이전트에서 지원하는 운영 체제에 나열된 운영 체제 중 하나를 실행해야 합니다.

1단계: 온프레미스 인스턴스 구성

온프레미스 인스턴스에 배포하기 전에 먼저, 온프레미스 인스턴스를 구성해야 합니다. Working with On-Premises Instances의 지침을 수행한 다음 이 페이지로 돌아오세요.

CodeDeploy 에이전트 설치

온프레미스 인스턴스를 구성한 후 CodeDeploy 에이전트 설치의 온프레미스 인스턴스 관련 단계를 수행하고 이 페이지로 돌아갑니다.

2단계: 샘플 애플리케이션 수정 버전 만들기

이 단계에서는 온프레미스 인스턴스에 배포할 샘플 애플리케이션 수정을 만듭니다.

온프레미스 인스턴스에 어떤 소프트웨어 및 기능이 이미 설치되어 있는지 또는 조직의 정책에 따라 어떤 소프트웨어 및 기능을 설치할 수 있는지 파악하기 어렵기 때문에 여기서 제공되는 샘플 애플리케이션 수정 버전은 배치 스크립트(Windows Server용) 또는 셸 스크립트(Ubuntu Server 및 RHEL용)를 사용하여 온프레미스 인스턴스의 위치에 텍스트 파일을 씁니다. Install, AfterInstall, ApplicationStart, ValidateService 등 여러 CodeDeploy의 배포 수명 주기 이벤트 각각에 대해 하나의 파일이 작성됩니다. BeforeInstall 배포 수명 주기 이벤트 중 스크립트가 실행되어 이 샘플의 이전 배포 중 기록된 이전 파일을 제거하고 온프레미스 인스턴스에 새 파일을 기록할 위치를 만듭니다.

> **Note**
>
> 다음 중 하나에 해당하면 샘플 애플리케이션 수정을 배포하지 못할 수 있습니다.
>
> - 온프레미스 인스턴스에서 CodeDeploy 에이전트를 시작한 사용자에게 스크립트를 실행할 권한이 없는 경우

- 사용자에게 스크립트에 나열된 위치에 폴더를 만들거나 삭제할 권한이 없는 경우
- 사용자에게 스크립트에 나열된 위치에 텍스트 파일을 만들 권한이 없는 경우

> **Note**
>
> Windows Server 인스턴스를 구성했는데 다른 샘플을 배포하고자 하는 경우 튜토리얼: CodeDeploy를 사용하여 "hello, world!" 애플리케이션 배포(Windows Server) 튜토리얼의 2단계: Windows Server Amazon EC2 인스턴스에 배포하도록 원본 콘텐츠 구성에 나와 있는 샘플을 사용할 수 있습니다.
> RHEL 인스턴스를 구성했는데 다른 샘플을 배포하고자 하는 경우 튜토리얼: Amazon EC2 인스턴스(Amazon Linux 또는 Red Hat Enterprise Linux 및 Linux, macOS 또는 Unix)에 WordPress 배포 튜토리얼의 2단계: Amazon Linux 또는 Red Hat Enterprise Linux Amazon EC2 인스턴스에 배포할 원본 콘텐츠 구성에 나와 있는 샘플을 사용할 수 있습니다.
> 현재, Ubuntu Server에 사용할 수 있는 대체 샘플은 없습니다.

1. 개발 머신에 샘플 애플리케이션 수정의 파일을 저장할 `CodeDeployDemo-OnPrem`이라는 하위 디렉터리(하위 폴더)를 만든 다음 이 폴더로 이동합니다. 예를 들어 Windows Server에서는 `c:\temp` 폴더, Ubuntu Server 및 RHEL에서는 `/tmp` 폴더를 루트 폴더로 사용한다고 가정합니다. 다른 폴더를 사용하는 경우 이 자습서 전체에서 해당 폴더로 바꾸십시오.

   Windows의 경우:

   ```
   mkdir c:\temp\CodeDeployDemo-OnPrem
   cd c:\temp\CodeDeployDemo-OnPrem
   ```

   Linux, macOS, Unix의 경우:

   ```
   mkdir /tmp/CodeDeployDemo-OnPrem
   cd /tmp/CodeDeployDemo-OnPrem
   ```

2. `CodeDeployDemo-OnPrem` 하위 폴더의 루트에서 텍스트 편집기를 사용하여 `appspec.yml`이라는 파일과 `install.txt`라는 파일을 만듭니다.

   Windows Server의 경우 `appspec.yml`:

   ```
   version: 0.0
   ```

```
os: windows
files:
  - source: .\install.txt
    destination: c:\temp\CodeDeployExample
hooks:
  BeforeInstall:
    - location: .\scripts\before-install.bat
      timeout: 900
  AfterInstall:
    - location: .\scripts\after-install.bat
      timeout: 900
  ApplicationStart:
    - location: .\scripts\application-start.bat
      timeout: 900
  ValidateService:
    - location: .\scripts\validate-service.bat
      timeout: 900
```

Ubuntu Server and RHEL의 경우 `appspec.yml`:

```
version: 0.0
os: linux
files:
  - source: ./install.txt
    destination: /tmp/CodeDeployExample
hooks:
  BeforeInstall:
    - location: ./scripts/before-install.sh
      timeout: 900
  AfterInstall:
    - location: ./scripts/after-install.sh
      timeout: 900
  ApplicationStart:
    - location: ./scripts/application-start.sh
      timeout: 900
  ValidateService:
    - location: ./scripts/validate-service.sh
      timeout: 900
```

AppSpec 파일에 대한 자세한 내용은 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 및 CodeDeploy AppSpec 파일 참조를 참조하세요.

`install.txt`:

```
The Install deployment lifecycle event successfully completed.
```

3. `CodeDeployDemo-OnPrem` 하위 폴더의 루트 아래에 `scripts` 하위 폴더를 만든 후 해당 하위 폴더로 이동합니다.

   Windows의 경우:

   ```
   mkdir c:\temp\CodeDeployDemo-OnPrem\scripts
   cd c:\temp\CodeDeployDemo-OnPrem\scripts
   ```

   Linux, macOS, Unix의 경우:

   ```
   mkdir -p /tmp/CodeDeployDemo-OnPrem/scripts
   cd /tmp/CodeDeployDemo-OnPrem/scripts
   ```

4. `scripts` 하위 폴더의 루트에서 텍스트 편집기를 사용하여 Windows Server의 경우 `before-install.bat`, `after-install.bat`, `application-start.bat`, `validate-service.bat`(이)라는 파일 4개를 작성하거나 Ubuntu Server 및 RHEL의 경우 `before-install.sh`, `after-install.sh`, `application-start.sh`, `validate-service.sh`(이)라는 파일 4개를 작성합니다.

   Windows Server의 경우:

   `before-install.bat`:

   ```
   set FOLDER=%HOMEDRIVE%\temp\CodeDeployExample

   if exist %FOLDER% (
     rd /s /q "%FOLDER%"
   )

   mkdir %FOLDER%
   ```

   `after-install.bat`:

   ```
   cd %HOMEDRIVE%\temp\CodeDeployExample
   ```

```
echo The AfterInstall deployment lifecycle event successfully completed. > after-
install.txt
```

`application-start.bat`:

```
cd %HOMEDRIVE%\temp\CodeDeployExample

echo The ApplicationStart deployment lifecycle event successfully completed. >
 application-start.txt
```

`validate-service.bat`:

```
cd %HOMEDRIVE%\temp\CodeDeployExample

echo The ValidateService deployment lifecycle event successfully completed. >
 validate-service.txt
```

Ubuntu Server and RHEL의 경우 :

`before-install.sh`:

```
#!/bin/bash
export FOLDER=/tmp/CodeDeployExample

if [ -d $FOLDER ]
then
 rm -rf $FOLDER
fi

mkdir -p $FOLDER
```

`after-install.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The AfterInstall deployment lifecycle event successfully completed." > after-
install.txt
```

`application-start.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ApplicationStart deployment lifecycle event successfully completed." >
 application-start.txt
```

`validate-service.sh`:

```
#!/bin/bash
cd /tmp/CodeDeployExample

echo "The ValidateService deployment lifecycle event successfully completed." >
 validate-service.txt

unset FOLDER
```

5. Ubuntu Server 및 RHEL에서만 네 개의 셸 스크립트에 실행 권한이 있는지 확인하세요.

```
chmod +x ./scripts/*
```

3단계: 애플리케이션 수정 버전을 번들링하여 Amazon S3에 업로드

애플리케이션 수정 버전을 배포하려면 파일을 번들링한 다음 파일 번들을 Amazon S3 버킷에 업로드해야 합니다. CodeDeploy를 사용하여 애플리케이션 생성 및 Amazon S3에 CodeDeploy의 개정 푸시(EC2 온프레미스 배포 전용)의 지침을 따르십시오. 애플리케이션과 배포 그룹에 어떤 이름이든 지정할 수 있지만 애플리케이션 이름으로 `CodeDeploy-OnPrem-App`을, 배포 그룹 이름으로 `CodeDeploy-OnPrem-DG`를 지정하는 것이 좋습니다. 이러한 지침을 완료한 후 이 페이지로 돌아옵니다.

> **Note**
>
> 또는 GitHub 리포지토리에 파일 번들을 업로드하고 GitHub 리포지토리에서 배포할 수 있습니다. 자세한 내용은 GitHub와 CodeDeploy 통합 단원을 참조하십시오.

4단계: 애플리케이션 수정 버전 배포

Amazon S3 버킷에 애플리케이션 수정 버전을 업로드한 후 온프레미스 인스턴스에 배포해 보십시오. CodeDeploy에서 배포 만들기의 지침을 수행한 다음 이 페이지로 돌아오세요.

5단계: 배포 확인

배포가 성공적인지 확인하려면 CodeDeploy 배포 세부 정보 보기의 지침을 수행해고 이 페이지로 돌아옵니다.

배포에 성공한 경우 `c:\temp\CodeDeployExample` 폴더(Windows Server의 경우) 또는 `/tmp/CodeDeployExample` (Ubuntu Server 및 RHEL의 경우)에서 텍스트 파일 네 개를 찾을 수 있습니다.

배포에 실패한 경우에는 View Instance Details 및 인스턴스 문제 해결의 문제 해결 단계를 수행하세요. 필요한 부분을 수정하고, 애플리케이션 수정을 다시 번들링하여 업로드한 다음 다시 배포해 보십시오.

6단계: 리소스 정리

이 튜토리얼에서 만든 리소스에 대해 비용이 청구되지 않도록 하려면 더 이상 사용할 필요가 없는 경우 Amazon S3 버킷을 삭제합니다. 또한 CodeDeploy 및 온프레미스 인스턴스의 애플리케이션 및 배포 그룹 레코드 등과 같은 관련 리소스를 정리할 수도 있습니다.

AWS CLI 또는 CodeDeploy 및 Amazon S3 콘솔과의 조합을 사용하여 리소스를 정리 AWS CLI 할 수 있습니다.

리소스 정리(CLI)

Amazon S3 버킷을 삭제하려면

- 버킷(예: `amzn-s3-demo-bucket`)에 대한 `--recursive` 스위치와 함께 rm 명령을 호출합니다. 버킷을 비롯해 버킷에 있는 모든 객체가 삭제됩니다.

```
aws s3 rm s3://your-bucket-name --recursive --region region
```

CodeDeploy에서 배포 그룹 레코드와 애플리케이션을 삭제하려면

- 애플리케이션(예: `CodeDeploy-OnPrem-App`)에 대해 delete-application 명령을 호출합니다. 배포 그룹 및 배포에 대한 레코드가 삭제됩니다.

```
aws deploy delete-application --application-name your-application-name
```

온프레미스 인스턴스의 등록을 취소하고 IAM 사용자를 삭제하려면

- 온프레미스 인스턴스 및 리전에 대해 deregister 명령을 호출합니다.

```
aws deploy deregister --instance-name your-instance-name --delete-iam-user --
region your-region
```

> **Note**
>
> 이 온프레미스 인스턴스와 연결된 IAM 사용자를 삭제하지 않으려는 경우 `--no-delete-iam-user` 옵션을 대신 사용합니다.

CodeDeploy 에이전트를 제거하고 온프레미스 인스턴스에서 구성 파일을 삭제하려면

- 온프레미스 인스턴스에서 uninstall 명령을 호출합니다.

```
aws deploy uninstall
```

이제 자습서에 사용된 리소스를 정리하는 모든 단계를 완료했습니다.

리소스 정리(콘솔)

Amazon S3 버킷을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.
2. 삭제하려는 버킷(예: `amzn-s3-demo-bucket`) 옆의 아이콘을 선택하고 버킷은 선택하지 않습니다.
3. 작업을 선택한 후 삭제를 선택합니다.
4. 버킷을 삭제할 것인지 묻는 메시지가 나타나면 [OK]를 선택합니다.

CodeDeploy에서 배포 그룹 레코드와 애플리케이션을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 애플리케이션을 선택합니다.
3. 삭제할 애플리케이션 이름(예: `CodeDeploy-OnPrem-App`)을 선택한 후 애플리케이션 삭제를 선택합니다.
4. 메시지가 표시되면 삭제를 확인할 애플리케이션의 이름을 입력한 후 삭제를 선택합니다.

AWS CodeDeploy 콘솔을 사용하여 온프레미스 인스턴스의 등록을 취소하거나 CodeDeploy 에이전트를 제거할 수 없습니다. 온프레미스 인스턴스의 등록을 취소하고 IAM 사용자를 삭제하려면의 지침을 따르세요.

튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포

이 튜토리얼에서는 CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 수정 버전을 배포합니다. Amazon EC2 Auto Scaling은 미리 정의된 조건을 사용하여 Amazon EC2 인스턴스를 시작한 다음 더 이상 필요하지 않을 때 해당 인스턴스를 종료합니다. Amazon EC2 Auto Scaling은 배포 로드를 처리할 수 있는 정확한 수의 Amazon EC2 인스턴스를 유지하도록 보장함으로써 CodeDeploy를 확장하는 데 도움이 됩니다. CodeDeploy와 Amazon EC2 Auto Scaling 통합에 대한 자세한 내용은 Amazon EC2 Auto Scaling과 CodeDeploy 통합 섹션을 참조하세요.

주제

- 사전 조건
- 1단계: Auto Scaling 그룹 생성 및 구성
- 2단계: Auto Scaling 그룹에 애플리케이션 배포
- 3단계: 결과 확인
- 4단계: Auto Scaling 그룹에서 Amazon EC2 인스턴스의 수 늘리기
- 5단계: 결과 다시 확인
- 6단계: 정리

사전 조건

이 자습서를 따르려면 다음을 수행하세요.

- 설정 및 구성CodeDeploy 시작하기, IAM 인스턴스 프로파일(**CodeDeployDemo-EC2-Instance-Profile**) 및 서비스 역할() AWS CLI 생성을 포함하여의 모든 단계를 완료합니다**CodeDeployDemo**. 서비스 역할은 사용자를 대신해 수행할 수 있는 서비스 권한을 부여하는 특별한 유형의 IAM 역할입니다.
- 시작 템플릿으로 Auto Scaling 그룹을 생성한 경우 다음 권한을 추가해야 합니다.
  - `ec2:RunInstances`
  - `ec2:CreateTags`
  - `iam:PassRole`

  자세한 내용은 2단계: 서비스 역할 생성, Auto Scaling 그룹을 위한 시작 템플릿 만들기 및 Amazon EC2 Auto Scaling 사용 설명서.의 시작 템플릿 지원을 참조하세요.
- Ubuntu Server 인스턴스 및 CodeDeploy와 호환되는 수정 버전을 생성하고 사용합니다. 수정에 대해 다음 중 하나를 수행할 수 있습니다.
  - 2단계: 샘플 애플리케이션 수정 버전 만들기 튜토리얼에서 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)의 샘플 수정 버전을 생성하고 사용합니다.
  - 수정을 직접 만들려면 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.
- 다음 인바운드 규칙을 사용하여 **CodeDeployDemo-AS-SG**라는 보안 그룹을 만듭니다.
  - 유형: HTTP
  - 출처: 모든 소스

  이 작업은 애플리케이션을 보고 배포 성공 여부를 확인하는 데 필요합니다. 보안 그룹을 생성하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 보안 그룹 생성을 참조하세요.

1단계: Auto Scaling 그룹 생성 및 구성

이 단계에서는 단일 Amazon Linux, RHEL 또는 Windows Server Amazon EC2 인스턴스를 포함하는 Auto Scaling 그룹을 생성합니다. 이후 단계에서는 Amazon EC2 인스턴스를 하나 더 추가하도록 Amazon EC2 Auto Scaling에 지시하고 CodeDeploy에서는 수정 버전을 새로 추가된 인스턴스에 배포합니다.

주제

- Auto Scaling 그룹을 생성하고 구성하려면(CLI 사용)
- Auto Scaling 그룹을 생성하고 구성하려면(콘솔 사용)

Auto Scaling 그룹을 생성하고 구성하려면(CLI 사용)

1. create-launch-template 명령을 호출하여 Amazon EC2 시작 템플릿을 생성합니다.

   이 명령을 호출하기 전에 이 튜토리얼에서 사용할 수 있는 AMI의 ID가 필요합니다(자리 표시자 *image-id*로 표시됨). 또한 Amazon EC2 인스턴스에 액세스할 수 있도록 Amazon EC2 인스턴스 키 페어의 이름도 필요합니다(자리 표시자 *key-name*으로 표시됨).

   이 튜토리얼에서 사용할 수 있는 AMI의 ID를 가져오려면:

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

   b. 탐색 창의 [Instances]에서 [Instances]를 선택한 후 [Launch Instance]를 선택합니다.

   c. Amazon Machine Image 선택(Choose an Amazon Machine Image) 페이지의 빠른 시작(Quick Start) 탭에서 Amazon Linux 2 AMI, Red Hat Enterprise Linux 7.1, Ubuntu Server 14.04 LTS 또는 Microsoft Windows Server 2012 R2 옆에 있는 AMI의 ID를 기록해 둡니다.

   > **Note**
   >
   > CodeDeploy와 호환되는 사용자 지정 AMI 버전이 있는 경우 빠른 시작 탭에서 찾는 대신 여기에서 해당 버전을 선택하세요. CodeDeploy 및 Amazon EC2 Auto Scaling에서 사용자 지정 AMI 사용에 대한 자세한 내용은 CodeDeploy 및 Amazon EC2 Auto Scaling에서 사용자 지정 AMI 사용 단원을 참조하세요.

   Amazon EC2 인스턴스 키 페어의 경우 Amazon EC2 인스턴스 키 페어의 이름을 사용합니다.

   create-launch-template 명령을 호출합니다.

   로컬 Linux, macOS 또는 Unix 머신의 경우

   ```
   aws ec2 create-launch-template \
     --launch-template-name CodeDeployDemo-AS-Launch-Template \
     --launch-template-data file://config.json
   ```

`config.json` 파일 내용은 다음과 같습니다.

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

로컬 Windows 머신의 경우

```
aws ec2 create-launch-template --launch-template-name CodeDeployDemo-AS-Launch-
Template --launch-template-data file://config.json
```

`config.json` 파일 내용은 다음과 같습니다.

```
{
  "InstanceType":"t1.micro",
  "ImageId":"image-id",
  "IamInstanceProfile":{
    "Name":"CodeDeployDemo-EC2-Instance-Profile"
  },
  "KeyName":"key-name"
}
```

`config.json` 파일과 함께 이러한 명령을 사용하면 다음 단계에서 t1.micro Amazon EC2 인스턴스 유형에 따라 생성되는 Auto Scaling 그룹에 대한 CodeDeployDemo-AS-Launch-Template이라는 Amazon EC2 시작 템플릿이 생성됩니다. `ImageId`, `IamInstanceProfile` 및 `KeyName`에 대한 입력 사항에 따라 시작 템플릿이 AMI ID, 시작 시 인스턴스에 전달할 IAM 역할과 관련된 인스턴스 프로파일의 이름 및 인스턴스에 연결할 때 사용할 Amazon EC2 키 페어도 지정합니다.

2. create-auto-scaling-group 명령을 호출하여 Auto Scaling 그룹을 생성합니다. 리전에 나열된 리전 중 하나에 있는 가용 영역 중 하나의 이름과 자리 표시자 `##  ##`으로 AWS 일반 참조표시되는의 엔드포인트가 필요합니다.

> **Note**
>
> 리전의 가용 영역 목록을 보려면 다음을 호출합니다.

```
aws ec2 describe-availability-zones --region region-name
```

예를 들어, 미국 서부(오레곤) 리전의 가용 영역 목록을 보려면 다음을 호출합니다.

```
aws ec2 describe-availability-zones --region us-west-2
```

리전 이름 식별자 목록은 섹션을 참조하세요리전별 리소스 키트 버킷 이름.

로컬 Linux, macOS 또는 Unix 머신의 경우

```
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name CodeDeployDemo-AS-Group \
  --launch-template CodeDeployDemo-AS-Launch-Template,Version='$Latest' \
  --min-size 1 \
  --max-size 1 \
  --desired-capacity 1 \
  --availability-zones availability-zone \
  --tags Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

로컬 Windows 머신의 경우

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name
 CodeDeployDemo-AS-Group --launch-template LaunchTemplateName=CodeDeployDemo-
AS-Launch-Template,Version="$Latest" --min-size 1 --max-size 1 --
desired-capacity 1 --availability-zones availability-zone --tags
 Key=Name,Value=CodeDeployDemo,PropagateAtLaunch=true
```

이 명령은 **CodeDeployDemo-AS-Launch-Template**이라는 Amazon EC2 시작 템플릿을 기반으로 **CodeDeployDemo-AS-Group**이라는 Auto Scaling 그룹을 생성합니다. 이 Auto Scaling 그룹에는 Amazon EC2 인스턴스 하나만 있으며, 이 인스턴스는 지정된 가용 영역에 생성됩니다. 이 Auto Scaling 그룹의 각 인스턴스에는 Name=CodeDeployDemo 태그가 있습니다. 이 태그는 나중에 CodeDeploy 에이전트를 설치할 때 사용됩니다.

3. **CodeDeployDemo-AS-Group**에 대해 describe-auto-scaling-groups 명령을 호출합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
 CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus,
 LifecycleState]" --output text
```

반환된 값에 `Healthy` 및 `InService`가 표시될 때까지 진행하지 마십시오.

4. Auto Scaling 그룹의 인스턴스에 CodeDeploy 배포에 사용할 CodeDeploy 에이전트가 설치되어 있어야 합니다. Auto Scaling 그룹이 생성될 때 추가된 태그를 AWS Systems Manager 사용하여에서 create-association 명령을 호출하여 CodeDeploy 에이전트를 설치합니다.

```
aws ssm create-association \
  --name AWS-ConfigureAWSPackage \
  --targets Key=tag:Name,Values=CodeDeployDemo \

 --parameters action=Install, name=AWSCodeDeployAgent \
  --schedule-expression "cron(0 2 ? * SUN *)"
```

이 명령은 Systems Manager 상태 관리자에 연결을 생성하여 Auto Scaling 그룹의 모든 인스턴스에 CodeDeploy 에이전트를 설치하고 매주 일요일 오전 2시에 업데이트를 시도합니다. CodeDeploy 에이전트에 대한 자세한 내용은 CodeDeploy 에이전트 작업을 참조하세요. Systems Manager에 대한 자세한 내용은 AWS Systems Manager란 무엇입니까를 참조하세요.

Auto Scaling 그룹을 생성하고 구성하려면(콘솔 사용)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 글로벌 탐색 모음에서 리전에 나열된 리전 중 하나와의 엔드포인트가 선택AWS 일반 참조되어 있는지 확인합니다. Amazon EC2 Auto Scaling 리소스는 지정한 리전에 연결되어 있으며 CodeDeploy는 일부 리전에서만 지원됩니다.
3. 탐색 창의 인스턴스에서 템플릿 시작을 선택합니다.
4. Create launch template(시작 템플릿 생성)을 선택합니다.
5. 시작 템플릿 이름 및 설명의 시작 템플릿 이름에 **`CodeDeployDemo-AS-Launch-Template`**을(를) 입력합니다. 다른 필드의 경우 기본값을 그대로 유지합니다.
6. Amazon Machine Image(AMI) 대화 상자에서 AMI 아래 드롭다운을 클릭하여 이 튜토리얼에서 사용할 수 있는 AMI를 선택합니다.

- AMI 드롭다운의 빠른 시작(Quick Start) 탭에서 Amazon Linux 2 AMI, Red Hat Enterprise Linux 7.1, Ubuntu Server 14.04 LTS 또는 Microsoft Windows Server 2012 R2 중 하나를 선택합니다.

> **Note**
>
> CodeDeploy와 호환되는 사용자 지정 AMI 버전이 있는 경우 빠른 시작 탭에서 찾는 대신 여기에서 해당 버전을 선택하세요. CodeDeploy 및 Amazon EC2 Auto Scaling에서 사용자 지정 AMI 사용에 대한 자세한 내용은 CodeDeploy 및 Amazon EC2 Auto Scaling에서 사용자 지정 AMI 사용 단원을 참조하세요.

7. 인스턴스 유형(Instance type)에서 드롭다운을 선택한 다음 t1.micro를 선택합니다. 검색 창을 사용하면 더 빨리 찾을 수 있습니다.
8. 키 페어(로그인)(Key pair(login)) 대화 상자에서 기존 키 페어 선택(Choose an existing key pair)을 선택합니다. 키 페어 선택 드롭다운 목록에서 이전 단계에서 만들었거나 사용한 Amazon EC2 인스턴스 키 페어를 선택합니다.
9. 네트워크 설정(Network settings) 대화 상자에서 가상 퍼블릭 클라우드(VPC)(Virtual Public Cloud (VPC))를 선택합니다.

   보안 그룹(Security groups) 드롭다운에서 튜토리얼의 사전 요구 사항 섹션(**CodeDeployDemo-AS-SG**)에서 생성한 보안 그룹을 선택합니다.
10. 고급 세부 정보 대화 상자를 펼칩니다. IAM 인스턴스 프로파일(IAM instance profile) 드롭다운에서 이전에 IAM 인스턴스 프로파일(IAM instance profile) 아래에서 생성한 IAM 역할(**CodeDeployDemo-EC2-Instance-Profile**)을 선택합니다.

    나머지 기본값을 그대로 둡니다.
11. Create launch template(시작 템플릿 생성)을 선택합니다.
12. 다음 단계(Next steps) 대화 상자에서 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.
13. 시작 템플릿 또는 구성 선택 페이지에서 Auto Scaling 그룹 이름에 **CodeDeployDemo-AS-Group**을(를) 입력합니다.
14. 시작 템플릿(Launch template) 대화 상자에서 시작 템플릿(**CodeDeployDemo-AS-Launch-Template**)을 작성해야 합니다. 작성하지 않고 드롭다운 메뉴에서 선택해도 됩니다. 기본값을 그대로 두고 다음을 선택합니다.

15. 인스턴스 시작 옵션 선택(Choose instance launch options) 페이지의 네트워크(Network) 섹션에서 VPC에 대해 기본 VPC를 선택합니다. 그런 다음 가용 영역 및 서브넷(Availability Zones and subnets)에 대해 기본 서브넷을 선택합니다. 기본값을 선택할 수 없는 경우 VPC를 생성해야 합니다. 자세한 내용은 Amazon VPC 시작하기를 참조하세요.
16. 인스턴스 유형 요구 사항(Instance type requirements) 섹션에서 기본 설정을 사용하여 이 단계를 단순화합니다. (시작 템플릿을 재정의하지 마세요.) 이 자습서에서는 시작 템플릿에 지정된 인스턴스 유형을 사용하여 온디맨드 인스턴스만 시작합니다.
17. 다음(Next)을 선택하고 고급 옵션 구성(Configure advanced options) 페이지로 이동합니다.
18. 기본값을 유지하고 다음을 선택합니다.
19. 그룹 크기 및 조정 정책 구성(Configure group size and scaling policies) 페이지에서 기본 그룹 크기(Group size) 값 1을 유지합니다. 다음을 선택합니다.
20. 알림을 구성하는 단계를 건너뛰고 다음을 선택합니다.
21. 태그 추가(Add tags) 페이지에서 나중에 CodeDeploy 에이전트를 설치할 때 사용할 태그를 추가합니다. 태그 추가를 선택합니다.

    a. 키에 **Name**을(를) 입력합니다.

    b. 값에 **CodeDeployDemo**을(를) 입력합니다.

    다음을 선택합니다.
22. 검토(Review) 페이지에서 Auto Scaling 그룹 정보를 검토한 다음 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.
23. 탐색 모음에서 Auto Scaling 그룹이 선택된 상태에서 **CodeDeployDemo-AS-Group**을(를) 선택한 후 인스턴스 관리 탭을 선택합니다. 수명 주기 열에 값 InService가 나타나고, 상태 열에 값 정상이 나타날 때까지 진행하지 마세요.
24. CodeDeploy 에이전트 설치의 단계에 따르고 `Name=CodeDeployDemo` 인스턴스 태그를 사용하여 CodeDeploy 에이전트를 설치합니다.

2단계: Auto Scaling 그룹에 애플리케이션 배포

이 단계에서는 Auto Scaling 그룹의 단일 Amazon EC2 인스턴스에 수정 버전을 배포합니다.

주제

- 배포를 만들려면(CLI)
- 배포를 만들려면(콘솔)

배포를 만들려면(CLI)

1. create-application 명령을 호출하여 **SimpleDemoApp**이라는 애플리케이션을 생성합니다.

```
aws deploy create-application --application-name SimpleDemoApp
```

2. 2단계: CodeDeploy에 대한 서비스 역할 생성의 지침에 따라 이미 서비스 역할을 만들었을 것입니다. 서비스 역할은 CodeDeploy에 Amazon EC2 인스턴스에 액세스하여 태그를 확장할(읽을) 수 있는 권한을 부여합니다. 서비스 역할 ARN이 필요합니다. 서비스 역할 ARN을 확인하려면 서비스 역할 ARN 확인(CLI)의 지침을 따르십시오.
3. 이제 서비스 역할 ARN이 있으므로 지정된 서비스 역할 ARN으로 create-deployment-group 명령을 호출하고 **CodeDeployDemo-AS-Group**이라는 Auto Scaling 그룹과 **CodeDeployDefault.OneAtATime**이라는 배포 구성을 사용하여 **SimpleDemoDG**라는 배포 그룹을 생성하고 **SimpleDemoApp**이라는 애플리케이션에 연결합니다.

> **Note**
>
> create-deployment-group 명령은 배포 및 인스턴스의 지정된 이벤트에 대해 주제의 구독자에게 Amazon SNS 알림을 전송하게 하는 트리거를 생성합니다. 또한 해당 명령은 Amazon CloudWatch 경보 모니터링 임계값에 도달한 경우 자동으로 배포를 롤백하고 배포를 중지하도록 경보를 설정하는 옵션을 지원합니다. 이 작업에 대한 명령은 이 자습서에 포함되지 않습니다.

로컬 Linux, macOS 또는 Unix 머신의 경우

```
aws deploy create-deployment-group \
  --application-name SimpleDemoApp \
  --auto-scaling-groups CodeDeployDemo-AS-Group \
  --deployment-group-name SimpleDemoDG \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --service-role-arn service-role-arn
```

로컬 Windows 머신의 경우

```
aws deploy create-deployment-group --application-name SimpleDemoApp --auto-scaling-
groups CodeDeployDemo-AS-Group --deployment-group-name SimpleDemoDG --deployment-
config-name CodeDeployDefault.OneAtATime --service-role-arn service-role-arn
```

4. create-deployment 명령을 호출하고 지정된 위치의 수정을 사용하여 **SimpleDemoApp**이라는 애플리케이션, **CodeDeployDefault.OneAtATime**이라는 배포 구성 및 **SimpleDemoDG**라는 배포 그룹과 연결된 배포를 만듭니다.

   Amazon Linux 및 RHEL Amazon EC2 인스턴스의 경우 로컬 Linux, macOS 또는 Unix 시스템에서 호출

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
  --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/
SampleApp_Linux.zip
```

   *bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름입니다. 예를 들어 미국 동부(오하이오) 리전의 경우 *bucket-name*을 aws-codedeploy-us-east-2로 바꿉니다. 버킷 이름 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

   Amazon Linux 및 RHEL Amazon EC2 인스턴스의 경우 로컬 Windows 시스템에서 호출

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
location bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Linux.zip
```

   *bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름입니다. 예를 들어 미국 동부(오하이오) 리전의 경우 *bucket-name*을 aws-codedeploy-us-east-2로 바꿉니다. 버킷 이름 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

   Windows Server Amazon EC2 인스턴스의 경우 로컬 Linux, macOS 또는 Unix 시스템에서 호출

```
aws deploy create-deployment \
  --application-name SimpleDemoApp \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name SimpleDemoDG \
```

```
  --s3-location bucket=bucket-name,bundleType=zip,key=samples/latest/
SampleApp_Windows.zip
```

*bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름입니다. 예를 들어 미국 동부(오하이오) 리전의 경우 *bucket-name*을 aws-codedeploy-us-east-2로 바꿉니다. 버킷 이름 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

Windows Server Amazon EC2 인스턴스의 경우 로컬 Windows 시스템에서 호출

```
aws deploy create-deployment --application-name SimpleDemoApp --deployment-config-
name CodeDeployDefault.OneAtATime --deployment-group-name SimpleDemoDG --s3-
location bucket=bucket-name,bundleType=zip,key=samples/latest/SampleApp_Windows.zip
```

*bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름입니다. 예를 들어 미국 동부(오하이오) 리전의 경우 *bucket-name*을 aws-codedeploy-us-east-2로 바꿉니다. 버킷 이름 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

> **Note**
>
> 현재 CodeDeploy 는 Ubuntu Server Amazon EC2 인스턴스를 배포하기 위한 샘플 수정 버전을 제공하지 않습니다. 수정을 직접 만들려면 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.

5. get-deployment 명령을 호출하여 배포에 성공했는지 확인합니다.

   이 명령을 호출하기 전에 배포 ID가 필요합니다. 이 ID는 create-deployment 명령에 대한 호출로 반환되어야 합니다. 배포 ID를 다시 가져와야 할 경우 list-deployments이라는 애플리케이션 및 **SimpleDemoApp**라는 배포 그룹에 대해 **SimpleDemoDG** 명령을 호출하세요.

   ```
   aws deploy list-deployments --application-name SimpleDemoApp --deployment-group-
   name SimpleDemoDG --query "deployments" --output text
   ```

   이제 배포 ID를 사용하여 get-deployment 명령을 호출합니다.

   ```
   aws deploy get-deployment --deployment-id deployment-id --query
    "deploymentInfo.status" --output text
   ```

반환된 값이 Succeeded일 때까지 계속하지 마십시오.

배포를 만들려면(콘솔)

1. 2단계: CodeDeploy에 대한 서비스 역할 생성의 지침에 따라 이미 서비스 역할을 만들었을 것입니다. 서비스 역할은 CodeDeploy에 인스턴스에 액세스하여 태그를 확장할(읽을) 수 있는 권한을 부여합니다. CodeDeploy 콘솔을 사용하여 애플리케이션 수정을 배포하기 전에 서비스 역할 ARN이 필요합니다. 서비스 역할 ARN을 확인하려면 서비스 역할 ARN 확인(콘솔)의 지침을 따르십시오.
2. 이제 서비스 역할 ARN이 있으므로 CodeDeploy 콘솔을 사용하여 애플리케이션 수정을 배포할 수 있습니다.

   에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

3. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
4. 애플리케이션 생성을 선택합니다.
5. 사용자 지정 애플리케이션을 선택합니다.
6. 애플리케이션 이름에 **SimpleDemoApp**을 입력합니다.
7. 컴퓨팅 플랫폼에서 EC2/온프레미스를 선택합니다.
8. 애플리케이션 생성을 선택합니다.
9. 배포 그룹 탭에서 Create deployment group(배포 그룹 생성)을 선택합니다.
10. Deployment group name(배포 그룹 이름)에 **SimpleDemoDG**을 입력합니다.
11. 서비스 역할에서 서비스 역할의 이름을 선택합니다.
12. 배포 유형에서 In-place(현재 위치)를 선택합니다.
13. 환경 구성(Environment configuration)에서 Auto Scaling 그룹(Auto Scaling groups)을 선택한 다음 **CodeDeployDemo-AS-Group**을 선택합니다.
14. 배포 구성에서 CodeDeployDefault.OneAtATime을 선택합니다.
15. 로드 밸런싱 활성화의 선택을 취소합니다.
16. [Create deployment group]을 선택합니다.

17. 배포 그룹 페이지에서 배포 생성을 선택합니다.
18. 수정 유형에서 내 애플리케이션은 Amazon S3에 저장됨을 선택합니다.
19. 개정 위치에 운영 체제 및 리전의 샘플 애플리케이션 위치를 입력합니다.

    Amazon Linux 및 RHEL Amazon EC2 인스턴스의 경우

리전	샘플 애플리케이션의 위치
미국 동부(오하이오) 리전	`http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip`
미국 동부(버지니아 북부) 리전	`http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip`
미국 서부(캘리포니아 북부) 리전	`http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip`
미국 서부(오리건) 리전	`http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip`
캐나다(중부) 리전	`http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip`
유럽(아일랜드) 리전	`http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip`

리전	샘플 애플리케이션의 위치
Europe (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip
Europe (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip
유럽(프랑크푸르트) 리전	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip
이스라엘(텔아비브) 리전	https://aws-codedeploy-il-central-1.s3.il-central-1.amazonaws.com/samples/latest/SampleApp_Linux.zip
Asia Pacific (Hong Kong) Region	https://aws-codedeploy-ap-east-1.s3.ap-east-1.amazonaws.com/samples/latest/SampleApp_Linux.zip
Asia Pacific (Tokyo) Region	http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip
아시아 태평양(서울) 리전	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip

리전	샘플 애플리케이션의 위치
아시아 태평양(싱가포르) 리전	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip
Asia Pacific (Sydney) Region	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip
Asia Pacific (Melbourne) Region	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/samples/latest/SampleApp_Linux.zip
아시아 태평양(뭄바이) 리전	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip
남아메리카(상파울루) 리전	http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip

Windows Server Amazon EC2 인스턴스의 경우

리전	샘플 애플리케이션의 위치
미국 동부(오하이오) 리전	http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/samples/latest/SampleApp_Windows.zip

리전	샘플 애플리케이션의 위치
미국 동부(버지니아 북부) 리전	http://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip
미국 서부(캘리포니아 북부) 리전	http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/samples/latest/SampleApp_Windows.zip
미국 서부(오리건) 리전	http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip
캐나다(중부) 리전	http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/samples/latest/SampleApp_Windows.zip
유럽(아일랜드) 리전	http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip
Europe (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip
Europe (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip

리전	샘플 애플리케이션의 위치
유럽(프랑크푸르트) 리전	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip
이스라엘(텔아비브) 리전	https://aws-codedeploy-il-central-1.s3.il-central-1.amazonaws.com/samples/latest/SampleApp_Windows.zip
아시아 태평양(홍콩)	https://aws-codedeploy-ap-east-1.s3.ap-east-1.amazonaws.com/samples/latest/SampleApp_Windows.zip
아시아 태평양(서울) 리전	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Windows.zip
아시아 태평양(싱가포르) 리전	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip
Asia Pacific (Sydney) Region	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip
Asia Pacific (Melbourne) Region	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/samples/latest/SampleApp_Windows.zip

리전	샘플 애플리케이션의 위치
아시아 태평양(뭄바이) 리전	`http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip`
남아메리카(상파울루) 리전	`http://s3-sa-east-1.amazonaws.com/aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip`

Ubuntu Server Amazon EC2 인스턴스의 경우

Amazon S3에 저장된 사용자 지정 애플리케이션 수정의 위치를 입력합니다.

20. 배포 설명은 비워 둡니다.
21. 고급을 확장합니다.
22. 배포 만들기를 선택합니다.

> **Note**
>
> 성공(Succeeded) 대신 실패(Failed)가 나타난 경우 배포 모니터링 및 문제 해결(애플리케이션 이름 **SimpleDemoApp**, 배포 그룹 이름 **SimpleDemoDG**를 사용하여)에서 몇 가지 기술을 시도할 수 있습니다.

3단계: 결과 확인

이 단계에서는 CodeDeploy가 Auto Scaling 그룹의 단일 Amazon EC2 인스턴스에 **SimpleDemoApp** 수정 버전을 배포했는지 확인합니다.

주제

- 결과를 확인하려면(CLI)
- 결과를 확인하려면(콘솔)

결과를 확인하려면(CLI)

먼저, Amazon EC2 인스턴스의 퍼블릭 DNS가 필요합니다.

를 사용하여 describe-instances 명령을 호출 AWS CLI 하여 Auto Scaling 그룹에 있는 Amazon EC2 인스턴스의 퍼블릭 DNS를 가져옵니다.

이 명령을 호출하기 전에 Amazon EC2 인스턴스의 ID가 필요합니다. ID를 가져오려면 이전에 수행한 것과 같이 **CodeDeployDemo-AS-Group**에 대해 describe-auto-scaling-groups를 호출합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --output text
```

이제 describe-instances 명령을 호출합니다.

```
aws ec2 describe-instances --instance-id instance-id --query
 "Reservations[0].Instances[0].PublicDnsName" --output text
```

반환되는 값은 Amazon EC2 인스턴스의 퍼블릭 DNS입니다.

웹 브라우저에서 다음과 같은 URL을 사용하여 Amazon EC2 인스턴스에 배포된 SimpleDemoApp 수정 버전을 표시합니다.

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

축하 페이지가 표시되면 CodeDeploy를 사용하여 Auto Scaling 그룹의 단일 Amazon EC2 인스턴스에 수정 버전을 성공적으로 배포한 것입니다.

그런 다음 Auto Scaling 그룹에 Amazon EC2 인스턴스를 추가합니다. Amazon EC2 Auto Scaling이 Amazon EC2 인스턴스를 추가하면 CodeDeploy가 새 인스턴스에 수정 버전을 배포합니다.

결과를 확인하려면(콘솔)

먼저, Amazon EC2 인스턴스의 퍼블릭 DNS가 필요합니다.

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

Amazon EC2 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택한 후 **CodeDeployDemo-AS-Group** 항목을 선택합니다.

인스턴스 탭의 목록에서 Amazon EC2 인스턴스 ID를 선택합니다.

인스턴스 페이지의 설명 탭에서 퍼블릭 DNS 값을 확인합니다.
ec2-01-234-567-890.compute-1.amazonaws.com과 같아야 합니다.

웹 브라우저에서 다음과 같은 URL을 사용하여 Amazon EC2 인스턴스에 배포된 SimpleDemoApp 수정 버전을 표시합니다.

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

축하 페이지가 표시되면 CodeDeploy를 사용하여 Auto Scaling 그룹의 단일 Amazon EC2 인스턴스에 수정 버전을 성공적으로 배포한 것입니다.

그런 다음 Auto Scaling 그룹에 Amazon EC2 인스턴스를 추가합니다. Amazon EC2 Auto Scaling이 Amazon EC2 인스턴스를 추가하면 CodeDeploy가 새 Amazon EC2 인스턴스에 수정 버전을 배포합니다.

4단계: Auto Scaling 그룹에서 Amazon EC2 인스턴스의 수 늘리기

이 단계에서는 Auto Scaling 그룹에 추가 Amazon EC2 인스턴스를 생성하도록 지시합니다. Amazon EC2 Auto Scaling에서 인스턴스를 생성하면 CodeDeploy가 해당 인스턴스에 수정 버전을 배포합니다.

주제

- Auto Scaling 그룹에서 Amazon EC2 인스턴스의 수를 확장하려면(CLI 사용)
- 배포 그룹의 Amazon EC2 인스턴스 수를 늘리려면(콘솔)

Auto Scaling 그룹에서 Amazon EC2 인스턴스의 수를 확장하려면(CLI 사용)

1. update-auto-scaling-group 명령을 호출하여 **CodeDeployDemo-AS-Group**이라는 Auto Scaling 그룹 내 Amazon EC2 인스턴스를 하나에서 두 개로 늘립니다.

   로컬 Linux, macOS 또는 Unix 머신의 경우

   ```
   aws autoscaling update-auto-scaling-group \
     --auto-scaling-group-name CodeDeployDemo-AS-Group \
     --min-size 2 \
     --max-size 2 \
     --desired-capacity 2
   ```

   로컬 Windows 머신의 경우

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-
AS-Group --min-size 2 --max-size 2 --desired-capacity 2
```

2. Auto Scaling 그룹에 Amazon EC2 인스턴스가 두 개 있는지 확인합니다. **CodeDeployDemo-AS-Group**에 대해 describe-auto-scaling-groups 명령을 호출합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
 CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].[HealthStatus,
 LifecycleState]" --output text
```

반환된 값에 모두 `Healthy` 및 `InService`가 표시될 때까지 진행하지 마십시오.

배포 그룹의 Amazon EC2 인스턴스 수를 늘리려면(콘솔)

1. Amazon EC2 탐색 모음의 Auto Scaling에서 Auto Scaling 그룹을 선택한 후 **CodeDeployDemo-AS-Group**을(를) 선택합니다.
2. 작업을 선택한 후 편집을 선택합니다.
3. 세부 정보 탭의 원하는, 최소 및 최대 상자에 **2**를 입력한 다음 저장을 선택합니다.
4. [Instances] 탭을 선택합니다. 목록에 새 Amazon EC2 인스턴스가 나타날 것입니다. (인스턴스가 나타나지 않는 경우 [Refresh] 버튼을 몇 번 선택해야 할 수 있습니다.) 수명 주기 열에 값 InService가 나타나고, 상태 열에 값 정상이 나타날 때까지 진행하지 마세요.

5단계: 결과 다시 확인

이 단계에서는 CodeDeploy가 Auto Scaling 그룹의 새 인스턴스에 SimpleDemoApp 수정 버전을 설치했는지 확인합니다.

주제

- 자동 배포 결과를 확인하려면(CLI)
- 자동 배포 결과를 확인하려면(콘솔)

자동 배포 결과를 확인하려면(CLI)

1. get-deployment 명령을 호출하기 전에 자동 배포의 ID가 필요합니다. ID를 가져오려면 list-deployments이라는 애플리케이션 및 **SimpleDemoApp**라는 배포 그룹에 대해 **SimpleDemoDG** 명령을 호출하세요.

   ```
   aws deploy list-deployments --application-name SimpleDemoApp --deployment-group-
   name SimpleDemoDG --query "deployments" --output text
   ```

   배포 ID가 두 개 있을 것입니다. get-deployment 명령의 호출에 아직 사용하지 않은 ID를 사용합니다.

   ```
   aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.
   [status, creator]" --output text
   ```

   배포 상태 외에도 명령 출력에 autoScaling이(가) 표시되어야 합니다(autoScaling은(는) Amazon EC2 Auto Scaling이 배포를 생성했음을 의미).

   배포 상태에 Succeeded가 표시될 때까지 진행하지 마십시오.

2. describe-instances 명령을 호출하기 전에 새 Amazon EC2 인스턴스의 ID가 필요합니다. 이 ID를 가져오려면 **CodeDeployDemo-AS-Group**에 대해 describe-auto-scaling-groups 명령을 또 한 번 호출합니다.

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
    CodeDeployDemo-AS-Group --query "AutoScalingGroups[0].Instances[*].InstanceId" --
   output text
   ```

   이제 describe-instances 명령을 호출합니다.

   ```
   aws ec2 describe-instances --instance-id instance-id --query
    "Reservations[0].Instances[0].PublicDnsName" --output text
   ```

   describe-instances 명령의 출력에서 새 Amazon EC2 인스턴스의 퍼블릭 DNS를 기록해 둡니다.

3. 웹 브라우저에서 다음과 같은 URL을 사용하여 Amazon EC2 인스턴스에 배포된 SimpleDemoApp 수정 버전을 표시합니다.

   ```
   http://ec2-01-234-567-890.compute-1.amazonaws.com
   ```

축하 페이지가 표시되면 CodeDeploy를 사용하여 Auto Scaling 그룹의 확장된 Amazon EC2 인스턴스에 수정 버전을 성공적으로 배포한 것입니다.

자동 배포 결과를 확인하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 배포를 선택합니다.

3. Amazon EC2 Auto Scaling에서 생성한 배포의 배포 ID를 선택합니다.

   .

4. 배포 페이지에는 배포에 대한 정보가 표시됩니다. 일반적으로는 사용자가 직접 배포를 생성하지만 Amazon EC2 Auto Scaling이 사용자를 대신해 배포 하나를 생성하여 수정 버전을 새 Amazon EC2 인스턴스에 배포했습니다.
5. 페이지 상단에 성공이 표시되면 인스턴스에서 결과를 확인합니다. 먼저 해당 인스턴스의 퍼블릭 DNS를 가져와야 합니다.
6. Amazon EC2 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택한 후 **CodeDeployDemo-AS-Group** 항목을 선택합니다.
7. 인스턴스 탭에서 새 Amazon EC2 인스턴스의 ID를 선택합니다.
8. 인스턴스 페이지의 설명 탭에서 퍼블릭 DNS 값을 확인합니다. **ec2-01-234-567-890.compute-1.amazonaws.com**과 같아야 합니다.

다음과 같은 URL을 사용하여 인스턴스에 배포된 `SimpleDemoApp` 수정을 표시합니다.

```
http://ec2-01-234-567-890.compute-1.amazonaws.com
```

축하 페이지가 표시되면 CodeDeploy를 사용하여 Auto Scaling 그룹의 확장된 Amazon EC2 인스턴스에 수정 버전을 성공적으로 배포한 것입니다.

6단계: 정리

이 단계에서는 이 튜토리얼을 진행하면서 사용한 리소스에 대해 요금이 계속 부과되지 않도록 Auto Scaling 그룹을 삭제합니다. 필요에 따라 Auto Scaling 구성 및 CodeDeploy 배포 구성 요소 레코드를 삭제할 수 있습니다.

주제

- 리소스를 정리하려면(CLI)
- 리소스를 정리하려면(콘솔)

리소스를 정리하려면(CLI)

1. **CodeDeployDemo-AS-Group**에 대해 delete-auto-scaling-group 명령을 호출해 Auto Scaling 그룹을 삭제합니다. 그러면 Amazon EC2 인스턴스도 종료됩니다.

   ```
   aws autoscaling delete-auto-scaling-group --auto-scaling-group-name CodeDeployDemo-AS-Group --force-delete
   ```

2. 필요에 따라 **CodeDeployDemo-AS-Launch-Template**이라는 시작 구성에 대해 delete-launch-template 명령을 호출하여 Auto Scaling 시작 템플릿을 삭제합니다.

   ```
   aws ec2 delete-launch-template --launch-template-name CodeDeployDemo-AS-Launch-Template
   ```

3. 필요에 따라 **SimpleDemoApp**(이)라는 애플리케이션에 대해 delete-application 명령을 호출하여 CodeDeploy에서 애플리케이션을 삭제합니다. 그러면 연결된 모든 배포, 배포 그룹 및 수정 레코드도 삭제됩니다.

   ```
   aws deploy delete-application --application-name SimpleDemoApp
   ```

4. Systems Manager 상태 관리자 연결을 삭제하려면 delete-association 명령을 호출합니다.

   ```
   aws ssm delete-association --assocation-id association-id
   ```

   describe-association 명령을 호출하여 *association-id*를 얻을 수 있습니다.

   ```
   aws ssm describe-association --name AWS-ConfigureAWSPackage --targets
    Key=tag:Name,Values=CodeDeployDemo
   ```

리소스를 정리하려면(콘솔)

Amazon EC2 인스턴스도 종료하는 Auto Scaling 그룹을 삭제하려면 다음을 수행합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. Amazon EC2 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택한 후 **CodeDeployDemo-AS-Group** 항목을 선택합니다.
3. [Actions], [Delete], [Yes, Delete]를 차례로 선택합니다.

(선택 사항) 시작 템플릿을 삭제하려면

1. 탐색 모음의 Auto Scaling에서 시작 구성을 선택한 다음 **CodeDeployDemo-AS-Launch-Template**을(를) 선택합니다.
2. [Actions], [Delete launch configuration], [Yes, Delete]를 차례로 선택합니다.

1. 필요에 따라 CodeDeploy에서 애플리케이션을 삭제합니다. 그러면 연결된 모든 배포, 배포 그룹 및 수정 레코드도 삭제됩니다. https://console.aws.amazon.com/codedeploy에서 CodeDeploy 콘솔을 엽니다.
2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

   탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.

3. 애플리케이션 목록에서 SimpleDemoApp을 선택합니다.
4. 애플리케이션 세부 정보 페이지에서 애플리케이션 삭제를 선택합니다.
5. 메시지가 표시되면 **Delete**를 입력한 후 삭제를 선택합니다.

Systems Manager 상태 관리자 연결을 삭제하려면

1. https://console.aws.amazon.com/systems-manager AWS Systems Manager 콘솔을 엽니다.
2. 탐색 창에서 상태 관리자를 선택합니다.
3. 생성한 연결을 선택하고 삭제를 선택합니다.

튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포

이 튜토리얼에서는 CodeDeploy를 사용하여 GitHub에서 Amazon Linux를 실행하는 단일 Amazon EC2 인스턴스, 단일 Red Hat Enterprise Linux(RHEL) 인스턴스 또는 단일 Windows Server 인스턴스로 샘플 애플리케이션 수정 버전을 배포합니다. CodeDeploy와 GitHub 통합에 대한 자세한 내용은 GitHub와 CodeDeploy 통합 단원을 참조하세요.

> **Note**
>
> 또한 CodeDeploy를 사용하여 GitHub에서 Ubuntu Server 인스턴스로 애플리케이션 수정을 배포할 수도 있습니다. 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)의 2단계: 샘플 애플리케이션 수정 버전 만들기에서 설명된 샘플 수정을 사용하거나 Ubuntu Server 인스턴스 및 CodeDeploy와 호환되는 수정을 생성할 수 있습니다. 수정을 직접 만들려면 CodeDeploy의 개정 계획 및 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 단원을 참조하세요.

주제

- 사전 조건
- 1단계: GitHub 계정 설정
- 2단계: GitHub 리포지토리 만들기
- 3단계: GitHub 리포지토리로 샘플 애플리케이션 업로드
- 4단계: 인스턴스 프로비저닝
- 5단계: 애플리케이션 및 배포 그룹 만들기
- 6단계: 인스턴스에 애플리케이션 배포
- 7단계: 배포 모니터링 및 확인
- 8단계: 정리

사전 조건

이 자습서를 시작하기 전에 다음 작업을 수행해야 합니다.

- 로컬 시스템에 Git를 설치합니다. Git를 설치하려면 Git 다운로드를 참조하세요.
- AWS CLI설치 및 구성을 비롯하여 CodeDeploy 시작하기의 단계를 완료합니다. 이는를 사용하여 GitHub에서 인스턴스 AWS CLI 로 개정을 배포하려는 경우에 특히 중요합니다.

1단계: GitHub 계정 설정

수정을 저장할 GitHub 리포지토리를 만들려면 GitHub 계정이 필요합니다. 이미 GitHub 계정이 있으면 2단계: GitHub 리포지토리 만들기 단계로 건너뜁니다.

1. https://github.com/join으로 이동합니다.
2. 사용자 이름, 이메일 주소 및 암호를 입력합니다.
3. [Sign up for GitHub]를 선택한 후 지침을 따릅니다.

2단계: GitHub 리포지토리 만들기

수정을 저장할 GitHub 리포지토리가 필요합니다.

GitHub 리포지토리가 이미 있는 경우 이 자습서 전체에서 기존 리포지토리의 이름으로 **`CodeDeployGitHubDemo`**를 대체한 다음 3단계: GitHub 리포지토리로 샘플 애플리케이션 업로드 단계로 건너뜁니다.

1. GitHub 홈 페이지에서 다음 중 하나를 수행합니다.
   - [Your repositories]에서 [New repository]를 선택합니다.
   - 탐색 모음에서 [Create new]([+])를 선택한 후 [New repository]를 선택합니다.
2. [Create a new repository] 페이지에서 다음을 수행합니다.
   - 리포지토리 이름 상자에 **`CodeDeployGitHubDemo`**를 입력합니다.
   - [Public]을 선택합니다.

> **Note**
>
> 기본 [Public] 옵션을 선택하면 모든 사람이 이 리포지토리를 볼 수 있습니다. Private(프라이빗) 옵션을 선택하여 리포지토리를 보고 커밋할 수 있는 사람을 제한할 수 있습니다.

- [Initialize this repository with a README] 확인란의 선택을 취소합니다. 대신에 다음 단계에서 `README.md` 파일을 수동으로 만듭니다.
- 리포지토리 생성을 선택합니다.

3. 로컬 머신 유형의 지침에 따라 명령줄을 사용하여 리포지토리를 만듭니다.

> **Note**
>
> GitHub에서 이중 인증을 활성화한 경우, 암호를 물으면 GitHub 로그인 암호 대신에 개인 액세스 토큰을 입력해야 합니다. 자세한 내용은 2FA 인증 코드 제공 단원을 참조하세요.

로컬 Linux, macOS 또는 Unix 머신의 경우

1. 터미널에서 다음 명령을 한 번에 하나씩 실행합니다. 여기에서 *user-name*은 GitHub 사용자 이름입니다.

```
mkdir /tmp/CodeDeployGitHubDemo
```

```
cd /tmp/CodeDeployGitHubDemo
```

```
touch README.md
```

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

2. /tmp/CodeDeployGitHubDemo 위치에서 터미널을 열린 상태로 둡니다.

로컬 Windows 머신의 경우

1. 관리자로 실행되는 명령 프롬프트에서 다음 명령을 한 번에 하나씩 실행합니다.

```
mkdir c:\temp\CodeDeployGitHubDemo
```

```
cd c:\temp\CodeDeployGitHubDemo
```

```
notepad README.md
```

2. 메모장에 README.md 파일을 저장합니다. 메모장을 닫습니다. 다음 명령을 한 번에 하나씩 실행합니다. 여기에서 *user-name*은 GitHub 사용자 이름입니다.

```
git init
```

```
git add README.md
```

```
git commit -m "My first commit"
```

```
git remote add origin https://github.com/user-name/CodeDeployGitHubDemo.git
```

```
git push -u origin master
```

3. c:\temp\CodeDeployGitHubDemo 위치에서 명령 프롬프트를 열린 상태로 둡니다.

3단계: GitHub 리포지토리로 샘플 애플리케이션 업로드

이 단계에서는 퍼블릭 Amazon S3 버킷에서 GitHub 리포지토리로 샘플 계정을 복사합니다. (간결한 설명을 위해 이 튜토리얼에서 제공되는 샘플 수정은 단일 웹 페이지입니다.)

> **Note**
>
> 샘플 수정 대신 자체 수정 중 하나를 사용하려는 경우 수정은 다음과 같아야 합니다.
>
> - CodeDeploy의 개정 계획 및 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가의 지침을 따라야 합니다.
> - 해당 인스턴스 유형에서 사용할 수 있어야 합니다.
> - GitHub 대시보드에서 액세스할 수 있어야 합니다.
>
> 자체 수정이 이러한 요구 사항을 충족하는 경우 5단계: 애플리케이션 및 배포 그룹 만들기 단계로 건너뜁니다.
> Ubuntu Server 인스턴스에 배포하는 경우, Ubuntu Server인스턴스 및 CodeDeploy와 호환되는 수정을 GitHub 리포지토리로 업로드해야 합니다. 자세한 내용은 CodeDeploy의 개정 계획 및 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 단원을 참조하세요.

주제

- 로컬 Linux, macOS 또는 Unix 시스템에서 샘플 수정 버전 푸시
- 로컬 Windows 시스템에서 샘플 수정 푸시

로컬 Linux, macOS 또는 Unix 시스템에서 샘플 수정 버전 푸시

예를 들어 `/tmp/CodeDeployGitHubDemo` 위치에서 아직 열려 있는 터미널을 사용하여 다음 명령을 한 번에 하나씩 실행합니다.

> **Note**
>
> Windows Server 인스턴스에 배포하려는 경우 명령에서 `SampleApp_Linux.zip`을(를) `SampleApp_Windows.zip`(으)로 대체합니다.

```
(Amazon S3 copy command)
```

```
unzip SampleApp_Linux.zip
```

```
rm SampleApp_Linux.zip
```

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

여기서 *(Amazon S3 ## ##)*은 다음 중 하나입니다.

- 미국 동부(오하이오) 리전에 `aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip . --region us-east-2`
- 미국 동부(버지니아 북부) 리전에 `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region us-east-1`
- 미국 서부(캘리포니아 북부) 리전에 `aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip . --region us-west-1`
- 미국 서부(오리건) 리전에 `aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region us-west-2`
- 캐나다(중부) 리전에 `aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip . --region ca-central-1`
- `aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region eu-west-1` 유럽(아일랜드) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region eu-west-2` 유럽(런던) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region eu-west-3` 유럽(파리) 리전의 경우
- 유럽(프랑크푸르트) 리전 `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --region eu-central-1`
- 이스라엘(텔아비브) 리전용 `aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Linux.zip . --region il-central-1`
- `aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Linux.zip . --region ap-east-1` 아시아 태평양(홍콩) 리전의 경우

- `aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --region ap-northeast-1` 아시아 태평양(도쿄) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --region ap-northeast-2` 아시아 태평양(서울) 리전의 경우
- 아시아 태평양(싱가포르) 리전에 `aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --region ap-southeast-1`
- `aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --region ap-southeast-2` 아시아 태평양(시드니) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/SampleApp_Linux.zip . --region ap-southeast-4` 아시아 태평양(멜버른) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1` 아시아 태평양(뭄바이) 리전의 경우
- 남아메리카(상파울루) 리전에 `aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-1`

로컬 Windows 시스템에서 샘플 수정 푸시

예를 들어 `c:\temp\CodeDeployGitHubDemo` 위치 등에서 명령 프롬프트가 열려 있는 상태에서 다음 명령을 한 번에 하나씩 실행합니다.

> **Note**
>
> Amazon Linux 또는 RHEL 인스턴스에 배포하려는 경우 명령에서 `SampleApp_Windows.zip`을(를) `SampleApp_Linux.zip`(으)로 대체합니다.

```
(Amazon S3 copy command)
```

새 하위 디렉터리가 아니라 로컬 디렉터리(예: `c:\temp\CodeDeployGitHubDemo`)에 직접 the ZIP 파일의 내용을 풉니다.

```
git add .
```

```
git commit -m "Added sample app"
```

```
git push
```

여기서 *(Amazon S3 ## ##)*은 다음 중 하나입니다.

- 미국 동부(오하이오) 리전에 `aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Windows.zip . --region us-east-2`
- 미국 동부(버지니아 북부) 리전에 `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Windows.zip . --region us-east-1`
- 미국 서부(캘리포니아 북부) 리전에 `aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Windows.zip . --region us-west-1`
- 미국 서부(오리건) 리전에 `aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Windows.zip . --region us-west-2`
- 캐나다(중부) 리전에 `aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Windows.zip . --region ca-central-1`
- `aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Windows.zip . --region eu-west-1` 유럽(아일랜드) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Windows.zip . --region eu-west-2` 유럽(런던) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Windows.zip . --region eu-west-3` 유럽(파리) 리전의 경우
- 유럽(프랑크푸르트) 리전 `aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Windows.zip . --region eu-central-1`
- 이스라엘(텔아비브) 리전용 `aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Windows.zip . --region il-central-1`
- `aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Windows.zip . --region ap-east-1` 아시아 태평양(홍콩) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Windows.zip . --region ap-northeast-1` 아시아 태평양(도쿄) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Windows.zip . --region ap-northeast-2` 아시아 태평양(서울) 리전의 경우
- 아시아 태평양(싱가포르) 리전에 `aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Windows.zip . --region ap-southeast-1`

- `aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Windows.zip . --region ap-southeast-2` 아시아 태평양(시드니) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/SampleApp_Windows.zip . --region ap-southeast-4` 아시아 태평양(멜버른) 리전의 경우
- `aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Windows.zip . --region ap-south-1` 아시아 태평양(뭄바이) 리전의 경우
- 남아메리카(상파울루) 리전에 `aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Windows.zip . --region sa-east-1`

Ubuntu Server 인스턴스로 자체 수정을 푸시하려면 로컬 repo로 계정을 복사하고 다음 명령을 호출합니다.

```
git add .
git commit -m "Added Ubuntu app"
git push
```

4단계: 인스턴스 프로비저닝

이 단계에는 샘플 애플리케이션을 배포할 인스턴스를 만들거나 구성합니다. CodeDeploy에서 지원하는 운영 체제 중 하나를 실행하는 Amazon EC2 인스턴스 또는 온프레미스 인스턴스에 배포할 수 있습니다. 자세한 내용은 CodeDeploy 에이전트에서 지원하는 운영 체제 단원을 참조하세요. CodeDeploy 배포에서 사용하기 위해 구성한 인스턴스가 이미 있는 경우에는 다음 단계로 건너뜁니다.

인스턴스를 프로비저닝하려면

1. Amazon EC2 인스턴스 시작(콘솔)의 지침에 따라 인스턴스를 프로비저닝할 수 있습니다.
2. 인스턴스를 시작할 때, 태그 추가에서 태그를 지정하는 것을 잊지 마십시오. 태그를 지정하는 방법에 대한 자세한 내용은 Amazon EC2 인스턴스 시작(콘솔) 단원을 참조하세요.

인스턴스에서 CodeDeploy 에이전트가 실행 중인지 확인하려면

- 에이전트가 실행 중인지 확인하려면 CodeDeploy 에이전트가 실행 중인지 확인의 지침을 따릅니다.

인스턴스를 성공적으로 프로비저닝했으면 CodeDeploy 에이전트가 실행 중인지 확인한 후 다음 단계로 진행합니다.

5단계: 애플리케이션 및 배포 그룹 만들기

이 단계에서는 CodeDeploy 콘솔 또는 AWS CLI 를 사용하여 GitHub 리포지토리에서 샘플 개정을 배포하는 데 사용할 애플리케이션 및 배포 그룹을 생성합니다.

애플리케이션 및 배포 그룹 만들기(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. 애플리케이션 생성을 선택한 후 사용자 지정 애플리케이션을 선택합니다.
4. 애플리케이션 이름에 **CodeDeployGitHubDemo-App**을 입력합니다.
5. Compute Platform(컴퓨팅 플랫폼)에서 EC2/On-premises(EC2/온프레미스)를 선택합니다.
6. 애플리케이션 생성을 선택합니다.
7. 배포 그룹 탭에서 Create deployment group(배포 그룹 생성)을 선택합니다.
8. Deployment group name(배포 그룹 이름)에 **CodeDeployGitHubDemo-DepGrp**을 입력합니다.
9. 서비스 역할에서 CodeDeploy 에 대한 서비스 역할 생성에서 생성한 CodeDeploy 서비스 역할의 이름을 선택합니다.
10. 배포 유형에서 In-place(현재 위치)를 선택합니다.
11. 환경 구성에서 사용 중인 인스턴스 유형에 따라 Amazon EC2 인스턴스 또는 온프레미스 인스턴스를 선택합니다. 키 및 값에 4단계: 인스턴스 프로비저닝의 일부로 인스턴스에 적용된 인스턴스 태그 키와 값을 입력합니다.
12. Deployment configuration(배포 구성)에서 CodeDeployDefault.AllatOnce를 선택합니다.
13. 로드 밸런서에서 Enable load balancing(로드 밸런싱 활성화)을 선택 해제합니다.
14. 고급을 확장합니다.
15. 경보에서 Ignore alarm configuration(경보 구성 무시)을 선택합니다.
16. 배포 그룹 생성을 선택하고 다음 단계로 계속 진행합니다.

애플리케이션 및 배포 그룹 만들기(CLI)

1. create-application 명령을 호출하여 CodeDeploy에 `CodeDeployGitHubDemo-App`(이)라는 애플리케이션을 생성합니다.

```
aws deploy create-application --application-name CodeDeployGitHubDemo-App
```

2. create-deployment-group 명령을 호출하여 `CodeDeployGitHubDemo-DepGrp`라는 배포 그룹을 생성합니다.

   - Amazon EC2 인스턴스에 배포하는 경우 *ec2-tag-key*는 4단계: 인스턴스 프로비저닝의 일부로 Amazon EC2 인스턴스에 적용된 Amazon EC2 인스턴스 태그 키입니다.
   - Amazon EC2 인스턴스에 배포하는 경우 *ec2-tag-value*는 4단계: 인스턴스 프로비저닝의 일부로 Amazon EC2 인스턴스에 적용된 Amazon EC2 인스턴스 태그 값입니다.
   - 온프레미스 인스턴스에 배포하는 경우, *on-premises-tag-key*는 4단계: 인스턴스 프로비저닝의 일부로 온프레미스 인스턴스에 적용된 온프레미스 인스턴스 태그 키입니다.
   - 온프레미스 인스턴스에 배포하는 경우, *on-premises-tag-value*는 4단계: 인스턴스 프로비저닝의 일부로 온프레미스 인스턴스에 적용된 온프레미스 인스턴스 태그 값입니다.
   - *service-role-arn*은 CodeDeploy에 대한 서비스 역할 생성에서 생성한 서비스 역할의 서비스 역할 ARN 입니다. (서비스 역할 ARN을 찾으려면 서비스 역할 ARN 확인(CLI)의 지침을 따르십시오.)

```
aws deploy create-deployment-group --application-name CodeDeployGitHubDemo-App
 --ec2-tag-filters Key=ec2-tag-key,Type=KEY_AND_VALUE,Value=ec2-tag-value --on-
premises-tag-filters Key=on-premises-tag-key,Type=KEY_AND_VALUE,Value=on-premises-
tag-value --deployment-group-name CodeDeployGitHubDemo-DepGrp --service-role-
arn service-role-arn
```

> **Note**
>
> create-deployment-group 명령은 배포 및 인스턴스의 지정된 이벤트에 대해 주제의 구독자에게 Amazon SNS 알림을 전송하게 하는 트리거를 생성합니다. 또한 해당 명령은 Amazon CloudWatch 경보 모니터링 임계값에 도달한 경우 자동으로 배포를 롤백하고 배포를 중지하도록 경보를 설정하는 옵션을 지원합니다. 이 작업에 대한 명령은 이 자습서에 포함되지 않습니다.

6단계: 인스턴스에 애플리케이션 배포

이 단계에서는 CodeDeploy 콘솔 또는 AWS CLI 를 사용하여 GitHub 리포지토리에서 인스턴스로 샘플 개정을 배포합니다.

수정을 배포하려면(콘솔)

1. 배포 그룹 세부 정보 페이지에서 배포 생성을 선택합니다.
2. 배포 그룹에서 **CodeDeployGitHubDemo-DepGrp**을(를) 선택합니다.
3. Revision type(수정 유형)에서 GitHub를 선택합니다.
4. [Connect to GitHub]에서 다음 중 하나를 수행합니다.
   - GitHub 계정에 대한 CodeDeploy 애플리케이션의 연결을 만들려면 별도의 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. GitHub 계정에서 이 연결을 식별하는 이름을 입력한 후 GitHub에 연결을 선택합니다. 웹 페이지에 CodeDeployGitHubDemo-App(이)라는 애플리케이션에 대해 GitHub와 상호 작용할 권한을 CodeDeploy에 부여하라는 메시지가 표시됩니다. 계속해서 5단계를 진행합니다.
   - 이미 만든 연결을 사용하려면 [GitHub account]에서 이름을 선택한 후 [Connect to GitHub]를 선택합니다. 계속해서 7단계를 진행합니다.
   - 다른 GitHub 계정에 대한 연결을 만들려면 별도의 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. [Connect to a different GitHub account]를 선택한 후 [Connect to GitHub]를 선택합니다. 계속해서 5단계를 진행합니다.
5. [Sign in] 페이지의 지침에 따라 GitHub 계정에 로그인합니다.
6. [Authorize application] 페이지에서 [Authorize application]을 선택합니다.
7. CodeDeploy 배포 생성 페이지에서 리포지토리 이름에 로그인하는 데 사용한 GitHub 사용자 이름과 그 뒤에 슬래시(/) 및 애플리케이션 수정 버전(예: ***my-github-user-name*/CodeDeployGitHubDemo**)을 푸시한 리포지토리 이름을 입력합니다.

   입력할 값을 잘 모르거나 다른 리포지토리를 지정하려면 다음과 같이 합니다.

   a. 별도의 웹 브라우저 탭에서 GitHub 대시보드로 이동합니다.

   b. [Your repositories]에서 마우스 포인터를 대상 리포지토리 이름 위에 올려놓습니다. 도구 설명에 GitHub 사용자 또는 조직 이름과 그 뒤에 슬래시(/) 및 리포지토리 이름이 표시됩니다. 이 값을 리포지토리 이름에 입력합니다.

> **Note**
>
> 대상 리포지토리 이름이 Your repositories(리포지토리)에 표시되지 않는 경우, Search GitHub(GitHub 검색) 상자를 사용하여 대상 리포지토리와 GitHub 사용자 또는 조직 이름을 찾습니다.

8. 커밋 ID 상자에 애플리케이션 수정을 GitHub에 푸시한 것과 연결된 커밋 ID를 입력합니다.

   입력할 값을 잘 모를 경우 다음과 같이 합니다.

   a. 별도의 웹 브라우저 탭에서 GitHub 대시보드로 이동합니다.

   b. 리포지토리에서 CodeDeployGitHubDemo를 선택합니다.

   c. 커밋 목록에서 애플리케이션 수정을 GitHub에 푸시한 것과 연결된 커밋 ID를 찾아 복사합니다. 이 ID는 일반적으로 40자이고 문자와 숫자로 구성됩니다. (일반적으로 더 긴 버전의 첫 10자인 더 짧은 커밋 ID 버전을 사용하지 마십시오.)

   d. 커밋 ID를 [Commit ID] 상자에 붙여 넣습니다.

9. [Deploy]를 선택하고 다음 단계로 계속 진행합니다.

수정을 배포하려면(CLI)

GitHub와 상호 작용하는 AWS CLI 명령(예: 다음에 호출할 create-deployment 명령)을 호출하려면 먼저 CodeDeploy에 GitHub 사용자 계정을 사용하여 `CodeDeployGitHubDemo-App` 애플리케이션의 GitHub와 상호 작용할 수 있는 권한을 부여해야 합니다. 현재는 CodeDeploy 콘솔을 사용하여 이를 수행해야 합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.

3. CodeDeployGitHubDemo-App를 선택합니다.

4. 배포 탭에서 배포 만들기를 선택합니다.

> **Note**
>
> 새 배포를 만들지 않습니다. 이는 현재 GitHub 사용자 계정을 대신하여 CodeDeploy에 GitHub와 상호 작용할 권한을 부여하는 유일한 방법입니다.

5. 배포 그룹에서 CodeDeployGitHubDemo-DepGrp를 선택합니다.
6. Revision type(수정 유형)에서 GitHub를 선택합니다.
7. [Connect to GitHub]에서 다음 중 하나를 수행합니다.
   - GitHub 계정에 대한 CodeDeploy 애플리케이션의 연결을 만들려면 별도의 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. [GitHub account]에 이 연결을 식별하는 이름을 입력한 후 [Connect to GitHub]를 선택합니다. 웹 페이지에 `CodeDeployGitHubDemo-App`(이)라는 애플리케이션에 대해 GitHub와 상호 작용할 권한을 CodeDeploy에 부여하라는 메시지가 표시됩니다. 계속해서 8단계를 진행합니다.
   - 이미 만든 연결을 사용하려면 [GitHub account]에서 이름을 선택한 후 [Connect to GitHub]를 선택합니다. 계속해서 10단계를 진행합니다.
   - 다른 GitHub 계정에 대한 연결을 만들려면 별도의 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. [Connect to a different GitHub account]를 선택한 후 [Connect to GitHub]를 선택합니다. 계속해서 8단계를 진행합니다.
8. [Sign in] 페이지의 지침에 따라 GitHub 사용자 이름 또는 이메일 및 암호로 로그인합니다.
9. [Authorize application] 페이지에서 [Authorize application]을 선택합니다.
10. CodeDeploy 배포 생성 페이지에서 취소를 선택합니다.
11. create-deployment 명령을 호출하여 GitHub 리포지토리에서 인스턴스로 수정을 배포합니다. 여기에서,
    - *`repository`*는 GitHub 계정 이름에 슬래시(`/`)와 리포지토리 이름(`CodeDeployGitHubDemo`)이 추가된 형태입니다(예: `MyGitHubUserName/CodeDeployGitHubDemo`).

      사용할 값을 잘 모르거나 다른 리포지토리를 지정하려면:

      1. 별도의 웹 브라우저 탭에서 GitHub 대시보드로 이동합니다.
      2. [Your repositories]에서 마우스 포인터를 대상 리포지토리 이름 위에 올려놓습니다. 도구 설명에 GitHub 사용자 또는 조직 이름과 그 뒤에 슬래시(`/`) 및 리포지토리 이름이 표시됩니다. 이것이 사용할 값입니다.

> **Note**
>
> [Your repositories]에 대상 리포지토리 이름이 표시되지 않는 경우, [Search GitHub] 상자를 사용하여 대상 리포지토리와 해당 GitHub 사용자 또는 조직 이름을 찾습니다.

- *commit-id*는 리포지토리에 푸시한 애플리케이션 수정과 연결된 커밋입니다(예: `f835159a...528eb76f`).

  사용할 값을 잘 모를 경우:

  1. 별도의 웹 브라우저 탭에서 GitHub 대시보드로 이동합니다.
  2. 리포지토리에서 CodeDeployGitHubDemo를 선택합니다.
  3. 커밋 목록에서 애플리케이션 수정을 GitHub에 푸시한 것과 연결된 커밋 ID를 찾습니다. 이 ID는 일반적으로 40자이고 문자와 숫자로 구성됩니다. (일반적으로 더 긴 버전의 첫 10자인 더 짧은 커밋 ID 버전을 사용하지 마십시오.) 이 값을 사용합니다.

로컬 Linux, macOS 또는 Unix 머신에서 작업하는 경우:

```
aws deploy create-deployment \
  --application-name CodeDeployGitHubDemo-App \
  --deployment-config-name CodeDeployDefault.OneAtATime \
  --deployment-group-name CodeDeployGitHubDemo-DepGrp \
  --description "My GitHub deployment demo" \
  --github-location repository=repository,commitId=commit-id
```

로컬 Windows 머신에서 작업하는 경우:

```
aws deploy create-deployment --application-name CodeDeployGitHubDemo-App --
deployment-config-name CodeDeployDefault.OneAtATime --deployment-group-name
 CodeDeployGitHubDemo-DepGrp --description "My GitHub deployment demo" --github-
location repository=repository,commitId=commit-id
```

7단계: 배포 모니터링 및 확인

이 단계에서는 CodeDeploy 콘솔 또는 AWS CLI 를 사용하여 배포의 성공을 확인합니다. 직접 만들거나 구성한 인스턴스에 배포된 웹 페이지를 웹 브라우저에서 확인합니다.

> **Note**
>
> Ubuntu Server 인스턴스에 배포하는 경우, 배포된 수정이 해당 인스턴스에서 정상 작동하는지 자체 테스트 전략에 따라 확인한 후 다음 단계로 진행합니다.

배포를 모니터링하고 확인하려면(콘솔)

1. 탐색 창에서 배포를 확장하고 배포을 선택합니다.
2. 배포 목록에서 애플리케이션 값이 CodeDeployGitHubDemo-App이고 배포 그룹 값이 CodeDeployGitHubDemo-DepGrp인 행을 찾습니다. 상태 열에 성공 또는 실패가 표시되지 않은 경우 새로 고침 버튼을 몇 차례 누릅니다.
3. 상태 열에 실패가 나타난 경우 인스턴스 정보 보기(콘솔)의 지침을 따라 배포 문제를 해결합니다.
4. 상태열에 성공이 나타난 경우 웹 브라우저를 통해 배포를 확인할 수 있습니다. 이 샘플 수정에서는 인스턴스에 단일 웹 페이지를 배포합니다. Amazon EC2 인스턴스에 배포하는 경우 웹 브라우저에서 해당 인스턴스의 `http://`*`public-dns`*(으)로 이동합니다(예: `http://ec2-01-234-567-890.compute-1.amazonaws.com`).
5. 웹 페이지가 보이면 성공한 것입니다. 이제 AWS CodeDeploy 를 사용하여 GitHub에서 수정을 배포했으므로 8단계: 정리 단원으로 건너뛸 수 있습니다.

배포를 모니터링하고 확인하려면(CLI)

1. list-deployments 명령을 호출하여 `CodeDeployGitHubDemo-App`이라는 애플리케이션의 배포 ID와 `CodeDeployGitHubDemo-DepGrp`라는 배포 그룹을 가져옵니다.

```
aws deploy list-deployments --application-name CodeDeployGitHubDemo-App --
deployment-group-name CodeDeployGitHubDemo-DepGrp --query "deployments" --output
 text
```

2. list-deployments 명령의 출력에 배포 ID를 입력하여 get-deployment 명령을 호출합니다.

```
aws deploy get-deployment --deployment-id deployment-id --query "deploymentInfo.
[status, creator]" --output text
```

3. [Failed]가 반환되면 인스턴스 정보 보기(콘솔)의 지침에 따라 배포 문제를 해결합니다.
4. [Succeeded]가 반환되면 이제 웹 브라우저를 통해 배포를 확인해 볼 수 있습니다. 이 샘플 수정은 인스턴스에 배포된 단일 웹 페이지입니다. Amazon EC2 인스턴스에 배포하는 경우 웹 브라우저에

서 Amazon EC2 인스턴스의 `http://public-dns`(으)로 이동하여 이 페이지를 볼 수 있습니다(예: `http://ec2-01-234-567-890.compute-1.amazonaws.com`).

5. 웹 페이지가 보이면 성공한 것입니다. AWS CodeDeploy 를 사용하여 GitHub 리포지토리에서 성공적으로 배포했습니다.

8단계: 정리

이 자습서를 진행하면서 사용한 리소스에 요금이 부과되지 않도록 Amazon EC2 인스턴스와 연결된 리소스를 종료해야 합니다. 원한다면 이 자습서와 연결된 CodeDeploy 배포 구성 요소 레코드를 삭제할 수 있습니다. 이 자습서 진행을 위해서만 GitHub 리포지토리를 사용한 경우에는 지금 그 리포지토리도 삭제할 수 있습니다.

CloudFormation 스택을 삭제하려면(템플 CloudFormation 릿을 사용하여 Amazon EC2 인스턴스를 생성한 경우)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.
2. Stacks(스택) 열에서 `CodeDeploySampleStack`으로 시작하는 스택을 선택합니다.
3. 삭제를 선택합니다.
4. 메시지가 나타나면 스택 삭제를 선택합니다. Amazon EC2 인스턴스 및 연결된 IAM 인스턴스 프로파일과 서비스 역할이 삭제됩니다.

온프레미스 인스턴스를 수동으로 등록 취소 및 정리하려면(온프레미스 인스턴스를 프로비저닝한 경우)

1. AWS CLI 를 사용하여 your-*`your-instance-name`*와 your-region으로 표시된 관련 리전에 대해 등록 취소 명령을 호출합니다.

```
aws deploy deregister --instance-name your-instance-name --no-delete-iam-user --
region your-region
```

2. 온프레미스 인스턴스에서 uninstall 명령을 호출합니다.

```
aws deploy uninstall
```

Amazon EC2 인스턴스를 수동으로 종료하려면(Amazon EC2 인스턴스를 수동으로 시작한 경우)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 인스턴스에서 인스턴스를 선택합니다.
3. 종료할 Amazon EC2 인스턴스 옆의 상자를 선택합니다. [Actions] 메뉴에서 [Instance State]를 가리킨 다음 [Terminate]를 선택합니다.
4. 메시지가 나타나면 [Yes, Terminate]를 선택합니다.

CodeDeploy 배포 구성 요소 레코드를 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. CodeDeployGitHubDemo-App를 선택합니다.
4. 애플리케이션 삭제를 선택합니다.
5. 메시지가 표시되면 **`Delete`**를 입력한 후 삭제를 선택합니다.

GitHub 리포지토리를 삭제하려면

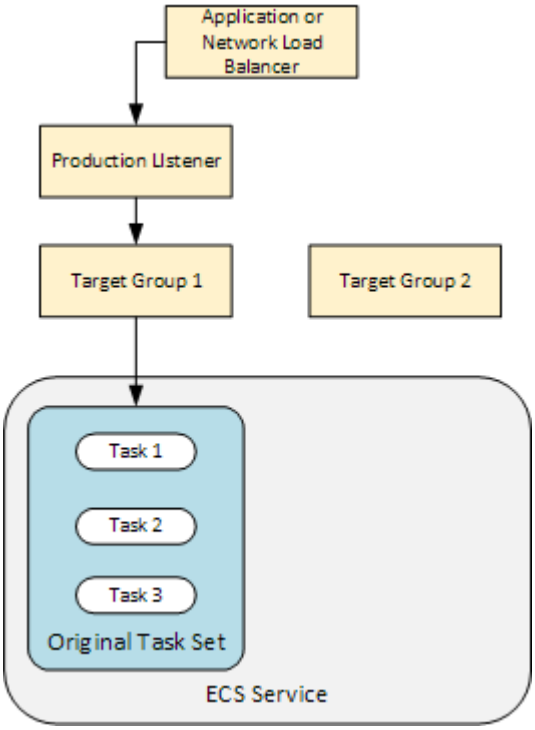
GitHub 도움말의 리포지토리 삭제를 참조하세요.

튜토리얼: Amazon ECS에 애플리케이션 배포

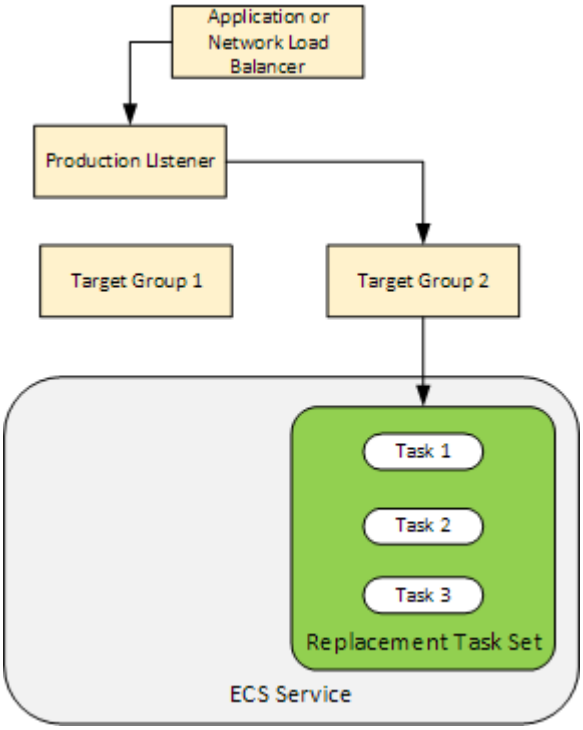
이 튜토리얼에서는 CodeDeploy를 사용하여 Amazon ECS에 애플리케이션을 배포하는 방법을 알아봅니다. 이미 만들고 Amazon ECS에 배포한 애플리케이션으로 시작합니다. 첫 번째 단계는 새 태그로 작업 정의 파일을 수정하여 애플리케이션을 업데이트하는 것입니다. 다음에는 CodeDeploy를 사용하여 업데이트를 배포합니다. 배포 중에 CodeDeploy는 새로운 대체 작업 세트에 업데이트를 설치합니다.

그런 다음, 원래 작업 세트에 있는 Amazon ECS 애플리케이션의 원래 버전에서 대체 작업 세트에 있는 업데이트된 버전으로 프로덕션 트래픽을 이동합니다.

Amazon ECS 배포 중에 CodeDeploy는 대상 그룹 두 개와 프로덕션 트래픽 리스너 하나로 구성된 로드 밸런서를 사용합니다. 다음 다이어그램에서는 배포가 시작되기 전에 로드 밸런서, 프로덕션 리스너, 대상 그룹 및 Amazon ECS 애플리케이션이 관련되는 방식을 보여 줍니다. 이 튜토리얼에서는 Application Load Balancer를 사용합니다. Network Load Balancer를 사용할 수도 있습니다.



성공적인 배포 후, 프로덕션 트래픽 리스너는 새로운 대체 작업 세트에 트래픽을 제공하고 원래 작업 세트는 종료됩니다. 다음 다이어그램에서는 성공적인 배포 후에 리소스가 관련되는 방식을 보여 줍니다. 자세한 내용은 Amazon ECS 배포 중에 발생하는 일 단원을 참조하십시오.



AWS CLI 를 사용하여 Amazon ECS에 애플리케이션을 배포하는 방법에 대한 자세한 내용은 자습서: 블루/그린 배포를 사용하여 서비스 생성을 참조하세요. CodePipeline을 사용하여 CodeDeploy가 있는 Amazon ECS 서비스에 대한 변경 사항을 감지하고 자동으로 배포하는 방법에 대한 자세한 내용은 튜토리얼: Amazon ECR 소스 및 ECS-to-CodeDeploy 배포를 사용하여 파이프라인 생성을 참조하세요.

이 자습서를 완료한 후에는 튜토리얼: Amazon ECS 서비스 배포 및 확인 테스트에서 CodeDeploy 애플리케이션과 생성한 배포 그룹을 사용하여 배포 확인 테스트를 추가할 수 있습니다.

주제

- 사전 조건
- 1단계: Amazon ECS 애플리케이션 업데이트
- 2단계: AppSpec 파일 생성
- 3단계: CodeDeploy 콘솔을 사용하여 애플리케이션 배포
- 4단계: 정리

사전 조건

이 자습서를 완료하려면 먼저 다음을 수행해야 합니다.

- CodeDeploy 시작하기의 2단계와 3단계를 완료합니다.

- 대상 그룹 두 개와 리스너 하나로 구성된 Application Load Balancer를 생성합니다. 콘솔을 사용하여 로드 밸런서를 생성하는 방법에 대한 자세한 내용은 CodeDeploy Amazon ECS 배포를 위한 로드 밸런서, 대상 그룹 및 리스너 설정 단원을 참조하세요. 를 사용하여 로드 밸런서를 생성하는 방법에 대한 자세한 내용은 Amazon Elastic Container Service 사용 설명서의 1단계: Application Load Balancer 생성을 AWS CLI참조하세요. 로드 밸런서를 생성할 때 이 자습서를 위해 다음을 기록해 둡니다.
  - 로드 밸런서의 이름입니다.
  - 대상 그룹의 이름입니다.
  - 로드 밸런서의 리스너에서 사용되는 포트입니다.
- Amazon ECS 클러스터 및 서비스를 생성합니다. 자세한 내용은 Amazon Elastic Container Service 사용 설명서에서 튜토리얼: 블루/그린 배포를 사용하여 서비스 생성의 2단계, 3단계, 4단계를 참조하세요. 이 자습서를 위해 다음을 기록해 둡니다.
  - Amazon ECS 클러스터의 이름입니다.
  - Amazon ECS 서비스에서 사용하는 작업 정의의 ARN입니다.
  - Amazon ECS 서비스에서 사용하는 컨테이너의 이름입니다.
- AppSpec 파일용 Amazon S3 버킷을 만듭니다.

1단계: Amazon ECS 애플리케이션 업데이트

이 단원에서는 해당 작업 정의의 새 버전을 사용하여 Amazon ECS 애플리케이션을 업데이트합니다. 업데이트된 수정은 새로운 키 및 태그 페어를 추가합니다. 3단계: CodeDeploy 콘솔을 사용하여 애플리케이션 배포에서는 Amazon ECS 애플리케이션의 업데이트된 버전을 배포합니다.

작업 정의를 업데이트하려면

1. https://console.aws.amazon.com/ecs/v2에서 콘솔을 엽니다.
2. 탐색 창에서 태스크 정의를 선택합니다.
3. Amazon ECS 서비스에 사용되는 태스크 정의를 선택합니다.
4. 태스크 정의 개정을 선택한 다음 새 개정 생성, 새 개정 생성을 선택합니다.
5. 이 자습서를 위해 태그를 추가하여 작업 정의에 대한 작은 업데이트를 수행합니다. 페이지 하단의 태그에서 새 키 및 값 페어를 입력하여 새 태그를 만듭니다.
6. 생성(Create)을 선택합니다.

   태스크 정의의 개정 번호가 하나 증가됩니다.

7. JSON 탭을 선택합니다. 다음 단계에서 이 정보가 필요하므로 다음 사항을 기록해 둡니다.

   - `taskDefinitionArn`의 값입니다. 형식은 `arn:aws:ecs:`*`aws-region`*`:`*`account-id`*`:task-definition/`*`task-definition-family`*`:`*`task-definition-revision`*입니다. 이 항목은 업데이트된 작업 정의의 ARN입니다.
   - `containerDefinitions` 요소에서 `name`의 값입니다. 이 항목은 컨테이너의 이름입니다.
   - `portMappings` 요소에서 `containerPort`의 값입니다. 이 항목은 컨테이너의 포트입니다.

2단계: AppSpec 파일 생성

이 단원에서는 AppSpec 파일을 생성하고 사전 조건 섹션에서 생성한 Amazon S3 버킷에 업로드합니다. Amazon ECS 배포를 위한 AppSpec 파일은 작업 정의, 컨테이너 이름 및 컨테이너 포트를 지정합니다. 자세한 내용은 Amazon ECS 배포용 AppSpec 파일 예제 및 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션 단원을 참조하세요.

AppSpec 파일을 생성하려면

1. YAML을 사용하여 AppSpec 파일을 생성하려는 경우 `appspec.yml`(이)라는 파일을 만듭니다. JSON을 사용하여 AppSpec 파일을 생성하려는 경우 `appspec.json`(이)라는 파일을 만듭니다.
2. AppSpec 파일에 YAML을 사용하는지 또는 JSON을 사용하는지에 따라 적절한 탭을 선택하고, 해당 내용을 방금 생성한 AppSpec 파일로 복사합니다. `TaskDefinition` 속성에는 1단계: Amazon ECS 애플리케이션 업데이트 단원에서 기록한 작업 정의 ARN을 사용합니다.

   JSON AppSpec

```
{
  "version": 0.0,
  "Resources": [
    {
      "TargetService": {
        "Type": "AWS::ECS::Service",
        "Properties": {
          "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number",
          "LoadBalancerInfo": {
            "ContainerName": "your-container-name",
            "ContainerPort": your-container-port
          }
        }
```

```
      }
    }
  ]
}
```

YAML AppSpec

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id:task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "your-container-name"
          ContainerPort: your-container-port
```

> **Note**
>
> 대체 작업 세트는 원래 작업 세트로부터 서브넷, 보안 그룹, 플랫폼 버전 및 할당된 퍼블릭 IP 값을 상속합니다. AppSpec 파일에서 선택적 속성을 설정하여 대체 작업 세트에 대한 이러한 값을 재정의할 수 있습니다. 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션 및 Amazon ECS 배포용 AppSpec 파일 예제 단원을 참조하세요.

3. AppSpec 파일을 이 튜토리얼의 사전 조건으로 생성한 S3 버킷에 업로드합니다.

3단계: CodeDeploy 콘솔을 사용하여 애플리케이션 배포

이 섹션에서는 CodeDeploy 애플리케이션과 배포 그룹을 생성하여 업데이트된 애플리케이션을 Amazon ECS에 배포합니다. 배포 중에 CodeDeploy는 애플리케이션에 대한 프로덕션 트래픽을 새로운 대체 작업 세트의 새로운 버전으로 이동합니다. 이 단계를 완료하려면 다음 항목이 필요합니다.

- Amazon ECS 클러스터 이름.
- Amazon ECS 서비스 이름.
- Application Load Balancer 이름
- 프로덕션 리스너 포트.

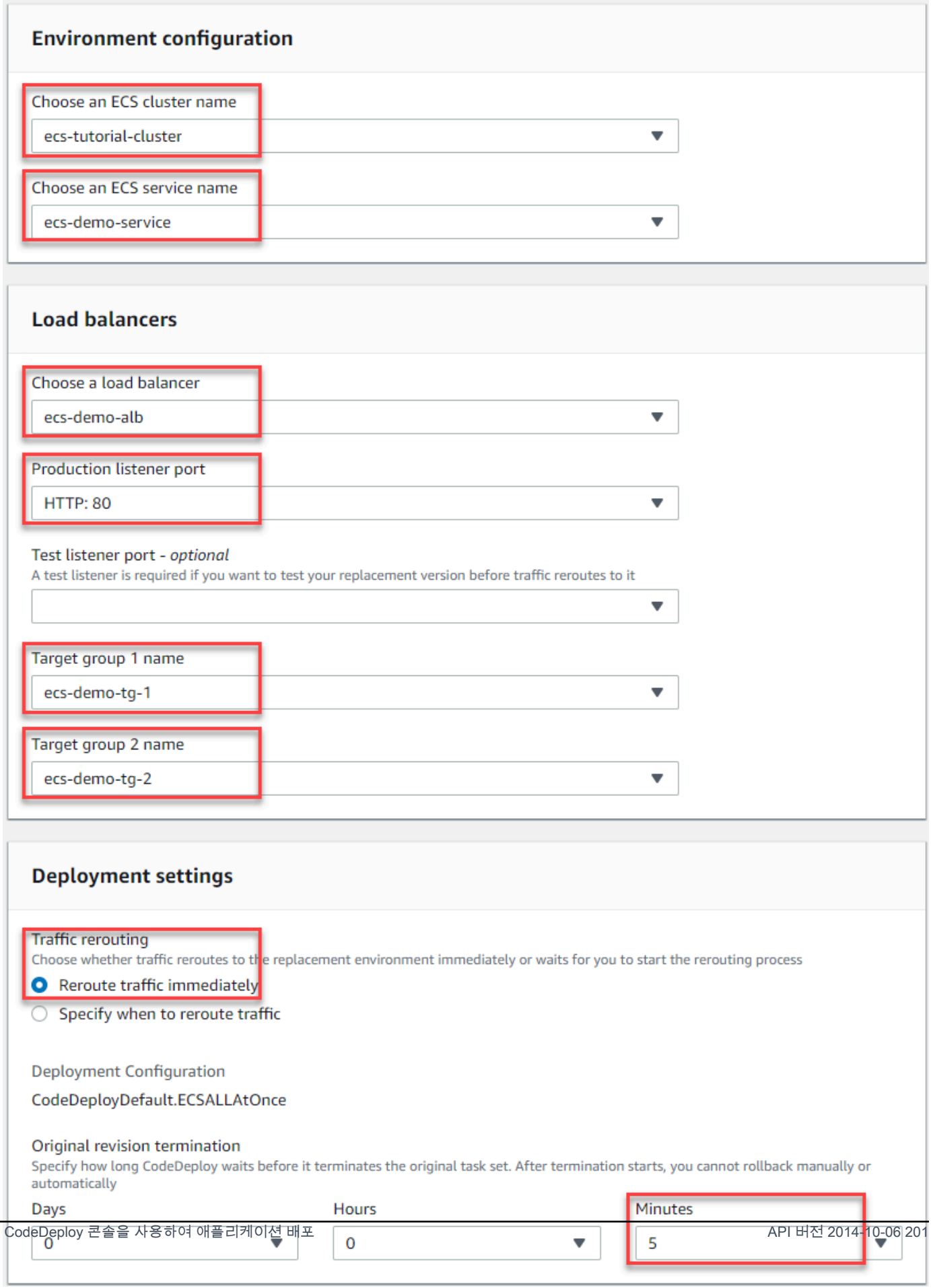
- 대상 그룹 이름.
- 생성한 S3 버킷의 이름.

CodeDeploy 애플리케이션을 만들려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy/ CodeDeploy 콘솔을 엽니다.
2. 애플리케이션 생성을 선택합니다.
3. 애플리케이션 이름에 **ecs-demo-codedeploy-app**을 입력합니다.
4. 컴퓨팅 플랫폼에서 Amazon ECS를 선택합니다.
5. 애플리케이션 생성을 선택합니다.

CodeDeploy 배포 그룹을 생성하려면

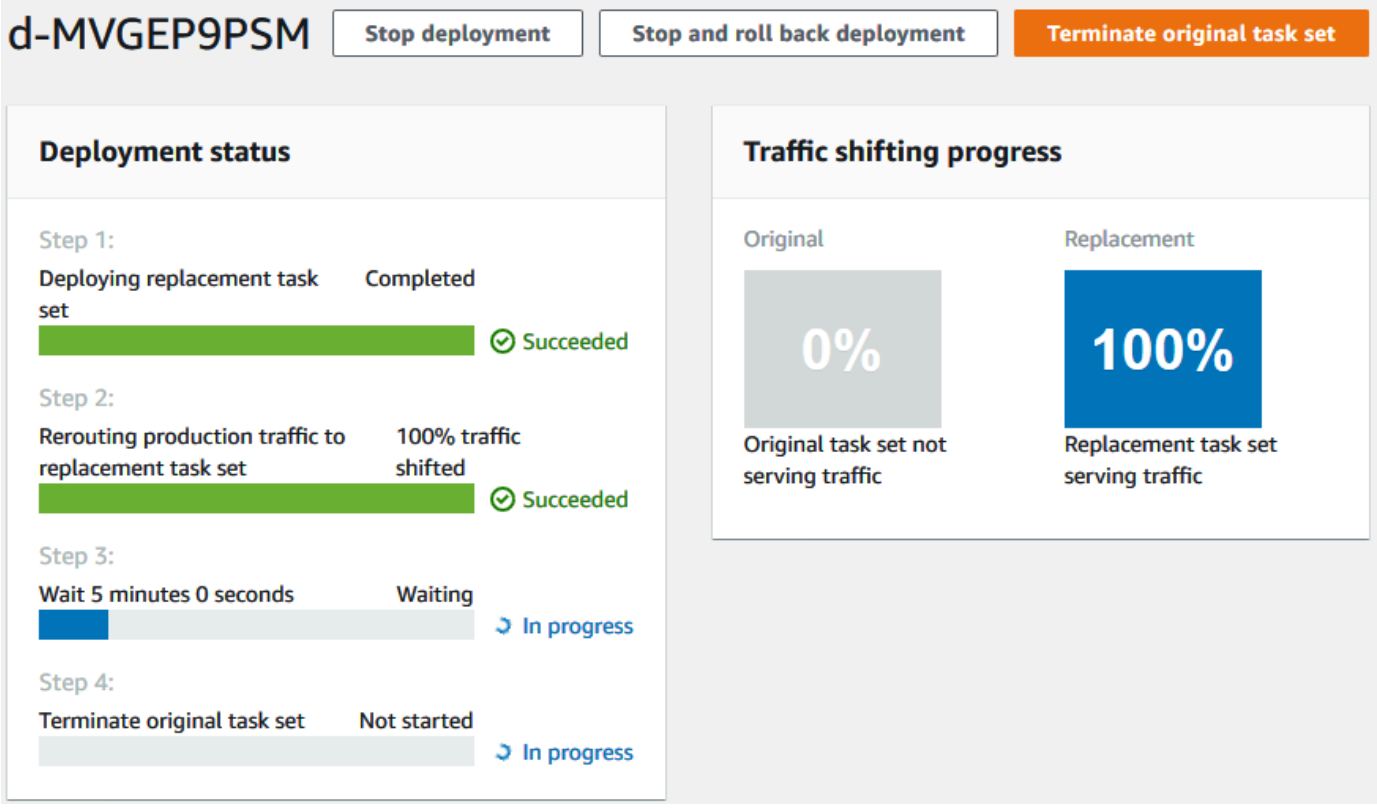
1. 애플리케이션 페이지의 Deployment groups(배포 그룹) 탭에서 Create deployment group(배포 그룹 생성)을 선택합니다.
2. Deployment group name(배포 그룹 이름)에 **ecs-demo-dg**을 입력합니다.
3. 서비스 역할에서 Amazon ECS에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할을 선택합니다. 자세한 내용은 에 대한 자격 증명 및 액세스 관리 AWS CodeDeploy 단원을 참조하십시오.
4. 환경 구성에서 Amazon ECS 클러스터 이름과 서비스 이름을 선택합니다.
5. 로드 밸런서에서 Amazon ECS 서비스에 트래픽을 공급하는 로드밸런서의 이름을 선택합니다.
6. 프로덕션 리스너 포트에서 Amazon ECS 서비스로 프로덕션 트래픽을 제공하는 리스너의 포트와 프로토콜을 선택합니다(예: HTTP: 80). 이 자습서에는 선택 사항인 테스트 리스너가 포함되지 않으므로, Test listener port(테스트 리스너 포트)에서 포트를 선택하지 마십시오.
7. Target group 1 name(대상 그룹 1 이름)과 Target group 2 name(대상 그룹 2 이름)에서 배포 중 트래픽을 라우팅하기 위한 다른 대상 그룹 두 개를 선택합니다. 해당 로드밸런서용으로 생성한 대상 그룹인지 확인합니다. 무엇이 대상 그룹 1에 사용되고 무엇이 대상 그룹 2에 사용되는지는 문제되지 않습니다.
8. Reroute traffic immediately(즉시 트래픽 다시 라우팅)를 선택합니다.
9. Original revision termination(원래 수정 종료)에서 0일, 0시간, 5분을 선택합니다. 이렇게 하면 기본값(1시간)을 선택하는 경우보다 더 빠르게 배포가 완료됩니다.


Environment configuration
Choose an ECS cluster name
ecs-tutorial-cluster
Choose an ECS service name
ecs-demo-service
Load balancers
Choose a load balancer
ecs-demo-alb
Production listener port
HTTP: 80
Test listener port - optional
A test listener is required if you want to test your replacement version before traffic reroutes to it
Target group 1 name
ecs-demo-tg-1
Target group 2 name
ecs-demo-tg-2
Deployment settings
Traffic rerouting
Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process
Reroute traffic immediately
Specify when to reroute traffic
Deployment Configuration
CodeDeployDefault.ECSALLAtOnce
Original revision termination
Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically
Days
0
Hours
0
Minutes
5

10. [Create deployment group]을 선택합니다.

Amazon ECS 애플리케이션을 배포하려면

1. 배포 그룹 콘솔 페이지에서 Create deployment(배포 생성)를 선택합니다.
2. Deployment group(배포 그룹)에서 ecs-demo-dg를 선택합니다.
3. Revision type(수정 유형)에서 My application is stored in Amazon S3(내 애플리케이션은 Amazon S3에 저장됨)를 선택합니다. Revision location(수정 위치)에 S3 버킷의 이름을 입력합니다.
4. Revision file type(수정 파일 유형)에서 .json 또는 .yaml을 적절하게 선택합니다.
5. (선택 사항) Deployment description(배포 설명)에 배포에 대한 설명을 입력합니다.
6. 배포 만들기를 선택합니다.
7. Deployment status(배포 상태)에서 배포를 모니터링할 수 있습니다. 프로덕션 트래픽의 100%가 대체 작업 세트로 라우팅된 후 5분 대기 시간이 만료되기 전에 원래 작업 세트 종료를 선택하여 원래 작업 세트를 즉시 종료할 수 있습니다. Terminate original task set(원래 작업 세트 종료)를 선택하지 않으면 지정한 5분 대기 시간이 만료된 후에 원래 작업 세트가 종료됩니다.



4단계: 정리

다음 자습서인 튜토리얼: Amazon ECS 서비스 배포 및 확인 테스트은(는) 이 자습서를 기반으로 진행되며 여기서 생성한 CodeDeploy 애플리케이션과 배포 그룹을 사용합니다. 해당 튜토리얼의 단계를 따르려는 경우 이 단계를 건너뛰고 여기서 생성한 리소스를 삭제하지 마십시오.

> **Note**
>
> AWS 계정에는 생성한 CodeDeploy 리소스에 대한 요금이 부과되지 않습니다.

이러한 단계의 리소스 이름은 이 튜토리얼에서 제안된 이름입니다(예: CodeDeploy 애플리케이션의 이름은 **ecs-demo-codedeploy-app**). 다른 이름을 사용한 경우 정리 중에 반드시 해당 이름을 사용해야 합니다.

1. delete-deployment-group 명령을 사용하여 CodeDeploy 배포 그룹을 삭제합니다.

   ```
   aws deploy delete-deployment-group --application-name ecs-demo-codedeploy-app --
   deployment-group-name ecs-demo-dg --region aws-region-id
   ```

2. delete-application 명령을 사용하여 CodeDeploy 애플리케이션을 삭제합니다.

   ```
   aws deploy delete-application --application-name ecs-demo-codedeploy-app --
   region aws-region-id
   ```

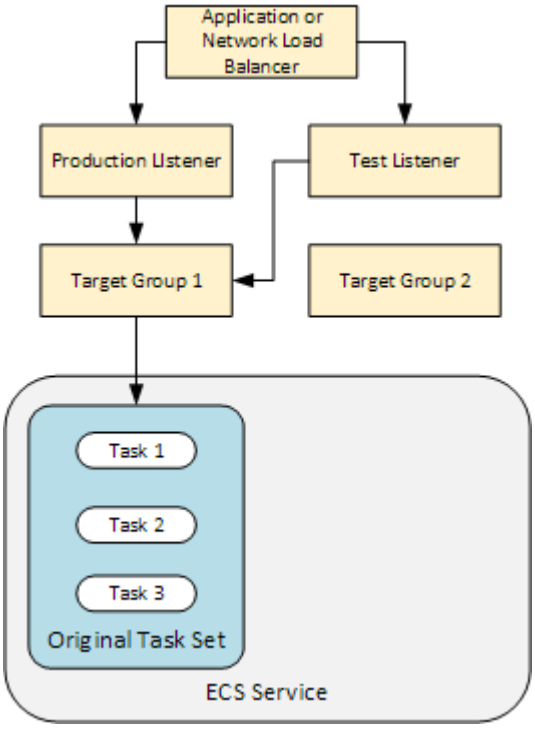
튜토리얼: Amazon ECS 서비스 배포 및 확인 테스트

이 튜토리얼에서는 Lambda 함수를 사용하여 업데이트된 Amazon ECS 애플리케이션의 배포 부분을 확인합니다. 이 튜토리얼에서는 튜토리얼: Amazon ECS에 애플리케이션 배포에서 사용한 CodeDeploy 애플리케이션, CodeDeploy 배포 그룹, Amazon ECS 애플리케이션을 사용합니다. 이 자습서를 시작하기 전에 해당 자습서를 완료하십시오.

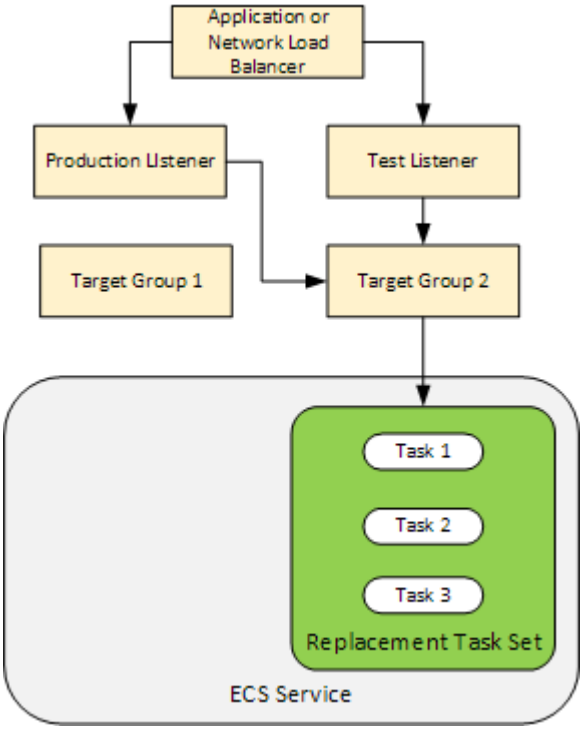
확인 테스트를 추가하려면 먼저 Lambda 함수에서 테스트를 구현합니다. 다음에는, 배포 AppSpec 파일에서 테스트할 수명 주기 후크에 대한 Lambda 함수를 지정합니다. 확인 테스트가 실패하면 배포가 중지되고 롤백되며 실패로 표시됩니다. 테스트가 성공하면 배포가 다음 배포 수명 주기 이벤트 또는 후크로 계속됩니다.

확인 테스트를 포함한 Amazon ECS 배포 중에 CodeDeploy는 프로덕션 트래픽 리스너 하나와 테스트 트래픽 리스너 하나라는 두 개의 대상 그룹으로 구성된 로드 밸런서를 사용합니다. 다음 다이어그램에

서는 배포가 시작되기 전에 로드 밸런서, 프로덕션 및 테스트 리스너, 대상 그룹 및 Amazon ECS 애플리케이션이 관련되는 방식을 보여 줍니다. 이 튜토리얼에서는 Application Load Balancer를 사용합니다. Network Load Balancer를 사용할 수도 있습니다.



Amazon ECS 배포 중에는 테스트를 위한 다섯 개의 수명 주기 후크가 있습니다. 이 튜토리얼에서는 세 번째 수명 주기 배포 후크인 `AfterAllowTestTraffic` 중에 테스트 하나를 구현합니다. 자세한 내용은 Amazon ECS 배포를 위한 수명 주기 이벤트 후크 목록 단원을 참조하십시오. 성공적인 배포 후, 프로덕션 트래픽 리스너는 새로운 대체 작업 세트에 트래픽을 제공하고 원래 작업 세트는 종료됩니다. 다음 다이어그램에서는 성공적인 배포 후에 리소스가 관련되는 방식을 보여 줍니다. 자세한 내용은 Amazon ECS 배포 중에 발생하는 일 단원을 참조하십시오.



> **Note**
>
> 이 자습서를 완료하면 AWS 계정에 요금이 부과될 수 있습니다. 여기에는 CodeDeploy AWS Lambda및 CloudWatch에 부과될 수 있는 요금이 포함됩니다. 자세한 내용은 AWS CodeDeploy 요금, AWS Lambda 요금 및 Amazon CloudWatch 요금을 참조하세요.

주제

- 사전 조건
- 1단계: 테스트 리스너 생성
- 2단계: Amazon ECS 애플리케이션 업데이트
- 3단계: 수명 주기 후크 Lambda 함수 생성
- 4단계: AppSpec 파일 업데이트
- 5단계: CodeDeploy 콘솔을 사용하여 Amazon ECS 서비스 배포
- 6단계: CloudWatch Logs에서 Lambda 후크 함수 출력 보기
- 7단계: 정리

사전 조건

이 자습서를 성공적으로 완료하려면 먼저 다음을 수행해야 합니다.

- 튜토리얼: Amazon ECS에 애플리케이션 배포에 대해 사전 조건의 사전 조건을 충족합니다.
- 튜토리얼: Amazon ECS에 애플리케이션 배포의 단계를 수행하세요. 다음 사항을 기록해 둡니다.
  - 로드 밸런서의 이름입니다.
  - 대상 그룹의 이름입니다.
  - 로드 밸런서의 리스너에서 사용되는 포트입니다.
  - 로드 밸런서의 ARN입니다. 이 항목을 사용하여 새 리스너를 생성합니다.
  - 대상 그룹 중 하나의 ARN입니다. 이 항목을 사용하여 새 리스너를 생성합니다.
  - 사용자가 생성하는 CodeDeploy 애플리케이션 및 배포 그룹입니다.
  - 사용자가 생성하는 AppSpec 파일입니다. 이 항목은 CodeDeploy 배포에서 사용됩니다. 이 튜토리얼에서는 이 파일을 편집합니다.

1단계: 테스트 리스너 생성

확인 테스트를 포함한 Amazon ECS 배포에는 두 번째 리스너가 필요합니다. 이 리스너는 대체 작업 세트에 있는 업데이트된 Amazon ECS 애플리케이션에 테스트 트래픽을 제공하는 데 사용됩니다. 확인 테스트는 테스트 트래픽에 대해 실행됩니다.

테스트 트래픽에 대한 리스너는 대상 그룹 중 하나를 사용할 수 있습니다. create-listener AWS CLI 명령을 사용하여 테스트 트래픽을 포트 8080으로 전달하는 기본 규칙으로 두 번째 리스너를 생성합니다. 로드 밸런서의 ARN과 대상 그룹 중 하나의 ARN을 사용합니다.

```
aws elbv2 create-listener --load-balancer-arn your-load-balancer-arn \
--protocol HTTP --port 8080 \
--default-actions Type=forward,TargetGroupArn=your-target-group-arn --region your-aws-region
```

2단계: Amazon ECS 애플리케이션 업데이트

이 단원에서는 해당 작업 정의의 새 버전을 사용하도록 Amazon ECS 애플리케이션을 업데이트합니다. 새 버전을 만들고 태그를 추가하여 마이너 업데이트를 새 버전에 추가합니다.

작업 정의를 업데이트하려면

1. https://console.aws.amazon.com/ecs/에서 Amazon ECS 클래식 콘솔을 엽니다.
2. 탐색 창에서 태스크 정의를 선택합니다.
3. Amazon ECS 서비스에 사용되는 작업 정의에 대한 확인란을 선택합니다.
4. Create new revision(새 수정 생성)을 선택합니다.
5. 태그를 추가하여 작업 정의에 대한 작은 업데이트를 수행합니다. 페이지 하단의 Tags(태그)에서 새 키 및 값 페어를 입력하여 새 태그를 만듭니다.
6. 생성(Create)을 선택합니다. 작업 정의의 수정 번호가 하나 증가된 것이 보여야 합니다.
7. JSON 탭을 선택합니다. `taskDefinitionArn`의 값을 기록해 둡니다. 형식은 `arn:aws:ecs:`*`aws-region`*`:`*`account-id`*`:task-definition/`*`task-definition-family`*`:`*`task-definition-revision`*입니다. 이 항목은 업데이트된 작업 정의의 ARN입니다.

3단계: 수명 주기 후크 Lambda 함수 생성

이 단원에서는 Amazon ECS 배포의 `AfterAllowTestTraffic` 후크에 대한 Lambda 함수를 구현합니다. Lambda 함수는 업데이트된 Amazon ECS 애플리케이션이 설치되기 전에 확인 테스트를 실행합니다. 이 튜토리얼의 경우 Lambda 함수는 `Succeeded`을(를) 반환합니다. 실제 배포 중에 확인 테스트는 확인 테스트의 결과에 따라 `Succeeded` 또는 `Failed`를 반환합니다. 또한 실제 배포 중에는 다른 Amazon ECS 배포 수명 주기 이벤트 후크(`BeforeInstall`, `AfterInstall`, `BeforeAllowTraffic`, `AfterAllowTraffic`) 중 하나 이상에 대한 Lambda 테스트 함수를 구현할 수 있습니다. 자세한 내용은 Amazon ECS 배포를 위한 수명 주기 이벤트 후크 목록 단원을 참조하십시오.

Lambda 함수를 생성하려면 IAM 역할이 필요합니다. 이 역할은 CloudWatch Logs에 쓰고 CodeDeploy 수명 주기 후크의 상태를 설정할 수 있는 권한을 Lambda 함수에 부여합니다.

IAM 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택한 후 역할 생성을 선택합니다.
3. 다음 속성을 사용하여 역할을 만듭니다.
   - 신뢰할 수 있는 엔터티: AWS Lambda.

- 권한: AWSLambdaBasicExecutionRole. 이 항목은 CloudWatch Logs에 기록할 수 있는 권한을 Lambda 함수에 부여합니다.
- 역할 이름: **`lambda-cli-hook-role`**.

자세한 내용은 AWS Lambda 실행 역할 생성을 참조하세요.

4. 생성한 역할에 `codedeploy:PutLifecycleEventHookExecutionStatus` 권한을 연결합니다. 이 항목은 배포 중에 CodeDeploy 수명 주기 후크의 상태를 설정할 수 있는 권한을 Lambda 함수에 부여합니다. 자세한 내용은 AWS Identity and Access Management 사용 설명서의 IAM 자격 증명 권한 추가 및 CodeDeploy API 참조의 PutLifecycleEventHookExecutionStatus를 참조하세요.

`AfterAllowTestTraffic` 후크 Lambda 함수를 생성하려면

1. 다음 콘텐츠를 가진 `AfterAllowTestTraffic.js`이라는 파일을 생성합니다:

```
'use strict';

 const AWS = require('aws-sdk');
 const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});

 exports.handler = (event, context, callback) => {

  console.log("Entering AfterAllowTestTraffic hook.");

  // Read the DeploymentId and LifecycleEventHookExecutionId from the event payload
  var deploymentId = event.DeploymentId;
  var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;
  var validationTestResult = "Failed";

  // Perform AfterAllowTestTraffic validation tests here. Set the test result
  // to "Succeeded" for this tutorial.
  console.log("This is where AfterAllowTestTraffic validation tests happen.")
  validationTestResult = "Succeeded";

  // Complete the AfterAllowTestTraffic hook by sending CodeDeploy the validation
 status
  var params = {
   deploymentId: deploymentId,
   lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
   status: validationTestResult // status can be 'Succeeded' or 'Failed'
```

```
  };

  // Pass CodeDeploy the prepared validation test results.
  codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
   if (err) {
    // Validation failed.
    console.log('AfterAllowTestTraffic validation tests failed');
    console.log(err, err.stack);
    callback("CodeDeploy Status update failed");
   } else {
    // Validation succeeded.
    console.log("AfterAllowTestTraffic validation tests succeeded");
    callback(null, "AfterAllowTestTraffic validation tests succeeded");
   }
  });
 }
```

2. Lambda 배포 패키지를 만듭니다.

```
zip AfterAllowTestTraffic.zip AfterAllowTestTraffic.js
```

3. `create-function` 명령을 사용하여 `AfterAllowTestTraffic` 후크에 대한 Lambda 함수를 생성합니다.

```
aws lambda create-function --function-name AfterAllowTestTraffic \
       --zip-file fileb://AfterAllowTestTraffic.zip \
       --handler AfterAllowTestTraffic.handler \
       --runtime nodejs10.x \
       --role arn:aws:iam::aws-account-id:role/lambda-cli-hook-role
```

4. `create-function` 응답에 있는 Lambda 함수 ARN을 기록해 둡니다. 다음 단계에서 CodeDeploy 배포의 AppSpec 파일을 업데이트할 때 이 ARN을 사용합니다.

4단계: AppSpec 파일 업데이트

이 단원에서는 `Hooks` 섹션을 사용하여 AppSpec 파일을 업데이트합니다. `Hooks` 섹션에서는 `AfterAllowTestTraffic` 수명 주기 후크에 대한 Lambda 함수를 지정합니다.

AppSpec 파일을 업데이트하려면

1. 튜토리얼: Amazon ECS에 애플리케이션 배포의 2단계: AppSpec 파일 생성에서 생성한 AppSpec 파일을 엽니다.
2. 2단계: Amazon ECS 애플리케이션 업데이트에서 기록한 작업 정의 ARN을 사용하여 `TaskDefinition` 속성을 업데이트합니다.
3. `Hooks` 섹션을 복사하여 AppSpec 파일에 붙여 넣습니다. 3단계: 수명 주기 후크 Lambda 함수 생성에서 기록한 Lambda 함수의 ARN을 사용하여 `AfterAllowTestTraffic` 이후에 ARN을 업데이트합니다.

   JSON AppSpec

   ```
   {
     "version": 0.0,
     "Resources": [
       {
         "TargetService": {
           "Type": "AWS::ECS::Service",
           "Properties": {
             "TaskDefinition": "arn:aws:ecs:aws-region-id:aws-account-id::task-
   definition/ecs-demo-task-definition:revision-number",
             "LoadBalancerInfo": {
               "ContainerName": "sample-website",
               "ContainerPort": 80
             }
           }
         }
       }
     ],
     "Hooks": [
       {
         "AfterAllowTestTraffic": "arn:aws:lambda:aws-region-id:aws-account-
   id:function:AfterAllowTestTraffic"
       }
     ]
   }
   ```

   YAML AppSpec

   ```
   version: 0.0
   Resources:
   ```

```
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:aws-region-id:aws-account-id::task-
definition/ecs-demo-task-definition:revision-number"
        LoadBalancerInfo:
          ContainerName: "sample-website"
          ContainerPort: 80
Hooks:
  - AfterAllowTestTraffic: "arn:aws:lambda:aws-region-id:aws-account-
id:function:AfterAllowTestTraffic"
```

4. AppSpec 파일을 저장하고 S3 버킷에 업로드합니다.

5단계: CodeDeploy 콘솔을 사용하여 Amazon ECS 서비스 배포

이 단원에서는 테스트 리스너를 위한 포트를 지정하여 배포 그룹을 업데이트합니다. 이 리스너는 1단계: 테스트 리스너 생성에서 생성한 리스너입니다. 배포 중에 CodeDeploy는 테스트 리스너를 사용하여 대체 작업 세트에 제공되는 테스트 트래픽을 사용하여 `AfterAllowTestTraffic` 배포 수명 주기 후크 중에 확인 테스트를 실행합니다. 확인 테스트는 `Succeeded` 결과를 반환하므로, 배포는 다음 배포 수명 주기 이벤트로 진행됩니다. 실제 시나리오에서 테스트 함수는 `Succeeded` 또는 `Failed`를 반환합니다.

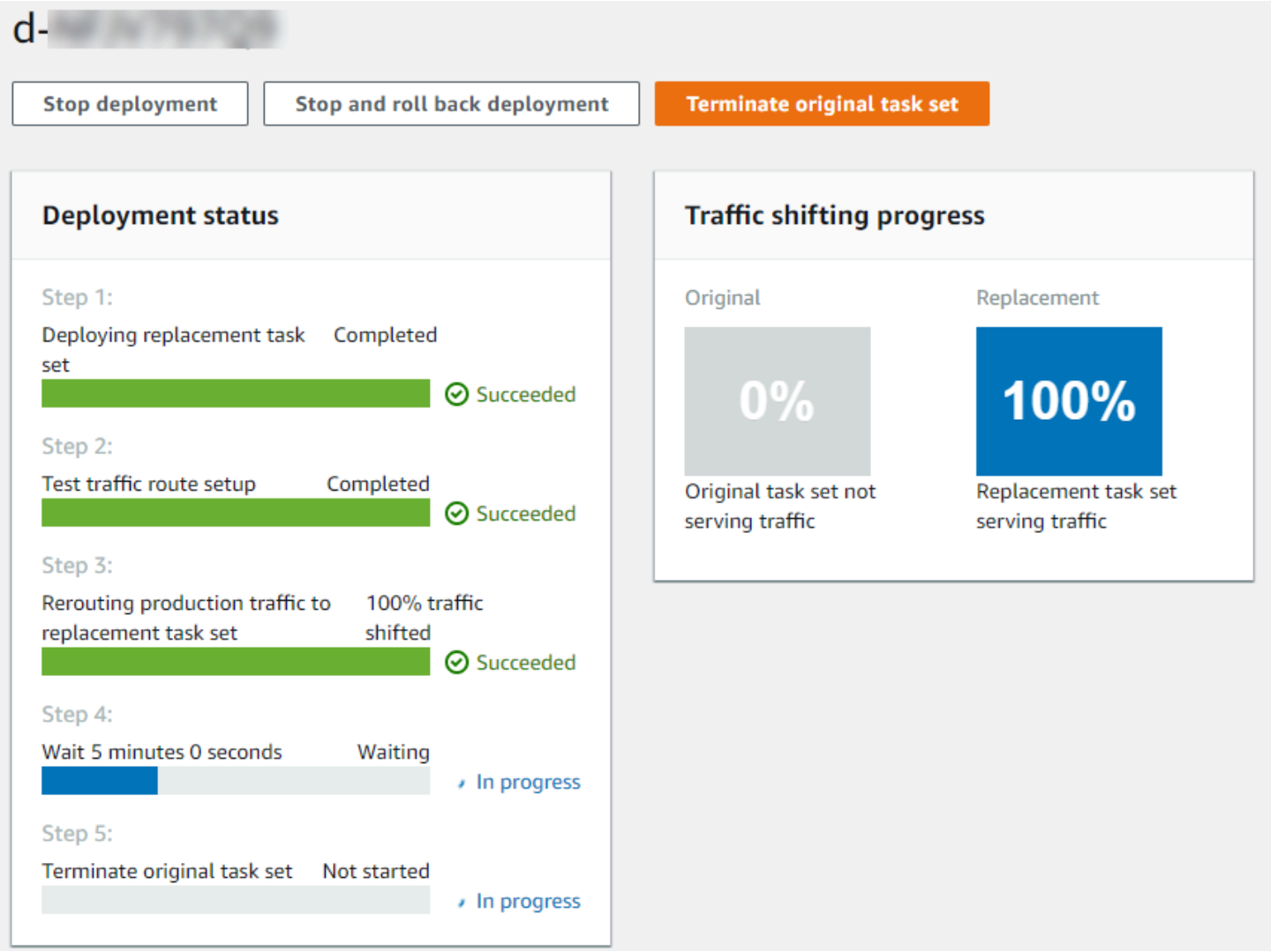
테스트 리스너를 배포 그룹에 추가하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy/ CodeDeploy 콘솔을 엽니다.
2. 탐색 창에서 Applications(애플리케이션)을 선택합니다.
3. 튜토리얼: Amazon ECS에 애플리케이션 배포에서 생성한 애플리케이션을 선택합니다. 제안된 이름을 사용한 경우 이 항목은 ecs-demo-codedeploy-app입니다.
4. Deployment groups(배포 그룹)에서, 튜토리얼: Amazon ECS에 애플리케이션 배포에서 생성한 배포 그룹을 선택합니다. 제안된 이름을 사용한 경우 이 항목은 ecs-demo-dg입니다.
5. 편집을 선택합니다.
6. Test listener port(테스트 리스너 포트)에서 이 튜토리얼의 앞부분에서 생성한 테스트 리스너를 위한 포트와 프로토콜을 선택합니다. 이 항목은 HTTP: 8080이어야 합니다.
7. 변경 사항 저장을 선택합니다.

Amazon ECS 애플리케이션을 배포하려면

1. 배포 그룹 콘솔 페이지에서 Create deployment(배포 생성)를 선택합니다.
2. Deployment group(배포 그룹)에서 ecs-demo-dg를 선택합니다.
3. Revision type(수정 유형)에서 My application is stored in Amazon S3(내 애플리케이션은 Amazon S3에 저장됨)를 선택합니다. 수정 버전 위치에 S3 버킷의 이름과 AppSpec 파일(예: **`s3://my-s3-bucket/appspec.json`**)을 입력합니다.
4. Revision file type(수정 파일 유형)에서 .json 또는 .yaml을 적절하게 선택합니다.
5. (선택 사항) Deployment description(배포 설명)에 배포에 대한 설명을 입력합니다.
6. 배포 만들기를 선택합니다.

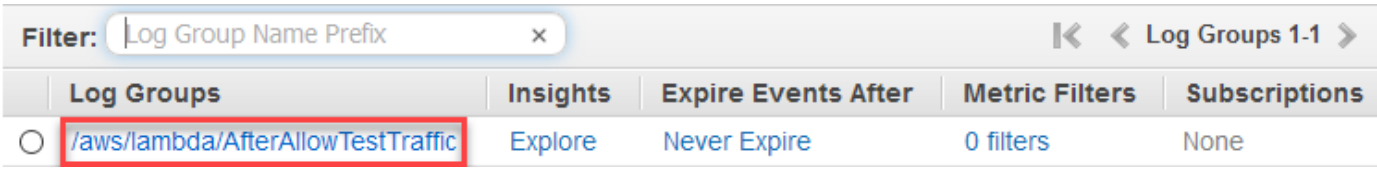
Deployment status(배포 상태)에서 배포를 모니터링할 수 있습니다. 프로덕션 트래픽의 100%가 대체 작업 세트로 라우팅된 후에는 원래 작업 세트 종료를 선택하여 원래 작업 세트를 즉시 종료할 수 있습니다. 원래 작업 세트 종료를 선택하지 않으면 배포 그룹을 생성할 때 지정한 기간 후에 원래 작업 세트가 종료됩니다.



6단계: CloudWatch Logs에서 Lambda 후크 함수 출력 보기

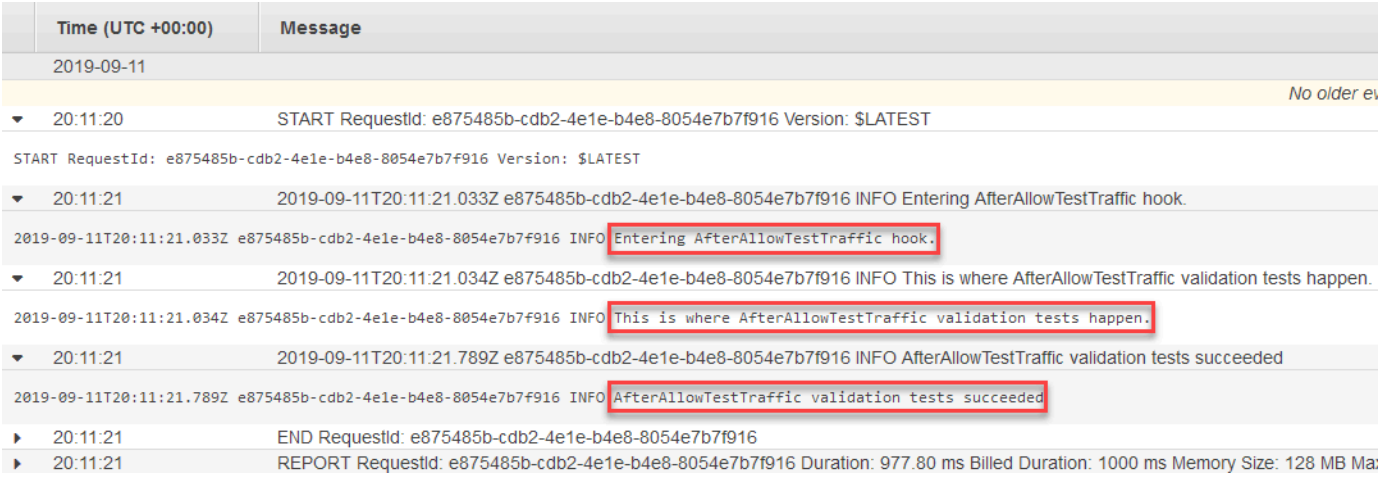
CodeDeploy 배포가 성공하면 Lambda 후크 함수의 확인 테스트도 성공합니다. CloudWatch Logs에서 후크 함수에 대한 로그를 조회하여 이 결과를 확인할 수 있습니다.

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택합니다. AppSpec 파일에서 지정한 Lambda 후크 함수에 대한 새로운 로그 그룹 하나가 보여야 합니다.



3. 새로운 로그 그룹을 선택합니다. 이 항목은 /aws/lambda/AfterAllowTestTrafficHook여야 합니다.

4. 로그 스트림을 선택합니다. 두 개 이상의 로그 그룹이 보이면 Last Event Time(마지막 이벤트 시간) 아래에 가장 최근 날짜와 시간이 있는 로그 그룹을 선택합니다.
5. 로그 스크림 이벤트를 확장하여 Lambda 후크 함수가 성공 메시지를 로그에 기록했는지 확인합니다. 다음은 `AfterAllowTraffic` Lambda 후크 함수가 성공했음을 보여 줍니다.



7단계: 정리

이 자습서를 완료한 후에는 사용하지 않는 리소스에 요금이 발생하지 않도록 연결된 리소스를 정리합니다. 이 단계의 리소스 이름은 이 튜토리얼에서 제안된 이름입니다(예: CodeDeploy 애플리케이션의 이름은**ecs-demo-codedeploy-app**). 다른 이름을 사용한 경우 정리에서 반드시 해당 이름을 사용해야 합니다.

자습서 리소스를 정리하려면

1. delete-deployment-group 명령을 사용하여 CodeDeploy 배포 그룹을 삭제합니다.

```
aws deploy delete-deployment-group --application-name ecs-demo-deployment-group --
deployment-group-name ecs-demo-dg --region aws-region-id
```

2. delete-application 명령을 사용하여 CodeDeploy 애플리케이션을 삭제합니다.

```
aws deploy delete-application --application-name ecs-demo-deployment-group --
region aws-region-id
```

3. delete-function 명령을 사용하여 Lambda 후크 함수를 삭제합니다.

```
aws lambda delete-function --function-name AfterAllowTestTraffic
```

4. delete-log-group 명령을 사용하여 CloudWatch 로그 그룹을 삭제합니다.

```
aws logs delete-log-group --log-group-name /aws/lambda/AfterAllowTestTraffic
```

자습서: CodeDeploy 및 AWS 서버리스 애플리케이션 모델을 사용하여 업데이트된 Lambda 함수 배포

AWS SAM은 서버리스 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. AWS SAM 템플릿의 YAML 구문을 CloudFormation 구문으로 변환하고 확장하여 Lambda 함수와 같은 서버리스 애플리케이션을 빌드합니다. 자세한 내용은 AWS Serverless Application Model이란 무엇입니까?를 참조하세요.

이 자습서에서는 AWS SAM을 사용하여 다음을 수행하는 솔루션을 생성합니다.

- Lambda 함수를 생성합니다.
- CodeDeploy 애플리케이션 및 배포 그룹을 생성합니다.
- CodeDeploy 수명 주기 후크 중에 배포 확인 테스트를 실행하는 Lambda 함수 두 개를 생성합니다.
- 언제 Lambda 함수가 업데이트되는지를 감지합니다. Lambda 함수를 업데이트하면 Lambda 함수의 원래 버전에서 업데이트된 버전으로 프로덕션 트래픽을 증분식으로 이동하는 CodeDeploy를 통해 배포가 트리거됩니다.

> **Note**
>
> 이 튜토리얼에서는 결과적으로 AWS 계정에 요금이 부과될 수 있는 리소스를 생성해야 합니다. 여기에는 CodeDeploy, Amazon CloudWatch 및에 대해 발생할 수 있는 요금이 포함됩니다 AWS Lambda. 자세한 내용은 CodeDeploy 요금, Amazon CloudWatch 요금 및 AWS Lambda 요금을 참조하세요.

주제

- 사전 조건
- 1단계: 인프라 설정
- 2단계: Lambda 함수 업데이트
- 3단계: 업데이트된 Lambda 함수 배포

- 4단계: 배포 결과 보기
- 5단계: 정리

사전 조건

이 자습서를 완료하려면 먼저 다음을 수행해야 합니다.

- CodeDeploy 시작하기의 단계를 수행하세요.
- AWS Serverless Application Model CLI를 설치합니다. 자세한 내용은 AWS SAM CLI 설치를 참조하세요.
- S3 버킷을 생성합니다. AWS SAM은 AWS SAM 템플릿에서 참조되는 아티팩트를이 버킷에 업로드합니다.

1단계: 인프라 설정

이 주제에서는를 AWS SAM 사용하여 AWS SAM 템플릿 및 Lambda 함수에 대한 파일을 생성하는 방법을 보여줍니다. 그런 다음 및 `deploy` 명령을 사용하여 AWS SAM `package` 인프라에서 구성 요소를 생성합니다. 인프라가 준비되면 CodeDeploy 애플리케이션 및 배포 그룹, 업데이트하고 배포할 Lambda 함수, Lambda 함수를 배포할 때 실행되는 확인 테스트가 포함된 Lambda 함수 두 개를 갖추게 됩니다. 완료되면 CloudFormation 를 사용하여 Lambda 콘솔 또는에서 구성 요소를 보고 Lambda 함수를 테스트 AWS CLI 할 수 있습니다.

주제

- 파일 생성
- AWS SAM 애플리케이션 패키징
- AWS SAM 애플리케이션 배포
- (선택 사항) 인프라 검사 및 테스트

파일 생성

인프라를 생성하려면 다음 파일을 생성해야 합니다.

- `template.yml`
- `myDateTimeFunction.js`
- `beforeAllowTraffic.js`

- `afterAllowTraffic.js`

주제

- AWS SAM 템플릿 생성
- Lambda 함수에 대한 파일 생성
- BeforeAllowTraffic Lambda 함수에 대한 파일 생성
- AfterAllowTraffic Lambda 함수에 대한 파일 생성

AWS SAM 템플릿 생성

인프라의 구성 요소를 지정하는 AWS SAM 템플릿 파일을 생성합니다.

AWS SAM 템플릿을 생성하려면

1. `SAM-Tutorial`이라는 디렉터리를 생성합니다.
2. `SAM-Tutorial` 디렉터리에서 `template.yml`이라는 파일을 생성합니다.
3. 다음 YAML 코드를 `template.yml`에 복사합니다. 이 파일은 AWS SAM 템플릿입니다.

```
AWSTemplateFormatVersion : '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: A sample SAM template for deploying Lambda functions.

Resources:
# Details about the myDateTimeFunction Lambda function
  myDateTimeFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: myDateTimeFunction.handler
      Runtime: nodejs18.x
# Instructs your myDateTimeFunction is published to an alias named "live".
      AutoPublishAlias: live
# Grants this function permission to call lambda:InvokeFunction
      Policies:
        - Version: "2012-10-17"
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: '*'
```

```
      DeploymentPreference:
# Specifies the deployment configuration
          Type: Linear10PercentEvery1Minute
# Specifies Lambda functions for deployment lifecycle hooks
          Hooks:
            PreTraffic: !Ref beforeAllowTraffic
            PostTraffic: !Ref afterAllowTraffic

# Specifies the BeforeAllowTraffic lifecycle hook Lambda function
  beforeAllowTraffic:
    Type: AWS::Serverless::Function
    Properties:
      Handler: beforeAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          Statement:
          - Effect: "Allow"
            Action:
              - "codedeploy:PutLifecycleEventHookExecutionStatus"
            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
# Grants this function permission to call lambda:InvokeFunction
          Statement:
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_beforeAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version

# Specifies the AfterAllowTraffic lifecycle hook Lambda function
  afterAllowTraffic:
    Type: AWS::Serverless::Function
```

```
    Properties:
      Handler: afterAllowTraffic.handler
      Policies:
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call
 codedeploy:PutLifecycleEventHookExecutionStatus
          - Effect: "Allow"
            Action:
              - "codedeploy:PutLifecycleEventHookExecutionStatus"
            Resource:
              !Sub 'arn:aws:codedeploy:${AWS::Region}:
${AWS::AccountId}:deploymentgroup:${ServerlessDeploymentApplication}/*'
        - Version: "2012-10-17"
          Statement:
# Grants this function permission to call lambda:InvokeFunction
          - Effect: "Allow"
            Action:
              - "lambda:InvokeFunction"
            Resource: !Ref myDateTimeFunction.Version
      Runtime: nodejs18.x
# Specifies the name of the Lambda hook function
      FunctionName: 'CodeDeployHook_afterAllowTraffic'
      DeploymentPreference:
        Enabled: false
      Timeout: 5
      Environment:
        Variables:
          NewVersion: !Ref myDateTimeFunction.Version
```

이 템플릿은 다음을 지정합니다. 자세한 내용은 AWS SAM 템플릿 개념을 참조하세요.

Lambda 함수 **myDateTimeFunction**

이 Lambda 함수가 게시되면 템플릿의 AutoPublishAlias줄은 이 함수를 live (이)라는 별칭에 연결합니다. 이 자습서의 뒷부분에서이 함수를 업데이트하면 프로덕션 트래픽을 원래 버전에서 업데이트된 버전으로 점진적으로 전환 AWS CodeDeploy 하여 배포가 트리거됩니다.

두 개의 Lambda 배포 확인 함수

다음 Lambda 함수는 CodeDeploy 수명 주기 후크 중에 실행됩니다. 이 함수에는 업데이트된 myDateTimeFunction의 배포를 확인하는 코드가 포함되어 있습니다. 확인 테스트의 결과는

PutLifecycleEventHookExecutionStatus API 메서드를 사용하여 CodeDeploy에 전달됩니다. 확인 테스트가 실패하면 배포가 실패하고 롤백됩니다.

- CodeDeployHook_beforeAllowTraffic은 BeforeAllowTraffic 후크 중에 실행됩니다.
- CodeDeployHook_afterAllowTraffic은 AfterAllowTraffic 후크 중에 실행됩니다.

두 함수의 이름은 모두 CodeDeployHook_로 시작합니다. CodeDeployRoleForLambda 역할은 이름이 이 접두사로 시작하는 Lambda 함수의 Lambda invoke 메서드에 대한 호출만 허용합니다. 자세한 내용은 AWS Lambda 배포를 위한 AppSpec 'hooks' 섹션와(과) CodeDeploy API 참조의 PutLifecycleEventHookExecutionStatus를 참조하세요.

업데이트된 Lambda 함수의 자동 감지

AutoPublishAlias 기간은 myDateTimeFunction 함수가 언제 변경되는지를 감지한 다음 live 별칭을 사용하여 함수를 배포합니다.

배포 구성

배포 구성에 따라 CodeDeploy 애플리케이션이 Lambda 함수의 원래 버전에서 새 버전으로 트래픽을 이동하는 속도가 결정됩니다. 이 템플릿은 미리 정의된 배포 구성인 Linear10PercentEvery1Minute를 지정합니다.

> **Note**
>
> AWS SAM 템플릿에서는 사용자 지정 배포 구성을 지정할 수 없습니다. 자세한 내용은 Create a Deployment Configuration 단원을 참조하십시오.

배포 수명 주기 후크 함수

Hooks 섹션은 수명 주기 이벤트 후크 중에 실행할 함수를 지정합니다. PreTraffic은 BeforeAllowTraffic 후크 중에 실행되는 함수를 지정합니다. PostTraffic는 AfterAllowTraffic 후크 중에 실행되는 함수를 지정합니다.

또 다른 Lambda 함수를 호출할 Lambda의 권한

지정된 lambda:InvokeFunction 권한은 AWS SAM 애플리케이션에서 사용하는 역할에 Lambda 함수를 호출할 수 있는 권한을 부여합니다. 이 권한은 확인 테스트 중에 CodeDeployHook_beforeAllowTraffic 및 CodeDeployHook_afterAllowTraffic 함수가 배포된 Lambda 함수를 호출할 때 필요합니다.

Lambda 함수에 대한 파일 생성

이 튜토리얼에서 나중에 업데이트하고 배포하는 함수에 대한 파일을 생성합니다.

> **Note**
>
> Lambda 함수는 AWS Lambda에서 지원되는 모든 런타임을 사용할 수 있습니다. 자세한 내용은 AWS Lambda 런타임을 참조하세요.

Lambda 함수를 만들려면

1. 텍스트 파일을 생성하고 `SAM-Tutorial` 디렉터리에 `myDateTimeFunction.js`로 저장합니다.
2. 다음 Node.js 코드를 `myDateTimeFunction.js`에 복사합니다.

```
'use strict';

    exports.handler = function(event, context, callback) {

      if (event.body) {
        event = JSON.parse(event.body);
      }

      var sc; // Status code
      var result = ""; // Response payload

      switch(event.option) {
        case "date":
          switch(event.period) {
            case "yesterday":
              result = setDateResult("yesterday");
              sc = 200;
              break;
            case "today":
              result = setDateResult();
              sc = 200;
              break;
            case "tomorrow":
              result = setDateResult("tomorrow");
              sc = 200;
              break;
```

```
            default:
              result = {
                "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
              };
              sc = 400;
              break;
          }
          break;

    /*      Later in this tutorial, you update this function by uncommenting
            this section. The framework created by AWS SAM detects the update
            and triggers a deployment by CodeDeploy. The deployment shifts
            production traffic to the updated version of this function.

            case "time":
            var d = new Date();
            var h = d.getHours();
            var mi = d.getMinutes();
            var s = d.getSeconds();

            result = {
              "hour": h,
              "minute": mi,
              "second": s
            };
            sc = 200;
            break;
    */
          default:
            result = {
              "error": "Must specify 'date' or 'time'."
            };
            sc = 400;
          break;
      }

      const response = {
        statusCode: sc,
        headers: { "Content-type": "application/json" },
        body: JSON.stringify( result )
      };

      callback(null, response);
```

```
    function setDateResult(option) {

      var d = new Date(); // Today
      var mo; // Month
      var da; // Day
      var y; // Year

      switch(option) {
        case "yesterday":
          d.setDate(d.getDate() - 1);
          break;
        case "tomorrow":
          d.setDate(d.getDate() + 1);
        default:
         break;
      }

      mo = d.getMonth() + 1; // Months are zero offset (0-11)
      da = d.getDate();
      y = d.getFullYear();

      result = {
        "month": mo,
        "day": da,
        "year": y
      };

      return result;
    }
  };
```

Lambda 함수는 어제, 오늘 또는 내일의 일, 월 및 연도를 반환합니다. 이 튜토리얼의 뒷부분에서, 지정하는 일 또는 시간(예: 일, 월, 년 또는 현재 시간, 분, 초)에 대한 정보를 반환하도록 함수를 업데이트하는 코드의 주석 처리를 해제합니다. 에서 생성한 프레임워크는 업데이트된 버전의 함수를 AWS SAM 감지하고 배포합니다.

> **Note**
>
> 이 Lambda 함수는 AWS Cloud9 자습서에서도 사용됩니다.는 클라우드 기반 통합 개발 환경 AWS Cloud9 입니다. 에서이 함수를 생성, 실행, 업데이트 및 디버깅하는 방법에 대한 자세한 내용은 자습서를 AWS Cloud9참조하세요. AWS LambdaAWS Cloud9

BeforeAllowTraffic Lambda 함수에 대한 파일 생성

`beforeAllowTraffic` 후크 Lambda 함수에 대한 파일을 생성합니다.

1. 텍스트 파일을 생성하고 `SAM-Tutorial` 디렉터리에 `beforeAllowTraffic.js`로 저장합니다.
2. 다음 Node.js 코드를 `beforeAllowTraffic.js`에 복사합니다. 이 함수는 배포의 `BeforeAllowTraffic` 후크 중에 실행됩니다.

```
'use strict';

    const AWS = require('aws-sdk');
    const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
    var lambda = new AWS.Lambda();

    exports.handler = (event, context, callback) => {

     console.log("Entering PreTraffic Hook!");

     // Read the DeploymentId and LifecycleEventHookExecutionId from the event
 payload
      var deploymentId = event.DeploymentId;
     var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

     var functionToTest = process.env.NewVersion;
     console.log("BeforeAllowTraffic hook tests started");
     console.log("Testing new function version: " + functionToTest);

     // Create parameters to pass to the updated Lambda function that
     // include the newly added "time" option. If the function did not
     // update, then the "time" option is invalid and function returns
     // a statusCode of 400 indicating it failed.
     var lambdaParams = {
      FunctionName: functionToTest,
      Payload: "{\"option\": \"time\"}",
      InvocationType: "RequestResponse"
```

```
    };

    var lambdaResult = "Failed";
    // Invoke the updated Lambda function.
    lambda.invoke(lambdaParams, function(err, data) {
     if (err){ // an error occurred
      console.log(err, err.stack);
      lambdaResult = "Failed";
     }
     else{ // successful response
      var result = JSON.parse(data.Payload);
      console.log("Result: " +  JSON.stringify(result));
         console.log("statusCode: " + result.statusCode);

         // Check if the status code returned by the updated
         // function is 400. If it is, then it failed. If
         // is not, then it succeeded.
      if (result.statusCode != "400"){
           console.log("Validation succeeded");
       lambdaResult = "Succeeded";
         }
         else {
           console.log("Validation failed");
         }

      // Complete the PreTraffic Hook by sending CodeDeploy the validation status
      var params = {
       deploymentId: deploymentId,
       lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
       status: lambdaResult // status can be 'Succeeded' or 'Failed'
      };

      // Pass CodeDeploy the prepared validation test results.
      codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
       if (err) {
        // Validation failed.
        console.log("CodeDeploy Status update failed");
        console.log(err, err.stack);
        callback("CodeDeploy Status update failed");
       } else {
        // Validation succeeded.
        console.log("CodeDeploy status updated successfully");
        callback(null, "CodeDeploy status updated successfully");
```

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

AfterAllowTraffic Lambda 함수에 대한 파일 생성

`afterAllowTraffic` 후크 Lambda 함수에 대한 파일을 생성합니다.

1. 텍스트 파일을 생성하고 `SAM-Tutorial` 디렉터리에 `afterAllowTraffic.js`로 저장합니다.
2. 다음 Node.js 코드를 `afterAllowTraffic.js`에 복사합니다. 이 함수는 배포의 `AfterAllowTraffic` 후크 중에 실행됩니다.

```
'use strict';

    const AWS = require('aws-sdk');
    const codedeploy = new AWS.CodeDeploy({apiVersion: '2014-10-06'});
    var lambda = new AWS.Lambda();

    exports.handler = (event, context, callback) => {

     console.log("Entering PostTraffic Hook!");

     // Read the DeploymentId and LifecycleEventHookExecutionId from the event
 payload
      var deploymentId = event.DeploymentId;
     var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

     var functionToTest = process.env.NewVersion;
     console.log("AfterAllowTraffic hook tests started");
     console.log("Testing new function version: " + functionToTest);

     // Create parameters to pass to the updated Lambda function that
     // include the original "date" parameter. If the function did not
     // update as expected, then the "date" option might be invalid. If
     // the parameter is invalid, the function returns
     // a statusCode of 400 indicating it failed.
     var lambdaParams = {
      FunctionName: functionToTest,
      Payload: "{\"option\": \"date\", \"period\": \"today\"}",
      InvocationType: "RequestResponse"
```

```
    };

    var lambdaResult = "Failed";
    // Invoke the updated Lambda function.
    lambda.invoke(lambdaParams, function(err, data) {
     if (err){ // an error occurred
      console.log(err, err.stack);
      lambdaResult = "Failed";
     }
     else{ // successful response
      var result = JSON.parse(data.Payload);
      console.log("Result: " +  JSON.stringify(result));
         console.log("statusCode: " + result.statusCode);

         // Check if the status code returned by the updated
         // function is 400. If it is, then it failed. If
         // is not, then it succeeded.
      if (result.statusCode != "400"){
           console.log("Validation of time parameter succeeded");
       lambdaResult = "Succeeded";
         }
         else {
           console.log("Validation failed");
         }

      // Complete the PostTraffic Hook by sending CodeDeploy the validation status
      var params = {
       deploymentId: deploymentId,
       lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
       status: lambdaResult // status can be 'Succeeded' or 'Failed'
      };

      // Pass CodeDeploy the prepared validation test results.
      codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data)
{
       if (err) {
        // Validation failed.
        console.log("CodeDeploy Status update failed");
        console.log(err, err.stack);
        callback("CodeDeploy Status update failed");
       } else {
        // Validation succeeded.
        console.log("CodeDeploy status updated successfully");
        callback(null, "CodeDeploy status updated successfully");
```

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

AWS SAM 애플리케이션 패키징

이제 SAM-Tutorial 디렉터리에는 다음과 같은 네 개의 파일이 있습니다.

- beforeAllowTraffic.js
- afterAllowTraffic.js
- myDateTimeFunction.js
- template.yml

이제 AWS SAM sam package 명령을 사용하여 Lambda 함수 및 CodeDeploy 애플리케이션에 대한 아티팩트를 생성하고 패키징할 준비가 되었습니다. 아티팩트는 S3 버킷에 업로드됩니다. 명령의 출력은 package.yml이라는 새 파일입니다. 이 파일은 다음 단계에서 AWS SAM sam deploy 명령에 사용됩니다.

> **Note**
>
> sam package 명령에 대한 자세한 내용은AWS Serverless Application Model 개발자 가이드의 AWS SAM CLI 명령 참조를 참조하세요.

SAM-Tutorial 디렉터리에서 다음을 실행합니다.

```
sam package \
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket amzn-s3-demo-bucket
```

s3-bucket 파라미터의 경우 이 자습서에 대한 사전 조건으로 생성한 Amazon S3 버킷을 지정합니다. 는 AWS SAM sam deploy 명령에서 사용하는 새 파일의 이름을 output-template-file 지정합니다.

AWS SAM 애플리케이션 배포

`package.yml` 파일과 함께 AWS SAM sam deploy 명령을 사용하여를 사용하여 Lambda 함수와 CodeDeploy 애플리케이션 및 배포 그룹을 생성합니다 CloudFormation.

> **Note**
>
> sam deploy 명령에 대한 자세한 내용은 AWS Serverless Application Model 개발자 가이드의 AWS SAM CLI 명령 참조를 참조하세요.

`SAM-Tutorial` 디렉터리에서 다음 명령을 실행합니다.

```
sam deploy \
  --template-file package.yml \
  --stack-name my-date-time-app \
  --capabilities CAPABILITY_IAM
```

`--capabilities CAPABILITY_IAM` 파라미터는 CloudFormation 을(를) 인증하여 IAM 역할을 생성하는 데 필요합니다.

(선택 사항) 인프라 검사 및 테스트

이 주제에서는 인프라 구성 요소를 보고 Lambda 함수를 테스트하는 방법을 보여 줍니다.

`sam deploy`를 실행한 후 스택의 결과를 보려면

1. https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.
2. 탐색 창에서 스택을 선택합니다. `my-date-time-app` 스택은 맨 위에 나타납니다.
3. Events(이벤트) 탭을 선택하여 어떤 이벤트가 완료되었는지를 확인합니다. 스택 생성이 진행 중인 동안 이벤트를 볼 수 있습니다. 스택 생성이 완료되면 모든 스택 생성 이벤트를 볼 수 있습니다.
4. 스택이 선택된 상태에서 Resources(리소스)를 선택합니다. 유형 열에서 Lambda 함수인 `myDateTimeFunction`, `CodeDeployHook_beforeAllowTraffic`, `CodeDeployHook_afterAllowTraffic`을(를) 볼 수 있습니다. 각 Lambda 함수의 물리적 ID 열에는 Lambda 콘솔에서 함수를 볼 수 있는 링크가 포함되어 있습니다.

> **Note**
>
> myDateTimeFunction Lambda 함수의 이름 앞에 스택의 이름이 추가되고 CloudFormation 여기에 식별자가 추가되므로 처럼 보입니다my-date-time-app-myDateTimeFunction-123456ABCDEF.

5. https://console.aws.amazon.com/codedeploy/에서 CodeDeploy 콘솔을 엽니다.
6. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
7. 이름이 로 시작하는 CloudFormation 에서 생성된 새 CodeDeploy 애플리케이션이 표시됩니다my-date-time-app-ServerlessDeploymentApplication. 이 애플리케이션을 선택합니다.
8. my-date-time-app-myDateTimeFunctionDeploymentGroup으로 시작하는 이름의 배포 그룹이 보여야 합니다. 이 배포 그룹을 선택합니다.

   Deployment configuration(배포 구성) 아래에 CodeDeployDefault.LambdaLinear10PercentEvery1Minute가 보여야 합니다.

(선택 사항) 함수를 테스트하려면(콘솔)

1. https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 탐색 창에서 my-date-time-app-myDateTimeFunction 함수를 선택합니다. 콘솔에서는 이름에 ID가 포함되므로 my-date-time-app-myDateTimeFunction-123456ABCDEF와 비슷하게 보입니다.
3. 테스트를 선택합니다.
4. Event name(이벤트 이름)에 테스트 이벤트의 이름을 입력합니다.
5. 테스트 이벤트에 대해 다음을 입력한 다음 Create(생성)을 선택합니다.

```
{
  "option": "date",
  "period": "today"
}
```

6. 테스트를 선택합니다. 테스트 이벤트 목록에서 테스트 이벤트만 보여야 합니다.

   Execution result(실행 결과)의 경우 succeeded(성공)가 보여야 합니다.
7. Execution result(실행 결과)에서 Details(세부 정보)를 확장하여 결과를 봅니다. 현재 월, 일 및 연도가 보여야 합니다.

(선택 사항) 함수(AWS CLI)를 테스트하려면

1. Lambda 함수의 ARN을 찾습니다. 함수를 볼 때 Lambda 콘솔 맨 위에 나타납니다.
2. 다음 명령을 실행합니다. *your-function-arn*을 함수ARN으로 바꿉니다.

```
aws lambda invoke \
--function your-function-arn \
--cli-binary-format raw-in-base64-out \
--payload "{\"option\": \"date\", \"period\": \"today\"}" out.txt
```

3. `out.txt`를 열고 결과에 현재 월, 일, 연도가 포함되어 있는지 확인합니다.

2단계: Lambda 함수 업데이트

이 주제에서는 `myDateTimeFunction.js` 파일을 업데이트합니다. 다음 단계에서는 이 파일을 사용하여 업데이트된 함수를 배포합니다. 이 함수는 Lambda 함수의 현재 버전에서 업데이트된 버전으로 프로덕션 트래픽을 이동하여 함수를 배포하도록 CodeDeploy를 트리거합니다.

Lambda 함수를 업데이트하려면

1. `myDateTimeFunction.js`을(를) 엽니다.
2. `switch` 블록에 있는 `time`이라는 `case`의 시작과 끝에서 두 개의 주석 마커("/*" 및 "*/")와 설명 텍스트를 제거합니다.

   주석 처리가 해제된 코드를 사용하면 새 파라미터인 `time`을 함수에 전달할 수 있습니다. `time`을 업데이트된 함수에 전달하면 이 함수는 현재 `hour`, `minute` 및 `second`를 반환합니다.
3. `myDateTimeFunction.js`을(를) 저장합니다. 이 템플릿은 다음과 같습니다.

```
'use strict';

exports.handler = function(event, context, callback) {

  if (event.body) {
    event = JSON.parse(event.body);
  }

  var sc; // Status code
  var result = ""; // Response payload

  switch(event.option) {
```

```
    case "date":
      switch(event.period) {
        case "yesterday":
          result = setDateResult("yesterday");
          sc = 200;
          break;
        case "today":
          result = setDateResult();
          sc = 200;
          break;
        case "tomorrow":
          result = setDateResult("tomorrow");
          sc = 200;
          break;
        default:
          result = {
            "error": "Must specify 'yesterday', 'today', or 'tomorrow'."
          };
          sc = 400;
          break;
      }
      break;
      case "time":
        var d = new Date();
        var h = d.getHours();
        var mi = d.getMinutes();
        var s = d.getSeconds();

        result = {
          "hour": h,
          "minute": mi,
          "second": s
        };
        sc = 200;
        break;

      default:
        result = {
          "error": "Must specify 'date' or 'time'."
        };
        sc = 400;
      break;
  }
```

```
  const response = {
    statusCode: sc,
    headers: { "Content-type": "application/json" },
    body: JSON.stringify( result )
  };

  callback(null, response);

  function setDateResult(option) {

    var d = new Date(); // Today
    var mo; // Month
    var da; // Day
    var y; // Year

    switch(option) {
      case "yesterday":
        d.setDate(d.getDate() - 1);
        break;
      case "tomorrow":
        d.setDate(d.getDate() + 1);
      default:
       break;
    }

    mo = d.getMonth() + 1; // Months are zero offset (0-11)
    da = d.getDate();
    y = d.getFullYear();

    result = {
      "month": mo,
      "day": da,
      "year": y
    };

    return result;
  }
};
```

3단계: 업데이트된 Lambda 함수 배포

이 단계에서는 업데이트된 `myDateTimeFunction.js`을(를) 사용하여 Lambda 함수의 배포를 업데이트하고 시작합니다. CodeDeploy 또는 AWS Lambda 콘솔에서 배포 진행 상황을 모니터링할 수 있습니다.

AWS SAM 템플릿의 `AutoPublishAlias: live` 줄을 사용하면 인프라가 `live`별칭을 사용하는 함수에 대한 업데이트를 감지합니다. 함수를 업데이트하면 함수의 원래 버전에서 업데이트된 버전으로 프로덕션 트래픽을 이동하는 CodeDeploy를 통해 배포가 트리거됩니다.

sam package 및 sam deploy 명령은 Lambda 함수의 배포를 업데이트하고 트리거하는 데 사용됩니다. AWS SAM 애플리케이션 패키징 및 AWS SAM 애플리케이션 배포에서 이러한 명령을 실행했습니다

업데이트된 Lambda 함수를 배포하려면

1. `SAM-Tutorial` 디렉터리에서 다음 명령을 실행합니다.

   ```
   sam package \
     --template-file template.yml \
     --output-template-file package.yml  \
     --s3-bucket amzn-s3-demo-bucket
   ```

   그러면 S3 버킷에서 업데이트된 Lambda 함수를 참조하는 새로운 아티팩트 세트가 생성됩니다.

2. `SAM-Tutorial` 디렉터리에서 다음 명령을 실행합니다.

   ```
   sam deploy \
     --template-file package.yml \
     --stack-name my-date-time-app \
     --capabilities CAPABILITY_IAM
   ```

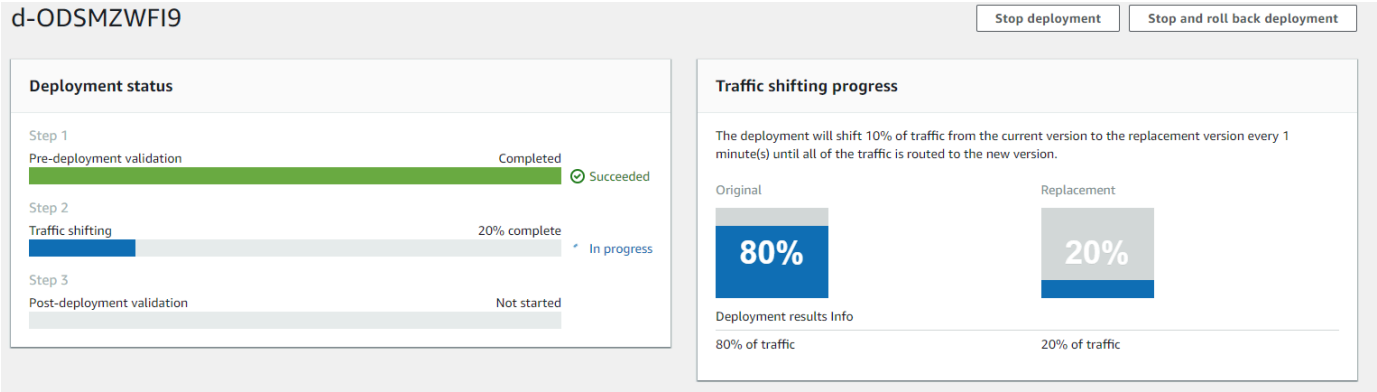
   스택 이름은 여전히 이므로는 스택 업데이트임을 `my-date-time-app` CloudFormation 인식합니다. 업데이트된 스택을 보려면 CloudFormation 콘솔을 반환하고 탐색 창에서 스택을 선택합니다.

(선택 사항) 배포 중에 트래픽을 보려면(CodeDeploy 콘솔)

1. https://console.aws.amazon.com/codedeploy/에서 CodeDeploy 콘솔을 엽니다.
2. 탐색 창에서 Applications(애플리케이션)를 확장한 다음 my-date-time-app-ServerlessDeploymentApplication 애플리케이션을 선택합니다.

3. Deployment groups(배포 그룹)에서 애플리케이션의 배포 그룹을 선택합니다. 상태는 In progress(진행 중)여야 합니다.
4. Deployment group history(배포 그룹 기록)에서 진행 중인 배포를 선택합니다.

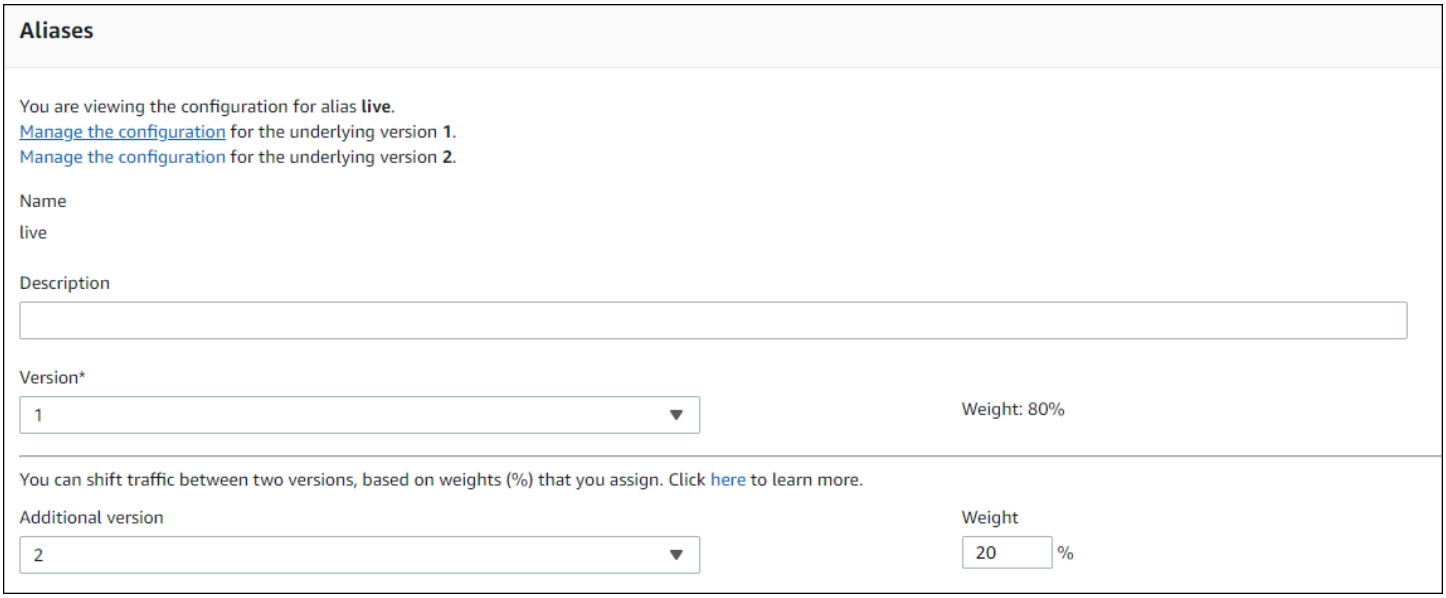
   트래픽 이동 진행률 표시줄과 이 페이지의 원본 및 대체 상자에 진행 상황이 표시됩니다.



(선택 사항) 배포 중에 트래픽을 보려면(Lambda 콘솔)

1. https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 탐색 창에서 `my-date-time-app-myDateTimeFunction` 함수를 선택합니다. 콘솔에서는 이름에 ID가 포함되므로 `my-date-time-app-myDateTimeFunction-123456ABCDEF`와 비슷하게 보입니다.
3. 별칭을 선택한 다음 라이브를 선택합니다.

원래 함수 버전(버전 1)과 업데이트된 함수 버전(버전 2) 옆에 있는 가중치는 이 AWS Lambda 콘솔 페이지가 로드될 때 얼마나 많은 트래픽이 각 버전에 제공되는지를 표시합니다. 이 페이지에서 가중치는 시간 경과에 따라 업데이트되지 않습니다. 1분에 한 번 페이지를 새로 고치면 버전 1의 가중치는 10%씩 감소하고 버전 2의 가중치가 100이 될 때까지 버전 2의 가중치는 10%씩 증가합니다.



4단계: 배포 결과 보기

이 단계에서는 배포의 결과를 봅니다. 배포가 성공하면 업데이트된 Lambda 함수가 프로덕션 트래픽을 수신하는 것을 확인할 수 있습니다. 배포가 실패하면 CloudWatch Logs를 사용하여 배포의 수명 주기 후크 중에 실행된 Lambda 함수에서 확인 테스트의 출력을 볼 수 있습니다.

주제

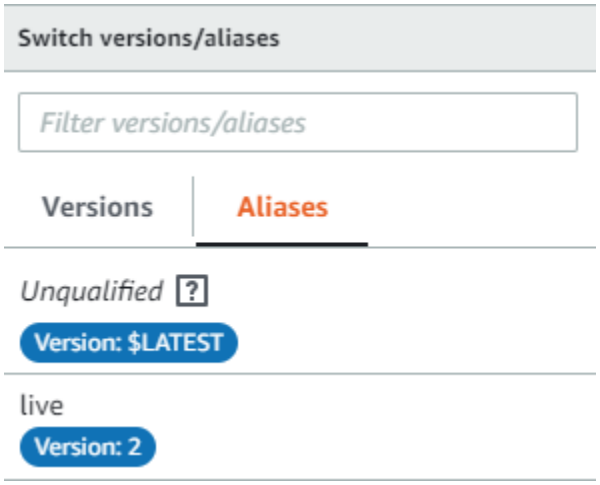
- 배포된 함수 테스트
- CloudWatch Logs에서 후크 이벤트 보기

배포된 함수 테스트

sam deploy 명령은 `my-date-time-app-myDateTimeFunction` Lambda 함수를 업데이트합니다. 함수 버전은 2로 업데이트되고 `live` 별칭에 추가됩니다.

Lambda 콘솔에서 업데이트를 보려면

1. https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.
2. 탐색 창에서 `my-date-time-app-myDateTimeFunction` 함수를 선택합니다. 콘솔에서는 이름에 ID가 포함되므로 `my-date-time-app-myDateTimeFunction-123456ABCDEF`와 비슷하게 보입니다.
3. Qualifiers(한정자)를 선택한 다음 Aliases(별칭)를 선택합니다. 배포가 완료된(약 10분) 후, `live` 별칭에서 Version: 2(버전 2)가 보여야 합니다.



4. Function code(함수 코드)에서 함수의 소스 코드를 봅니다. 변경 내용이 나타나야 합니다.
5. (선택 사항) 2단계: Lambda 함수 업데이트의 테스트 명령을 사용하여 업데이트된 함수를 테스트할 수 있습니다. 다음 페이로드가 있는 새 테스트 이벤트를 생성한 다음, 결과에 현재 시간, 분 및 초가 포함되어 있는지 확인합니다.

```
{
    "option": "time"
  }
```

AWS CLI 를 사용하여 업데이트된 함수를 테스트하려면 다음 명령을 실행한 다음 `out.txt`를 열어 결과에 현재 시간, 분 및 초가 포함되어 있는지 확인합니다.

```
aws lambda invoke --function your-function-arn --payload "{\"option\": \"time\"}"
 out.txt
```

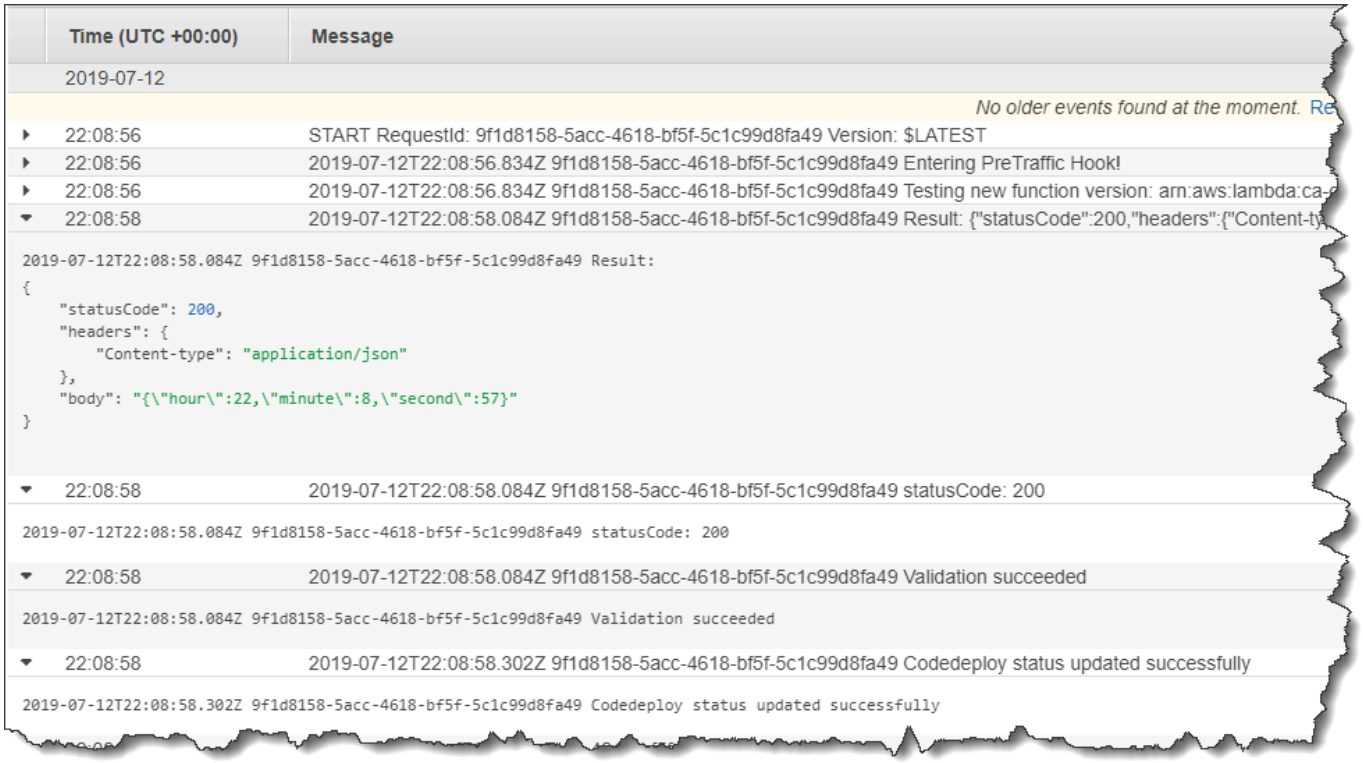
> **Note**
>
> 배포가 완료되기 전에 AWS CLI 를 사용하여 함수를 테스트하면 예상치 못한 결과가 발생할 수 있습니다. 이 결과는 CodeDeploy가 1분마다 트래픽의 10%를 업데이트된 버전으로 증분식으로 이동하기 때문입니다. 배포 중에 일부 트래픽은 여전히 원래 버전을 가리키므로, `aws lambda invoke`가 원래 버전을 사용할 수 있습니다. 10분 후에 배포가 완료되고 모든 트래픽이 함수의 새 버전을 가리킵니다.

CloudWatch Logs에서 후크 이벤트 보기

`BeforeAllowTraffic` 후크 중에 CodeDeploy는 `CodeDeployHook_beforeAllowTraffic` Lambda 함수를 실행합니다. `AfterAllowTraffic` 후크 중에 CodeDeploy는 `CodeDeployHook_afterAllowTraffic` Lambda 함수를 실행합니다. 각 함수는 새 `time` 파라미터를 사용하여 함수의 업데이트된 버전을 호출하는 평가 테스트를 실행합니다. Lambda 함수의 업데이트가 성공하면 `time` 옵션으로 인해 오류가 발생하지 않고 확인이 성공합니다. 함수가 업데이트되지 않은 경우 인식되지 않는 파라미터로 인해 오류가 발생하고 확인이 실패합니다. 이러한 확인 테스트는 데모용일 뿐입니다. 각자 고유의 테스트를 작성하여 배포를 확인합니다. CloudWatch Logs 콘솔을 사용하여 확인 테스트를 볼 수 있습니다.

CodeDeploy 후크 이벤트를 보려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택합니다.
3. 로그 그룹 목록에서 /aws/lambda/CodeDeployHook_beforeAllowTraffic 또는 /aws/lambda/CodeDeployHook_afterAllowTraffic을 선택합니다.
4. 로그 스트림을 선택합니다. 하나만 보여야 합니다.
5. 이벤트를 확장하여 세부 정보를 확인합니다.



5단계: 정리

이 자습서에서 사용한 리소스에 대한 추가 요금이 부과되지 않도록 AWS SAM 템플릿에서 생성한 리소스와 Lambda 검증 함수에서 생성한 CloudWatch 로그를 삭제합니다.

CloudFormation 스택을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.
2. Stacks(스택) 열에서 `my-date-time-app` 스택을 선택한 다음 Delete(삭제)를 선택합니다.
3. 메시지가 나타나면 스택 삭제를 선택합니다. Lambda 함수, CodeDeploy 애플리케이션 및 배포 그룹,에서 생성한 IAM 역할 AWS SAM 이 삭제됩니다.

CloudWatch Logs에서 로그를 삭제하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 로그를 선택합니다.
3. 로그 그룹 목록에서 /aws/lambda/CodeDeployHook_beforeAllowTraffic 옆의 버튼을 선택합니다.
4. Actions(작업)에서 Delete log group(로그 그룹 삭제)을 선택한 다음 Yes, Delete(예, 삭제합니다)를 선택합니다.
5. 로그 그룹 목록에서 /aws/lambda/CodeDeployHook_afterAllowTraffic 옆의 버튼을 선택합니다.
6. Actions(작업)에서 Delete log group(로그 그룹 삭제)을 선택한 다음 Yes, Delete(예, 삭제합니다)를 선택합니다.

CodeDeploy 에이전트 작업

AWS CodeDeploy 에이전트는 인스턴스에 설치 및 구성하면 해당 인스턴스를 CodeDeploy 배포에 사용할 수 있는 소프트웨어 패키지입니다.

AWS 는 CodeDeploy 에이전트의 최신 마이너 버전을 적극적으로 지원합니다. 최신 릴리스 버전은 섹션을 참조하세요CodeDeploy 에이전트의 버전 기록.

> **Note**
>
> CodeDeploy 에이전트는 EC2/온프레미스 컴퓨팅 플랫폼에 배포하는 경우에만 필요합니다. Amazon ECS 또는 AWS Lambda 컴퓨팅 플랫폼을 사용하는 배포에는 에이전트가 필요하지 않습니다.

에이전트가 설치되면 구성 파일은 인스턴스에 있습니다. 구성 파일은 에이전트 작동 방식을 지정하는 데 사용되며 이 구성 파일은가 인스턴스와 상호 작용 AWS CodeDeploy 할 때 사용할 디렉터리 경로 및 기타 설정을 지정합니다. 이 파일의 일부 구성 옵션은 변경할 수 있습니다. CodeDeploy 에이전트 구성 작업에 대한 자세한 내용은 CodeDeploy 에이전트 구성 참조 단원을 참조하세요.

CodeDeploy 에이전트 작업(예: 설치, 업데이트 및 버전 확인 단계 등)에 대한 자세한 내용은 CodeDeploy 에이전트 작업 관리 단원을 참조하세요.

주제

- CodeDeploy 에이전트에서 지원하는 운영 체제
- CodeDeploy 에이전트용 통신 프로토콜 및 포트
- CodeDeploy 에이전트의 버전 기록
- CodeDeploy 프로세스 관리
- 애플리케이션 수정 버전 및 로그 파일 정리
- CodeDeploy 에이전트에 의해 설치된 파일
- CodeDeploy 에이전트 작업 관리

CodeDeploy 에이전트에서 지원하는 운영 체제

지원되는 Amazon EC2 AMI 운영 체제

CodeDeplo 에이전트는 다음 Amazon EC2 AMI 운영 체제에서 테스트되었습니다.

- Amazon Linux 2023(x86_64, aarch64)
- Amazon Linux 2(x86_64, aarch64)
- Red Hat Enterprise Linux(RHEL) 8, 9, 10(x86_64, aarch64)
- Oracle Linux 8, 9, 10(x86_64, aarch64)
- Rocky Linux 9, 10(x86_64, aarch64)
- CentOS 스트림 9, 10(x86_64, aarch64)
- SLES 15(x86_64, aarch64)
- Debian 11, 12, 13(x86_64, aarch64)
- Ubuntu Server 16.04 LTS, 18.04 LTS, 20.04 LTS, 22.04 LTS, 24.04 LTS, 25.04, 26.04 LTS(x86_64, aarch64)
- Windows Server 2016, 2019, 2022, 2025(x86_64)
- Apple 실리콘을 사용하는 EC2 Mac 인스턴스의 macOS 15, 26, 27(aarch64)

> **Note**
>
> CodeDeploy 에이전트 버전 2.0.x 이상에서만 테스트된 플랫폼은 Windows Server 2025, RHEL 10, Oracle Linux 8, 9, 10, Rocky Linux 9, 10, CentOS Stream 9, 10, SLES 15, Debian 11, 12, 13, Ubuntu Server 24.04, 25.04, 26.04입니다.
> macOS는 CodeDeploy 에이전트 버전 2.1.2 이상에서만 지원됩니다. 에이전트는 Apple 실리콘에서 macOS 15, 26, 27을 지원합니다.

CodeDeploy 에이전트는 필요에 따라 조정할 수 있도록 오픈 소스로 제공됩니다. 다른 Amazon EC2 AMI 운영 체제에서도 사용할 수 있습니다. 자세한 내용을 확인하려면 GitHub에서 CodeDeploy 에이전트 리포지토리로 이동하세요.

지원되는 온프레미스 운영 체제

CodeDeploy 에이전트는 다음 온프레미스 운영 체제에서 테스트되었습니다.

- Windows Server 2016, 2019, 2022, 2025(x86_64)
- Red Hat Enterprise Linux(RHEL) 8, 9, 10(x86_64, aarch64)
- Oracle Linux 8, 9, 10(x86_64, aarch64)
- Rocky Linux 9, 10(x86_64, aarch64)
- CentOS 스트림 9, 10(x86_64, aarch64)
- SLES 15(x86_64, aarch64)
- Debian 11, 12, 13(x86_64, aarch64)
- Ubuntu Server 16.04 LTS, 18.04 LTS, 20.04 LTS, 22.04 LTS, 24.04 LTS, 25.04, 26.04 LTS(x86_64, aarch64)
- Apple 실리콘을 사용하는 macOS 15, 26, 27(aarch64)

> **Note**
>
> CodeDeploy 에이전트 버전 2.0.x 이상에서만 테스트된 플랫폼은 Windows Server 2025, RHEL 10, Oracle Linux 8, 9, 10, Rocky Linux 9, 10, CentOS Stream 9, 10, SLES 15, Debian 11, 12, 13, Ubuntu Server 24.04, 25.04, 26.04입니다.
> macOS는 CodeDeploy 에이전트 버전 2.1.2 이상에서만 지원됩니다. 에이전트는 Apple 실리콘에서 macOS 15, 26, 27을 지원합니다.

CodeDeploy 에이전트는 필요에 따라 조정할 수 있도록 오픈 소스로 제공됩니다. 다른 온프레미스 운영 체제에서도 사용할 수 있습니다. 자세한 내용을 확인하려면 GitHub에서 CodeDeploy 에이전트 리포지토리로 이동하세요.

CodeDeploy 에이전트용 통신 프로토콜 및 포트

CodeDeploy 에이전트는 포트 443을 통해 HTTPS를 사용하여 아웃바운드 통신을 수행합니다.

CodeDeploy 에이전트는 EC2 인스턴스에서 실행될 때 EC2 메타데이터 엔드포인트를 사용하여 인스턴스 관련 정보를 검색합니다. 인스턴스 메타데이터 서비스 액세스 제한 및 부여에 대해 자세히 알아보세요.

CodeDeploy 에이전트의 버전 기록

인스턴스에서는 지원되는 CodeDeploy 에이전트 버전을 실행 중이어야 합니다. 현재 지원되는 최소 버전은 2.1.x입니다.

> **Note**
>
> CodeDeploy 에이전트는 최신 버전을 사용하는 것이 좋습니다. 문제가 있는 경우 AWS Support에 문의하기 전에 최신 버전으로 업데이트하세요. 업그레이드 정보는 CodeDeploy 에이전트 업데이트 섹션을 참조하세요.

> **버전 2.x 롤아웃 상태**
>
> CodeDeploy 에이전트 버전 2.x가 롤아웃되고 AWS 리전 있으며 일부 리전에서는 아직 사용할 수 없습니다. 버전 2.x는 옵트인 업그레이드입니다. 이전 개정에서 버전 2.x로 업그레이드하려면 섹션을 참조하세요CodeDeploy 에이전트 설치. 설치할 수 있는 버전을 알아보려면 섹션을 참조하세요에서 사용 가능한 최신 에이전트 버전 확인 AWS 리전.

다음 표에는 CodeDeploy 에이전트의 모든 릴리스와 각 릴리스에 포함된 기능 및 기능 향상이 나와 있습니다.

버전	릴리스 날짜	세부 정보
2.1.2	2026년 10월 13일	추가됨: Apple 실리콘에서 macOS 15, 26, 27 지원. 지침은 macOS용 CodeDeploy 에이전트 설치 섹션을 참조하세요. 추가됨: Amazon S3에서 애플리케이션 개정의 병렬 범위 다운로드, 기본적으로 꺼짐. 자세한 내용은 CodeDeploy 에이전트 구성 참조 섹션을 참조하세요. 수정됨: 수명 주기 이벤트 스크립트 출력은 `codedeploy-agent-deployments.log` 배포 ID 접두사가 붙은 줄별로 작성됩니다.
2.1.1	2026년 10월 2일	수정됨: 이제 에이전트가 AWS 유럽 서버린 클라우드의 올바른 CodeDeploy 엔드포인트에 연결됩니다.

버전	릴리스 날짜	세부 정보
2.1.0	2026년 9월 7일	추가됨: RESTART 배포 모드에 대한 기본 지원으로, 배포 그룹의 마지막으로 성공한 배포의 개정을 사용하여 대상 인스턴스에서 애플리케이션을 다시 시작합니다. 에이전트는 이미 인스턴스에 있는 개정의 복사본에서를 설치합니다. 복사본이 누락되거나 불완전한 경우 에이전트는 개정을 다운로드하므로 배포가 계속 완료됩니다. 배포 모드 자체는 AWS CodeDeploy API 참조의 deploymentMode를 참조하세요. 변경됨(보안): 이제 에이전트가 AppSpec 파일 경로가 애플리케이션 개정 디렉터리 외부에서 확인되는 배포를 거부합니다. 와 같이 개정 내에 중첩된 경로`configs/appspec.yml` 는 여전히 지원됩니다.
2.0.1	2026년 8월 24일	수정: Windows에서는 백슬래시(\)로 시작하거나 구분 기호가 두 개 이상인 AppSpec 파일 경로(예: `//my-folder`)가 이제 드라이브의 루트가 아닌 개정 내에서 확인됩니다. 지원되는 `source` 값의 예는 섹션을 참조하세요AppSpec 'files' 섹션(EC2/온프레미스 배포만 해당).

버전	릴리스 날짜	세부 정보
2.0.0	2026년 7월 15일	변경됨: CodeDeploy 에이전트는 이제 Rust 기반이며 단일 독립형 네이티브 바이너리로 제공됩니다. Ruby 런타임 종속성이 삭제되었습니다. Ruby는 더 이상 에이전트를 설치하거나 실행할 필요가 없습니다. 에이전트는 기본적으로 버전 1.8.x와 동작 호환되며, 아래에 명시되지 않는 한 기존 구성 파일, AppSpec 처리, 수명 주기 후크 의미 체계, 온디스크 레이아웃 및 배포 결과는 변경되지 않습니다. 추가됨: Windows Server 2025, RHEL 10 및 Ubuntu Server 26.04를 비롯한 운영 체제 지원과 추가 Linux 배포 및 ARM(aarch64) 변형이 확장되었습니다. 이 버전으로 테스트된 플랫폼의 전체 목록은 섹션을 참조하세요CodeDeploy 에이전트에서 지원하는 운영 체제. 추가됨: 온디맨드 에이전트 자체 업데이트를 위한 `codedeploy-agent update` 명령입니다. 추가됨: 로컬 명령 포트. 이는 에이전트 상태를 쿼리하고 명령을 로컬로 주입하기 위해 루프백 주소(127.0.0.1)에서만 수신하는 선택적 TCP 관리 인터페이스입니다. 기본적으로 비활성화되어 있으며에서 활성화됩니다`enable_command_port` . 변경됨: 로컬 배포 명령이 이제 입니다`codedeploy-agent deploy-local` . 이전 버전과의 호환성을 위해 `codedeploy-local` 명령을 계속 사용할 수 있습니다. 모든 명령 옵션이 보존되며 로컬 경로, Amazon S3(`s3://`) 및 GitHub 번들 소스는 여전히 지원됩니다. 변경됨: Linux에서 에이전트는 기본 시스템 단위(`codedeploy-agent.service`)로 관리되며, SysV init.d 스크립트는 더 이상 설치되지 않습니다. `systemctl` 를 사용하여 에이전트를 관리합니다. 변경됨(보안): TLS 인증서 확인은 항상 활성화되어 있으며 코어 덤프는 기본적으로 비활성화되어 있습니다.

버전	릴리스 날짜	세부 정보
		추가됨: 자체 보안 요구 사항을 충족하기 위해 활성화할 수 있는 선택적 강화 설정입니다. 각 설정은 기본적으로 꺼져 있으며 원하는 설정을 개별적으로 활성화합니다. 심볼 링크, 경로 순회 또는 안전하지 않은 권한이 포함된 번들을 거부하고, 수명 주기 이벤트 후크의 환경을 제한하고, 에이전트 디렉터리 및 로그 파일의 권한을 제한할 수 있습니다. 전체 목록은 CodeDeploy 에이전트 구성 참조 단원을 참조하세요. 변경됨: 명령줄 설치 스크립트가 이제 Bash(이전 Ruby)로 작성되었습니다. 또는 `curl`가 필요합니다`wget`. 변경됨: 버전 2.0.x의 설치 스크립트는 리전 `aws-codedeploy-` *`region-identifier`* 버킷의 `latestv2/` 접두사 아래에 게시됩니다. `latest/` 접두사는 버전 1.8.x 설치 스크립트를 계속 제공합니다. Changed: AWS now는 버전 2.0.x를 `AWSCodeDeployAgentV2` Systems Manager Distributor 패키지로 게시합니다. `AWSCodeDeployAgent` 패키지는 버전 1.8.x를 계속 제공합니다. 제거됨: `ssl_verify_peer` 구성 설정 및 Ruby ProcessManager 구성 키입니다. 중요: 현재 버전 1.8.x에서 2.0.0으로 자동 업데이트 경로가 없습니다. 2.0.0으로 업그레이드하려면 `AWSCodeDeployAgentV2` Systems Manager Distributor 패키지를 사용하여 설치하거나 리전 `latestv2/install` 스크립트를 수동으로 실행합니다. 지침은 CodeDeploy 에이전트 설치 섹션을 참조하세요.
1.8.1	2026년 2월 3일	수정: S3 엔드포인트 버그 수정. 추가됨: Windows CA 목록에 Sectigo CA 인증서.
1.8.0	2025년 7월 31일	변경됨: Windows용 CodeDeploy 에이전트에서 번들된 Ruby를 3.2로 업그레이드했습니다.

버전	릴리스 날짜	세부 정보
1.7.1	2024년 11월 14일	변경됨: 보안 패치에 대한 종속성이 업데이트되었습니다.
1.7.0	2024년 3월 6일	추가됨: CodeDeploy 에이전트 구성 파일에 대한 `:disable_imds_v1:` 구성 설정입니다. IMDSv2 오류 발생 시 IMDSv1으로의 대체 실행을 비활성화하려면 이 설정을 사용합니다. 기본값은 `false`입니다(폴백 활성화). 자세한 내용은 CodeDeploy 에이전트 구성 참조를 참조하세요. 추가됨: Red Hat Enterprise Linux 9(RHEL 9) 운영 체제에 대한 지원입니다. 추가됨: Ubuntu Server의 Ruby 버전 3.1 및 3.2에 대한 지원입니다. 수정됨: CodeDeploy 에이전트 구성 파일 로드에 실패할 경우, CodeDeploy 에이전트가 이제 사용자 친화적인 오류를 생성합니다. 변경됨: Windows용 CodeDeploy 에이전트에서 Ruby를 2.7.8-1로 업그레이드했습니다.
1.6.0	2023년 3월 30일	추가됨: Ruby 3.1, 3.2에 대한 지원. 추가됨: Amazon Linux 2023에 대한 지원. 추가됨: Windows Server 2022에 대한 지원. 변경됨: 이제 Windows Server 인스턴스의 기본 `verbose` 설정이 `false`입니다. Windows에서 로그 파일의 디버그 메시지를 계속 출력하려면 `verbose`를 `true`로 설정해야 합니다. 제거됨: Windows Server 2016 및 Windows Server 2012 R2에 대한 지원. 제거됨: Amazon Linux 2018.03.x에 대한 지원.

버전	릴리스 날짜	세부 정보
1.5.0	2023년 3월 3일	추가됨: Ruby 3에 대한 지원. 추가됨: Ubuntu 22.04에 대한 지원. 수정됨: 시작 직후 CodeDeploy 에이전트를 다시 시작하면 에이전트가 중지되는 문제. 변경됨: 이제 후크 스크립트를 실행하는 동안 에이전트 서비스가 예기치 않게 다시 시작되면 CodeDeploy 에이전트가 에이전트 시작 시 호스트 배포에 실패합니다. 이 수정을 통해 배포를 재시도하기 전에 70분의 제한 시간 동안 기다리지 않아도 됩니다. 사용 중단 공지: CodeDeploy 에이전트 1.5.0은 Windows Server 2016 및 Windows Server 2012 R2를 지원하는 마지막 릴리스입니다. 제거됨: Ubuntu 14.04 LTS, Windows Server 2008 R2 및 Windows Server 2008 R2 32비트에서 CodeDeploy 에이전트 지원.
1.4.1	2022년 12월 6일	수정됨: 로깅과 관련된 보안 취약점이 수정되었습니다. 개선 사항: 호스트 명령을 폴링할 때의 로깅이 개선되었습니다.

버전	릴리스 날짜	세부 정보
1.4.0	2022년 8월 31일	추가됨: Hat Enterprise Linux 8 지원 추가: Windows용 CodeDeploy 에이전트에서 긴 파일 경로를 지원합니다. 긴 파일 경로를 활성화하려면 적절한 Windows 레지스트리 키를 설정한 다음 에이전트를 다시 시작해야 합니다. 자세한 내용은 파일 경로가 길면 “No such file or directory(해당 파일 또는 디렉터리가 없음)” 오류가 발생함 섹션을 참조하세요. 해결됨: 디스크가 가득 찼을 때 압축 해제 작업 문제 이제 CodeDeploy 에이전트는 가득 찬 디스크를 나타내는 압축 해제 종료 코드 50을 감지하고 부분적으로 추출된 파일을 제거하며 예외를 발생시켜 CodeDeploy 서버에 오류를 게시합니다. 오류 메시지는 수명 주기 이벤트 오류 메시지로 표시되며 호스트 수준 배포는 중단되거나 제한 시간이 초과되지 않고 중지됩니다. 해결됨: 에이전트에 오류를 일으키는 문제. 해결됨: 엣지 사례 경합 상태에서 후크 시간 초과가 발생하는 문제. 스크립트가 없는 후크는 이제 계속 유지되며 더 이상 실패나 시간 초과를 일으키지 않습니다. 변경됨: CodeDeploy 에이전트 `bin` 디렉터리의 `update` 스크립트는 더 이상 사용되지 않으므로 제거되었습니다. 변경됨: 이제 Windows Server용 CodeDeploy 에이전트에 Ruby 2.7이 번들로 제공됩니다. 변경됨: 배포 번들의 소스(Amazon S3 또는 GitHub)에 따라 후크 스크립트에서 사용할 새 환경 변수가 추가되었습니다. 자세한 내용은 후크의 환경 변수 가용성 섹션을 참조하세요.

버전	릴리스 날짜	세부 정보
		Important 사용 중단 공지: CodeDeploy 에이전트 1.4.0은 32비트 Windows Server용 설치 프로그램을 포함하는 마지막 릴리스입니다. 사용 중단 공지: CodeDeploy 에이전트 1.4.0은 Windows Server 2008 R2를 지원하는 마지막 릴리스입니다. 제거됨: 다음 Amazon EC2 AMI에서 CodeDeploy 에이전트 지원: Amazon Linux 2014.09, 2016.03, 2016.09 및 2017.03.

버전	릴리스 날짜	세부 정보
1.3.2	2021년 5월 6일	**Important** CodeDeploy 에이전트 1.3.2는 CVE-2018-1000201를 처리하며, 이는 해당 에이전트를 실행하는 Windows 호스트에 영향을 줍니다. CVE는 CodeDeploy 에이전트에 종속적인 Ruby-ffi를 인용합니다. 에이전트가 Amazon EC2 Systems Manager(SSM)와 함께 설치되었고 자동으로 업데이트되도록 설정된 경우 별도의 작업이 필요하지 않습니다. 그렇지 않으면 에이전트를 수동으로 업데이트하는 작업이 필요합니다. 에이전트를 업그레이드하려면 Windows Server에서 CodeDeploy 에이전트 업데이트의 지침을 따릅니다. 수정됨: Ubuntu 20.04 이상에서 CodeDeploy 에이전트를 설치할 때 발생하는 문제. 수정됨: 상대 경로가 올바르게 처리되지 않았기 때문에 압축된 파일을 추출할 때 간헐적으로 발생하는 문제. 추가됨: Windows 인스턴스의 AWS PrivateLink 및 VPC 엔드포인트 지원. 추가됨: AppSpec 파일 개선 사항(아래에 설명된대로). • 이제 로컬 배포를 만들 때 AppSpec 파일에 대한 사용자 정의 파일 이름을 지정할 수 있습니다. 자세한 내용은 로컬 배포 생성 섹션을 참조하세요. • 이제 AppSpec 파일이 `.yaml` 파일 확장명을 가집니다. • 이제 새로운 `file_exists_behavior` 설정 옵션을 사용하여 배포된 파일을 AppSpec 파일에서 덮어씁니다. 자세한 내용은 AppSpec 'files' 섹션(EC2/온프레미스 배포만 해당) 섹션을 참조하세요.

버전	릴리스 날짜	세부 정보
		업그레이드됨: 이제 CodeDeploy는 AWS SDK for Ruby 3.0을 사용합니다.
1.3.1	2020년 12월 22일	수정됨: 온프레미스 인스턴스가 시작되지 않도록 하는 1.3.0의 문제.
1.3.0	2020년 11월 10일	**Important** 이 버전은 더 이상 사용되지 않습니다. 수정됨: 더 이상 사용되지 않는 만료된 인증서를 제거했습니다. 수정됨:에서 사용하는 에이전트 제거 스크립트에서 프롬프트 메시지를 제거하여 호스트 또는 플릿을 이전 버전의 에이전트로 더 쉽게 다운그레이드할 수 AWS Systems Manager있도록 했습니다.
1.2.1	2020년 9월 23일	변경됨: v2에서 v3로의 AWS SDK for Ruby 종속성이 업그레이드되었습니다. 추가됨: IMDS V2에 대한 지원. IMDSv2 HTTP 요청이 실패할 경우 IMDSv1에 대한 자동 대체 기능을 포함합니다. 변경됨: 보안 패치에 대한 Rake 및 Rubyzip 종속성이 업데이트되었습니다. 수정됨: 빈 PID 파일이 `No CodeDeploy Agent Running` 상태를 반환하고 에이전트 시작시 PID 파일을 정리합니다.

버전	릴리스 날짜	세부 정보
1.1.2	2020년 8월 4일	추가됨: Ubuntu Server 19.10 및 20.04에 대한 지원. 참고: 버전 19.10이 수명 종료 날짜에 도달했으며 더 이상 Ubuntu 또는 CodeDeploy에서 지원되지 않습니다. 추가됨: Linux와 Ubuntu의 메모리 효율성이 향상되어 예약된 메모리를 보다 적시에 릴리스합니다. 추가됨: 때로 에이전트가 응답하지 않는 문제의 원인이었던 Windows Server “자동 정리”와의 호환성. 추가됨: 정리 중에 비어 있지 않은 디렉터리를 무시하여 배포 실패를 방지합니다. 추가됨: 로스앤젤레스(LA)의 AWS Local Zone에 대한 지원. 추가됨: 인스턴스 메타데이터에서 AZ를 추출하여 AWS 로컬 영역에 대한 호환성을 제공합니다. 추가됨: 이제 사용자는 하위 디렉터리에 아카이브를 제공할 수 있으며 루트 디렉터리에 저장할 필요가 없습니다. 추가됨: 메모리 누수의 원인이 될 수 있는 Rubyzip의 문제를 감지했습니다. Rubyzip을 사용하기 전에 먼저 시스템에 설치된 압축 해제 유틸리티를 사용하도록 압축 해제 명령을 업데이트했습니다. 추가됨: 에이전트 구성 설정으로서의 `:enable_auth_policy:` . 변경됨: 이제 압축 해제 경고가 무시되므로 배포가 계속됩니다.

버전	릴리스 날짜	세부 정보
1.1.0	2020년 6월 30일	변경됨: CodeDeploy 에이전트의 버전 관리는 이제 Ruby 표준 버전 관리 규칙을 따릅니다. 추가됨: 명령줄에서 특정 에이전트 버전을 설치할 수 있도록 설치 및 업데이트 명령에 새로운 파라미터가 도입되었습니다. 제거됨: Linux 및 Ubuntu용 CodeDeploy 에이전트 자동 업데이트 프로그램이 제거되었습니다. CodeDeploy 에이전트의 자동 업데이트를 구성하려면 를 사용하여 CodeDeploy 에이전트 설치를 AWS Systems Manager 참조하세요.
1.0.1.1597	2018년 11월 15일	기능 향상: CodeDeploy는 Ubuntu 18.04를 지원합니다. 기능 향상: CodeDeploy는 Ruby 2.5를 지원합니다. 기능 향상: CodeDeploy는 FIPS 엔드포인트를 지원합니다. FIPS 엔드포인트에 대한 자세한 내용은 FIPS 140-2 개요를 참조하세요. CodeBuild와 함께 사용할 수 있는 엔드포인트는 CodeDeploy 리전 및 엔드포인트를 참조하세요.
1.0.1.1518	2018년 6월 12일	기능 향상: 폴링 요청을 수락하는 동안 CodeDeploy 에이전트가 닫혔을 때 오류가 발생하는 문제가 해결되었습니다. 기능 향상: 배포가 진행 중일 때 CodeDeploy 에이전트가 닫히지 않도록 해주는 배포 추적 기능이 추가되었습니다. 기능 향상: 파일을 삭제할 때 성능을 개선하였습니다.

버전	릴리스 날짜	세부 정보
1.0.1.1458	2018년 3월 6일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능 향상: 인증서 검증이 더 많은 신뢰할 수 있는 기관을 지원하도록 개선되었습니다. 기능 향상: BeforeInstall 수명 주기 이벤트를 포함하는 배포 도중 로컬 CLI가 실패하는 문제가 해결되었습니다. 기능 향상: CodeDeploy 에이전트를 업데이트하면 활성 배포가 실패하는 문제가 해결되었습니다.
1.0.1.1352	2017년 11월 16일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능: CodeDeploy 에이전트가 설치된 로컬 머신 또는 인스턴스에서 EC2/온프레미스 배포를 테스트 및 디버그하기 위한 새로운 기능을 도입했습니다.
1.0.1.1106	2017년 5월 16일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능: 최근에 성공한 배포의 애플리케이션 개정의 일부가 아닌 대상 위치에서 콘텐츠를 처리하기 위한 새로운 지원이 도입되었습니다. 기존 콘텐트에 대한 배포 옵션에 이제 콘텐츠 유지, 콘텐츠 덮어쓰기 또는 배포 실패가 포함됩니다. 기능 향상: CodeDeploy 에이전트가 AWS SDK for Ruby (aws-sdk-core 2.9.2) 버전 2.9.2와 호환되도록 했습니다.

버전	릴리스 날짜	세부 정보
1.0.1.1095	2017년 3월 29일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능 향상: 중국(베이징) 리전에 CodeDeploy 에이전트에 대한 지원이 추가되었습니다. 기능 향상: 수명 주기 이벤트 후크에서 호출한 경우 Windows Server 인스턴스에서 실행되도록 Puppet이 활성화되었습니다. 기능 향상: `untar` 작업의 처리가 개선되었습니다.
1.0.1.1067	2017년 1월 6일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능 향상: 배포 실패에 대한 보다 구체적인 원인을 포함하도록 많은 수의 오류 메시지가 수정되었습니다. 기능 향상: 일부 배포 중 CodeDeploy 에이전트가 올바른 애플리케이션 수정 버전을 식별하지 못하도록 하는 문제가 해결되었습니다. 기능 향상: `untar` 작업 전/후 `pushd` 및 `popd` 사용을 되돌립니다.
1.0.1.1045	2016년 11월 21일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능 향상: CodeDeploy 에이전트가 (AWS SDK for Ruby aws-sdk-core 2.6.11) 버전 2.6.11과 호환되도록 했습니다.

버전	릴리스 날짜	세부 정보
1.0.1.1037	2016년 10월 19일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. Amazon Linux, RHEL 및 Ubuntu Server 인스턴스용 CodeDeploy 에이전트가 다음 변경으로 업데이트되었습니다. Windows Server 인스턴스의 경우 최신 버전은 1.0.1.998로 유지됩니다. 기능 향상: 이제 에이전트가 인스턴스에 설치된 Ruby 버전을 확인할 수 있게 되어 해당 버전을 사용해 `codedeploy-agent` 스크립트를 호출할 수 있습니다.
1.0.1.1011.1	2016년 8월 17일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능 향상: 셸 지원 관련 문제로 인해 버전 1.0.1.1011에서 도입된 변경 사항이 제거되었습니다. 이 에이전트 버전은 기능 측면에서는 2016년 7월 11일에 출시된 버전 1.0.1.998과 동등합니다.
1.0.1.1011	2016년 8월 15일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. Amazon Linux, RHEL 및 Ubuntu Server 인스턴스용 CodeDeploy 에이전트가 다음 변경으로 업데이트되었습니다. Windows Server 인스턴스의 경우 최신 버전은 1.0.1.998로 유지됩니다. 기능: systemd init 시스템이 사용 중인 운영 체제에서 Bash 셸을 사용하여 CodeDeploy 에이전트를 호출하기 위한 지원이 추가되었습니다. 기능 향상: CodeDeploy 에이전트 및 CodeDeploy 에이전트 업데이트 프로그램에서 Ruby 2.x의 모든 버전에 대한 지원이 활성화되었습니다. 업데이트된 CodeDeploy 에이전트는 더 이상 Ruby 2.0에만 종속되지 않습니다. Ruby 2.0은 CodeDeploy 에이전트 설치 프로그램의 deb 및 rpm 버전에 계속 필요합니다.

버전	릴리스 날짜	세부 정보
1.0.1.998	2016년 7월 11일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능 향상: 루트 이외의 사용자 프로필로 CodeDeploy 에이전 트를 실행하기 위한 지원이 수정되었습니다. 환경 변수와의 충 돌을 방지하기 위해 USER 변수가 CODEDEPLOY_USER 로 대 체되었습니다.
1.0.1.966	2016년 6월 16일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능: 루트 이외의 사용자 프로필로 CodeDeploy 에이전트를 실행하기 위한 지원이 도입되었습니다. 기능 향상: CodeDeploy 에이전트에서 배포 그룹에 대해 보관 하도록 하려는 애플리케이션 개정의 개수를 지정하는 지원이 수정되었습니다. 기능 향상: CodeDeploy 에이전트가 AWS SDK for Ruby (aws-sdk-core 2.3) 버전 2.3과 호환되도록 했습니다. 기능 향상: 배포 중 UTF-8 인코딩 관련 문제가 해결되었습니다. 기능 향상: 프로세스 이름 식별 시 정확성이 개선되었습니다.
1.0.1.950	2016년 3월 24일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한 다면 배포에 실패할 수 있습니다. 기능: 설치 프록시 지원이 추가되었습니다. 기능 향상: 최신 버전이 이미 설치되어 있는 경우 CodeDeploy 에이전트를 다운로드하지 않도록 설치 스크립트가 업데이트되 었습니다.

버전	릴리스 날짜	세부 정보
1.0.1.934	2016년 2월 11일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한다면 배포에 실패할 수 있습니다. 기능: CodeDeploy 에이전트에서 배포 그룹에 대해 보관하도록 하려는 애플리케이션 개정의 개수를 지정하는 지원이 도입되었습니다.
1.0.1.880	2016년 1월 11일	참고: 이 버전은 더 이상 지원되지 않기 때문에 사용한다면 배포에 실패할 수 있습니다. 기능 향상: CodeDeploy 에이전트가 AWS SDK for Ruby (aws-sdk-core 2.2) 버전 2.2와 호환되도록 했습니다. 버전 2.1.2는 계속해서 지원됩니다.
1.0.1.854	2015년 11월 17일	참고: 이 버전은 더 이상 지원되지 않습니다. 이 버전을 사용한다면 배포에 실패할 수 있습니다. 기능: SHA-256 해시 알고리즘에 대한 지원이 도입되었습니다. 기능: `.version` 파일에서 버전 추적 지원이 도입되었습니다. 기능: 환경 변수를 통해 배포 그룹 ID를 사용할 수 있게 되었습니다. 기능 향상: Amazon CloudWatch Logs를 사용한 CodeDeploy 에이전트 로그 모니터링 지원이 추가되었습니다.

관련 정보는 다음 자료를 참조하세요.

- CodeDeploy 에이전트의 버전 확인
- CodeDeploy 에이전트 설치

CodeDeploy 에이전트 버전 기록은 GitHub의 릴리스 리포지토리를 참조하세요.

CodeDeploy 프로세스 관리

CodeDeploy 에이전트는 systemd를 사용하여 지원되는 모든 Linux 배포에서 에이전트 프로세스를 관리합니다.

버전 2.0.x 이상의 경우 에이전트는 기본 시스템 단위(codedeploy-agent.service)만 제공합니다. /etc/init.d/codedeploy-agent 스크립트가 없습니다. systemctl를 사용하여 에이전트를 관리하거나 , start, stop restart또는 status 하위 명령을 사용하여 바이너리를 직접 호출합니다.

버전 1.8.x 이하의 경우 rpm 및 deb 배포 모두의 시작 스크립트와 함께 제공됩니다/etc/init.d/codedeploy-agent. systemd를 지원하는 시스템에서 service 명령을 사용하면 systemd가 프로세스를 관리하도록 허용하는 대신 init.d 스크립트를 실수로 실행할 수 있습니다. 이 문제를 방지하려면 에이전트 작업에 systemctl를 사용합니다.

CodeDeploy 에이전트를 관리하려면 systemctl 유틸리티를 사용합니다.

```
sudo systemctl start codedeploy-agent
sudo systemctl stop codedeploy-agent
sudo systemctl restart codedeploy-agent
sudo systemctl status codedeploy-agent
```

macOS에서 에이전트는에 com.amazon.aws.codedeploy-agent정의된 시작 데몬 로 실행됩니다/Library/LaunchDaemons/com.amazon.aws.codedeploy-agent.plist.는 부팅 시 에이전트를 시작하고 충돌 시 에이전트를 다시 시작합니다. macOS에서 에이전트를 관리하려면 launchctl 유틸리티를 사용합니다.

```
sudo launchctl bootstrap system /Library/LaunchDaemons/com.amazon.aws.codedeploy-
agent.plist
sudo launchctl bootout system/com.amazon.aws.codedeploy-agent
sudo launchctl kickstart -k system/com.amazon.aws.codedeploy-agent
sudo launchctl print system/com.amazon.aws.codedeploy-agent
```

이러한 명령은 각각 에이전트의 상태를 시작, 중지, 다시 시작 및 표시합니다.

애플리케이션 수정 버전 및 로그 파일 정리

CodeDeploy 에이전트는 인스턴스에 개정 및 로그 파일을 보관합니다. CodeDeploy 에이전트는 이러한 아티팩트를 정리해 디스크 공간을 절약합니다.

애플리케이션 수정 버전 배포 로그: 에이전트 구성 파일에서 :max_revisions: 옵션을 사용하여 양의 정수를 입력해 보관할 애플리케이션 개정 수를 지정할 수 있습니다. 또한 CodeDeploy는 이러한 수정 버전에 대한 로그 파일을 보관합니다. 최근에 성공한 배포에 대한 로그 파일을 제외한 그 외 모든 항목은 삭제됩니다. 실패한 배포 수가 보유 개정 수를 초과하더라도 로그 파일은 항상 보존됩니다. 값을 지정하지 않으면 CodeDeploy는 현재 배포된 개정 이외에 최근 개정 5개를 보유합니다.

CodeDeploy 로그: Amazon Linux, Ubuntu Server, RHEL 및 macOS 인스턴스의 경우 CodeDeploy 에이전트는 `/var/log/aws/codedeploy-agent` 폴더 아래의 로그 파일을 교체합니다. 로그 파일은 매일 00:00:00(인스턴스 시간) 또는 파일이 64MB에 도달하는 시점 중 먼저 도래하는 시점에 교체됩니다. 같은 날 두 번째 교체가 발생하면 에이전트는 숫자 접미사(예: `codedeploy-agent.20260715.1.log`)를 추가합니다. 에이전트는 7일 후에 로그 파일을 삭제합니다. 교체된 로그 파일의 이름 지정 패턴은 `codedeploy-agent.`*`YYYYMMDD`*`.log`입니다.

CodeDeploy 에이전트에 의해 설치된 파일

CodeDeploy 에이전트는 개정, 배포 기록 및 배포 스크립트를 인스턴스의 루트 디렉터리에 저장합니다. 이 디렉터리의 기본값과 위치는 다음과 같습니다.

`'/opt/codedeploy-agent/deployment-root'` Amazon Linux, Ubuntu Server, RHEL 및 macOS 인스턴스용

Windows Server 인스턴스의 `'C:\ProgramData\Amazon\CodeDeploy'`

CodeDeploy 에이전트 구성 파일의 root_dir 설정을 사용하여 디렉터리의 이름과 위치를 구성할 수 있습니다. 자세한 내용은 CodeDeploy 에이전트 구성 참조 섹션을 참조하세요.

다음은 루트 디렉터리의 파일과 디렉터리 구조를 보여주는 예시입니다. 이 구조는 배포 그룹 N개가 있으며, 각 배포 그룹에는 배포 N개가 포함된다고 가정합니다.

```
|--deployment-root/
|-- deployment group 1 ID
|    |-- deployment 1 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment 2 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment N ID
|    |    |-- Contents and logs of the deployment's revision
|-- deployment group 2 ID
|    |-- deployment 1 ID
|    |    |-- bundle.tar
```

```
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|    |-- deployment 2 ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|    |-- deployment N ID
|    |    |-- bundle.tar
|    |    |-- deployment-archive
|    |    |    | -- contents of the deployment's revision
|    |    |-- logs
|    |    |    | -- scripts.log
|-- deployment group N ID
|    |-- deployment 1 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment 2 ID
|    |    |-- Contents and logs of the deployment's revision
|    |-- deployment N ID
|    |    |-- Contents and logs of the deployment's revision
|-- deployment-instructions
|    |-- [deployment group 1 ID]_cleanup
|    |-- [deployment group 2 ID]_cleanup
|    |-- [deployment group N ID]_cleanup
|    |-- [deployment group 1 ID]_install.json
|    |-- [deployment group 2 ID]_install.json
|    |-- [deployment group N ID]_install.json
|    |-- [deployment group 1 ID]_last_successful_install
|    |-- [deployment group 2 ID]_last_successful_install
|    |-- [deployment group N ID]_last_successful_install
|    |-- [deployment group 1 ID]_most_recent_install
|    |-- [deployment group 2 ID]_most_recent_install
|    |-- [deployment group N ID]_most_recent_install
|-- deployment-logs
|    |-- codedeploy-agent-deployments.log
```

- Deployment Group ID 폴더는 각 배포 그룹을 대표합니다. 배포 그룹 디렉터리의 이름은 배포 그룹 디렉터리의 ID입니다(예: `acde1916-9099-7caf-fd21-012345abcdef`). 각각의 배포 그룹 디렉터리에는 해당 배포 그룹의 각 배포 시도에 해당하는 하위 디렉터리가 하나씩 있습니다.

batch-get-deployments 명령을 사용하여 배포 그룹 ID를 찾을 수 있습니다.

- Deployment ID 폴더는 배포 그룹 내 개별 배포를 대표합니다. 각 배포 디렉터리의 이름은 배포 디렉터리의 ID입니다. 각 폴더는 다음을 포함합니다.
  - bundle.tar는 배포 개정 내용을 담고 있는 압축 파일입니다. 개정을 보려면 zip 압축 해제 유틸리티를 사용합니다.
  - deployment-archive는 배포 개정 내용을 담고 있는 디렉터리입니다.
  - logs는 `scripts.log` 파일을 포함하는 디렉터리입니다. 이 파일은 배포의 AppSpec 파일에서 지정하는 모든 스크립트 출력을 나열합니다.

  배포에 대한 폴더를 찾고 싶지만 배포 ID 또는 배포 그룹 ID를 모르는 경우 AWS CodeDeploy 콘솔 또는 AWS CLI 를 사용하여 찾을 수 있습니다. 자세한 내용은 CodeDeploy 배포 세부 정보 보기 섹션을 참조하세요.

  배포 그룹에 아카이브할 수 있는 기본 최대 배포 수는 5개입니다. 5개 아카이브 후 다른 배포를 아카이브하면 가장 오래된 아카이브가 삭제됩니다. CodeDeploy 에이전트 구성 파일의 max_revisions 설정을 사용하여 기본값을 변경할 수 있습니다. 자세한 내용은 CodeDeploy 에이전트 구성 참조 섹션을 참조하세요.

  > **Note**
  >
  > 아카이브된 배포가 사용하는 하드 디스크 공간을 복구하고 싶다면, max_revisions 설정을 1이나 2 같은 낮은 수치로 조정하세요. 다음 배포에서는 아카이빙된 배포를 삭제하여 번호가 지정한 수와 같도록 합니다.

- deployment-instructions는 각 배포 그룹을 위한 텍스트 파일 4개를 포함합니다.
  - [배포 그룹 ID]-정리, 배포 중에 실행되는 각 명령의 실행 취소 버전이 있는 텍스트 파일입니다. 예제 파일 이름은 `acde1916-9099-7caf-fd21-012345abcdef-cleanup`입니다.
  - [Deployment Group ID]-install.json은 가장 최근 배포에서 생성된 JSON 파일입니다. 배포 과정에서 실행된 명령을 확인할 수 있습니다. 예제 파일 이름은 `acde1916-9099-7caf-fd21-012345abcdef-install.json`입니다.
  - [배포 그룹 ID]_last_successful_install은 마지막으로 성공한 배포의 아카이브 디렉터리를 나열하는 텍스트 파일입니다. 이 파일은 CodeDeploy 에이전트가 배포 애플리케이션의 모든 파일을 인스턴스로 복사할 때 생성됩니다. 이 파일은 CodeDeploy 에이전트가 다음 배포 과정에서 실행할 `ApplicationStop` 및 `BeforeInstall` 스크립트를 결정하는 데 사용합니다. 예제 파일 이름은 `acde1916-9099-7caf-fd21-012345abcdef_last_successful_install`입니다.

  - [Deployment Group ID]_most_recent_install은 가장 최근 배포의 아카이브 디렉터리 이름을 나열하는 텍스트 파일입니다. 이 파일은 배포의 파일을 다운로드했을 때 생성됩니다. [배포 그룹 ID]_last_successful_install 파일은 다운로드한 파일이 최종 대상으로 복사될 때이 파일 뒤에 생성됩니다. 예제 파일 이름은 `acde1916-9099-7caf-fd21-012345abcdef_most_recent_install`입니다.
- deployment-logs는 다음 로그 파일을 포함합니다.
  - codedeploy-agent.yyyymmdd.log 파일은 배포가 실행된 각 날짜에 생성됩니다. 각 로그 파일에는 해당 날짜의 배포에 대한 정보가 들어 있습니다. 이러한 로그 파일은 권한 문제 같은 디버깅 문제를 해결하는 데 유용합니다. 로그 파일의 기본 이름은 `codedeploy-agent.log`입니다. 날이 지나면 적용 일자가 파일 이름에 삽입됩니다. 예를 들어 오늘이 2018년 1월 3일이라면, `codedeploy-agent.log`에서 오늘의 모든 배포 관련 정보를 확인할 수 있습니다. 내일, 즉 2018년 1월 4일에는 로그 파일 이름이 `codedeploy-agent.20180103.log`로 변경됩니다.
  - codedeploy-agent-deployments.log는 개별 배포에 대해 `scripts.log` 파일 내용을 컴파일합니다. `scripts.log` 파일은 각 `logs` 폴더의 `Deployment ID` 하위 폴더에 있습니다. 이 파일의 항목 앞에는 배포 ID가 붙습니다. 예를 들어 ID가 `[d-ABCDEF123]LifecycleEvent - BeforeInstall`인 배포 중에는 "d-ABCDEF123"이 작성될 수 있습니다. `codedeploy-agent-deployments.log`가 최대 크기가 되면, CodeDeploy 에이전트가 이를 계속 작성하며 오래된 내용은 삭제됩니다.

CodeDeploy 에이전트 작업 관리

이 섹션의 지침은 CodeDeploy 에이전트 설치, 제거, 다시 설치 또는 업데이트 방법과 CodeDeploy 에이전트가 실행 중인지 여부를 확인하는 방법을 보여줍니다.

> **Tip**
>
> 이 페이지의 일부 지침은 에이전트 버전에 따라 다릅니다. 설치된 버전을 잘 모르는 경우 `/opt/codedeploy-agent/bin/codedeploy-agent --version` (Linux 또는 macOS) 또는 `& 'C:\ProgramData\Amazon\CodeDeploy\bin\codedeploy-agent.exe' --version` (Windows)를 실행합니다. 해당 명령이 인식되지 않으면 버전 1.8.x 이하를 실행 중인 것입니다. 자세한 내용은 CodeDeploy 에이전트의 버전 확인 섹션을 참조하세요.

주제

- CodeDeploy 에이전트가 실행 중인지 확인

- CodeDeploy 에이전트의 버전 확인
- CodeDeploy 에이전트 설치
- CodeDeploy 에이전트 업데이트
- CodeDeploy 에이전트 설치 제거
- CloudWatch로 CodeDeploy 에이전트 로그 전송

CodeDeploy 에이전트가 실행 중인지 확인

이 섹션에서는 인스턴스에서 CodeDeploy 에이전트의 실행이 중지되었다고 의심하는 경우 에이전트를 실행하도록 하는 명령에 대해 설명합니다.

주제

- Amazon Linux 또는 RHEL용 CodeDeploy 에이전트가 실행 중인지 확인합니다.
- Ubuntu Server용 CodeDeploy 에이전트가 실행 중인지 확인
- Windows Server용 CodeDeploy 에이전트가 실행 중인지 확인
- macOS용 CodeDeploy 에이전트가 실행 중인지 확인

Amazon Linux 또는 RHEL용 CodeDeploy 에이전트가 실행 중인지 확인합니다.

CodeDeploy 에이전트가 설치되고 실행 중인지 확인하려면 인스턴스에 로그인하여 다음 명령을 실행합니다.

```
systemctl status codedeploy-agent
```

명령이 오류를 반환하면 CodeDeploy 에이전트가 설치되지 않은 것입니다. Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치의 설명에 따라 설치합니다.

CodeDeploy 에이전트가 설치되어 실행 중인 경우 `The AWS CodeDeploy agent is running.` (종료 코드 0)과 같은 메시지가 표시됩니다.

`The AWS CodeDeploy agent is not running.` (종료 코드 3)과 같은 메시지가 표시되면 서비스를 시작하고 다음 두 명령을 한 번에 하나씩 실행합니다.

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

버전 2.0.x 이상의 경우 `codedeploy-agent status` 명령은 LSB 호환 종료 코드: `0` (실행 중), `3` (실행 중 아님), `4` (알 수 없음)을 반환합니다. `systemctl status`를 사용하면 단위 상태가 표시됩니다 (예: `Active: active (running)`).

Ubuntu Server용 CodeDeploy 에이전트가 실행 중인지 확인

CodeDeploy 에이전트가 설치되고 실행 중인지 확인하려면 인스턴스에 로그인하여 다음 명령을 실행합니다.

```
systemctl status codedeploy-agent
```

명령이 오류를 반환하면 CodeDeploy 에이전트가 설치되지 않은 것입니다. Ubuntu Server용 CodeDeploy 에이전트 설치의 설명에 따라 설치합니다.

CodeDeploy 에이전트가 설치되어 실행 중인 경우 `The AWS CodeDeploy agent is running.` (종료 코드 0)과 같은 메시지가 표시됩니다.

`The AWS CodeDeploy agent is not running.` (종료 코드 3)과 같은 메시지가 표시되면 서비스를 시작하고 다음 두 명령을 한 번에 하나씩 실행합니다.

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

Windows Server용 CodeDeploy 에이전트가 실행 중인지 확인

CodeDeploy 에이전트가 설치되고 실행 중인지 확인하려면 인스턴스에 로그인하여 다음 명령을 실행합니다.

```
powershell.exe -Command Get-Service -Name codedeployagent
```

다음과 유사한 출력 화면이 표시되어야 합니다.

```
Status   Name               DisplayName
------   ----               -----------
Running codedeployagent    CodeDeploy Host Agent Service
```

명령이 오류를 반환하면 CodeDeploy 에이전트가 설치되지 않은 것입니다. Windows Server용 CodeDeploy 에이전트 설치의 설명에 따라 설치합니다.

`Status`에 `Running` 이외의 다른 상태가 표시되면 다음 명령을 사용해 서비스를 시작합니다.

```
powershell.exe -Command Start-Service -Name codedeployagent
```

다음 명령을 사용하여 서비스를 시작할 수 있습니다.

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

다음 명령을 사용하여 서비스를 중지할 수 있습니다.

```
powershell.exe -Command Stop-Service -Name codedeployagent
```

macOS용 CodeDeploy 에이전트가 실행 중인지 확인

CodeDeploy 에이전트가 설치되고 실행 중인지 확인하려면 인스턴스에 로그인하여 다음 명령을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent status
```

파일을 찾을 수 없어서 명령이 오류를 반환하면 CodeDeploy 에이전트가 설치되지 않은 것입니다. macOS용 CodeDeploy 에이전트 설치의 설명에 따라 설치합니다.

CodeDeploy 에이전트가 설치되어 실행 중인 경우 `The AWS CodeDeploy agent is running.` (종료 코드 0)과 같은 메시지가 표시됩니다.

`The AWS CodeDeploy agent is not running.` (종료 코드 3)과 같은 메시지가 표시되면 다음 명령으로 에이전트를 시작합니다.

```
sudo launchctl kickstart system/com.amazon.aws.codedeploy-agent
```

이 명령이 서비스를 찾을 수 없다고 보고하면 시작된 데몬이 로드되지 않습니다. 를 로드하고 다음 명령으로 시작합니다.

```
sudo launchctl bootstrap system /Library/LaunchDaemons/com.amazon.aws.codedeploy-
agent.plist
```

시작된 데몬의 상태를 보려면 다음을 실행합니다.

```
sudo launchctl print system/com.amazon.aws.codedeploy-agent
```

CodeDeploy 에이전트의 버전 확인

인스턴스에서 실행 중인 CodeDeploy 에이전트의 버전은 두 가지 방법으로 확인할 수 있습니다.

먼저 버전 2.0.x 이상에서는 모든 플랫폼에서 `codedeploy-agent --version` 명령을 사용합니다.

둘째, 인스턴스의 `.version` 파일에서 버전을 볼 수 있습니다. Amazon Linux, RHEL, Ubuntu Server 및 macOS에서 파일은에 `/opt/codedeploy-agent/.version`있고 Windows Server에서는에 있습니다`C:\ProgramData\Amazon\CodeDeploy\.version`. 파일의 내용은 버전에 따라 다릅니다.

- 버전 2.0.x 이상의 경우 파일에는 형식의 베어 버전 번호가 포함됩니다`agent_version: 2.0.0`. 이 값은 에이전트가 CodeDeploy 서비스에 보고하는 버전과 일치합니다.
- 버전 1.8.x 이하의 경우 버전 문자열은 형식`OFFICIAL_1.0.1.854_rpm`(또는 `_deb`, `_msi`)을 사용합니다.

주제

- Amazon Linux 또는 RHEL에서 버전 확인
- Ubuntu Server에서 버전 확인
- Windows Server에서 버전 확인
- macOS에서 버전 확인
- 에서 사용 가능한 최신 에이전트 버전 확인 AWS 리전

Amazon Linux 또는 RHEL에서 버전 확인

인스턴스에 로그인하여 다음 명령을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent --version
```

2.0.0 이전 버전의 경우 다음을 사용할 수도 있습니다.

```
sudo yum info codedeploy-agent
```

Ubuntu Server에서 버전 확인

인스턴스에 로그인하여 다음 명령을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent --version
```

2.0.0 이전 버전의 경우 다음을 사용할 수도 있습니다.

```
sudo dpkg -s codedeploy-agent
```

Windows Server에서 버전 확인

인스턴스에 로그인하여 다음 명령을 실행합니다.

```
& 'C:\ProgramData\Amazon\CodeDeploy\bin\codedeploy-agent.exe' --version
```

macOS에서 버전 확인

인스턴스에 로그인하여 다음 명령을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent --version
```

설치된 패키지의 버전을 볼 수도 있습니다.

```
pkgutil --pkg-info com.amazon.aws.codedeploy-agent
```

에서 사용 가능한 최신 에이전트 버전 확인 AWS 리전

배포한에서 사용할 수 있는 최신 버전을 확인하려면 해당 리전의 CodeDeploy 리소스 키트 버킷을 쿼리 AWS 리전 합니다. 각 리전 버킷에는 현재 해당 리전에서 릴리스된 에이전트 패키지의 이름을 지정하는 `latestv2/LATEST_VERSION` 파일이 들어 있습니다. 파일은 공개적으로 읽을 수 있으므로 검색하는 데 자격 증명이 필요하지 않습니다.

```
curl https://bucket-name.s3.region-identifier.amazonaws.com/latestv2/LATEST_VERSION
```

bucket-name 및 *region-identifier*를 배포하는 리전의 리소스 키트 버킷 및 리전 식별자로 바꿉니다. 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

이 명령은 다음과 비슷한 출력을 반환합니다.

```
{
    "rpm_x86_64": "releases/codedeploy-agent-2.1.0-9745926189.x86_64.rpm",
    "rpm_aarch64": "releases/codedeploy-agent-2.1.0-9745926189.aarch64.rpm",
    "deb_x86_64": "releases/codedeploy-agent_2.1.0-9745926189_amd64.deb",
    "deb_aarch64": "releases/codedeploy-agent_2.1.0-9745926189_arm64.deb",
    "msi": "releases/codedeploy-agent-2.1.0-9745926189.msi"
}
```

각 키는 패키지 유형의 이름을 지정하고 RPM 및 DEB의 경우 인스턴스 아키텍처의 이름도 지정합니다. 각 값은 해당 패키지의 Amazon S3 객체 키이며 에이전트 버전 및 빌드 번호는 파일 이름의 일부이며 형식입니다*agent-version*-*build-number*. 이전 출력에서 리전에서 사용할 수 있는 최신 버전은 2.1.0입니다.

버전 2.1.2 이상의 경우 파일에 macOS 설치 관리자 패키지라는 이름의 `pkg_aarch64` 키도 있습니다.

이 파일 이름의 버전을 설치하거나 업데이트하려면 CodeDeploy 에이전트 설치 및 섹션을 참조하세요CodeDeploy 에이전트 업데이트. 각 버전에 포함된 기능은 섹션을 참조하세요CodeDeploy 에이전트의 버전 기록.

CodeDeploy 에이전트 설치

EC2 인스턴스 또는 온프레미스 서버에서 CodeDeploy를 사용하려면 CodeDeploy 에이전트를 먼저 설치해야 합니다. AWS Systems Manager 또는 명령줄을 사용하여 CodeDeploy 에이전트를 설치할 수 있습니다. Systems Manager에 대한 자세한 내용은 AWS Systems Manager란 무엇입니까를 참조하세요.

설치할 권장 버전은 CodeDeploy 에이전트의 버전 기록 섹션을 참조하세요.

주제

- 를 사용하여 CodeDeploy 에이전트 설치 AWS Systems Manager
- 명령줄을 사용하여 CodeDeploy 에이전트 설치

를 사용하여 CodeDeploy 에이전트 설치 AWS Systems Manager

AWS Management Console 또는를 사용하여 AWS CLI 를 사용하여 Amazon EC2 또는 온프레미스 인스턴스에 CodeDeploy 에이전트를 설치할 수 있습니다 AWS Systems Manager. 특정 버전을 설치

하거나 항상 에이전트의 최신 버전을 설치하도록 선택할 수 있습니다. 에 대한 자세한 내용은 란 AWS Systems Manager 무엇입니까?를 AWS Systems Manager참조하십시오.

Amazon S3 버킷에서 CodeDeploy 에이전트를 설치할 수도 있습니다. Amazon S3 다운로드 링크 사용에 대한 자세한 내용은 명령줄을 사용하여 CodeDeploy 에이전트 설치 섹션을 참조하세요.

주제

- 사전 조건
- CodeDeploy 에이전트 설치

사전 조건

CodeDeploy 시작하기의 단계에 따라 IAM 권한 및 AWS CLI를 설정합니다.

Systems Manager를 사용하여 온프레미스 서버에 CodeDeploy 에이전트를 설치하는 경우 Amazon EC2 Systems Manager 온프레미스 서버를 등록해야 합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 하이브리드 환경에서 Systems Manager 설정을 참조하세요.

CodeDeploy 에이전트 설치

Systems Manager를 사용하여 CodeDeploy 에이전트를 설치하기 전에 먼저 인스턴스가 Systems Manager에 대해 올바르게 구성되어 있는지 확인해야 합니다.

SSM 에이전트 설치 또는 업데이트

Amazon EC2 인스턴스에서는 CodeDeploy 에이전트를 사용하려면 해당 인스턴스가 2.3.274.0 이상 버전을 실행하고 있어야 합니다. CodeDeploy 에이전트를 설치하기 전에, 인스턴스에 SSM 에이전트를 업데이트하거나 설치합니다(아직 하지 않은 경우).

SSM 에이전트는에서 제공하는 일부 Amazon EC2 AMIs에 사전 설치되어 제공됩니다 AWS. 자세한 내용은 SSM 에이전트가 사전 설치된 Amazon Machine Image(AMI)를 참조하세요.

> **Note**
>
> 인스턴스의 운영 체제도 CodeDeploy 에이전트에서 지원되는지 확인합니다. 자세한 내용은 CodeDeploy 에이전트에서 지원하는 운영 체제 섹션을 참조하세요.

Linux를 실행하는 인스턴스에 SSM 에이전트를 설치하거나 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 Linux 인스턴스에 SSM 에이전트 설치 및 구성을 참조하세요.

Windows Server를 실행하는 인스턴스에 SSM 에이전트를 설치하거나 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 Windows 인스턴스에 SSM 에이전트 설치 및 구성을 참조하세요.

EC2 Mac 인스턴스에 SSM 에이전트를 설치하거나 업데이트하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 macOS용 EC2 인스턴스에 SSM 에이전트 수동 설치 및 제거를 참조하세요.

(선택 사항) Systems Manager 사전 조건 확인

SystemsManager Run Command를 사용하여 CodeDeploy 에이전트를 설치하기 전에 인스턴스가 Systems Manager 최소 요구 사항을 충족하는지 확인하세요. 자세한 내용은 AWS Systems Manager 사용 설명서에서 AWS Systems Manager설정을 참조하세요.

CodeDeploy 에이전트 설치

SSM을 사용하여 CodeDeploy를 한 번만 설치하거나, 최신 버전을 계속 설치하도록 일정을 설정할 수 있습니다.

CodeDeploy 에이전트를 설치하려면 원하는 에이전트 버전의 Systems Manager Distributor 패키지를 선택한 다음 AWS Systems Manager 배포자를 사용하여 패키지 설치 또는 업데이트의 단계를 따릅니다. 패키지는 에이전트 버전에 따라 다릅니다.

- 버전 2.0.x 이상에서 `AWSCodeDeployAgentV2` 패키지를 선택합니다. 이 패키지는 버전 2.1.2부터 macOS를 지원하는 유일한 패키지입니다.
- 버전 1.8.x 이하에서 `AWSCodeDeployAgent` 패키지를 선택합니다.

명령줄을 사용하여 CodeDeploy 에이전트 설치

명령줄을 사용하여 CodeDeploy 에이전트를 설치하고 실행하는 방법에 대해서는 다음 주제를 참조하세요.

주제

- Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치
- Ubuntu Server용 CodeDeploy 에이전트 설치
- Windows Server용 CodeDeploy 에이전트 설치
- macOS용 CodeDeploy 에이전트 설치

Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치

인스턴스에 로그인하여 다음 명령을 한 번에 하나씩 실행합니다. yum을 사용하여 패키지를 설치할 때 첫 번째 명령인 `sudo yum update`를 먼저 실행하는 것이 가장 좋습니다. 그러나 모든 패키지를 업데이트하지 않으려면 이 명령을 건너뛸 수 있습니다.

```
sudo yum update
```

버전 2.0.x 이상의 경우:

```
sudo yum install wget
```

버전 1.8.x 이하의 경우:

```
sudo yum install ruby wget
```

(선택 사항) 이전 에이전트 캐싱 정보의 AMI를 정리하려면 다음 스크립트를 실행합니다.

```
#!/bin/bash
CODEDEPLOY_BIN="/opt/codedeploy-agent/bin/codedeploy-agent"
$CODEDEPLOY_BIN stop
yum erase codedeploy-agent -y
```

홈 디렉터리로 이동합니다.

```
cd /home/ec2-user
```

> **Note**
>
> 앞의 명령에서 `/home/ec2-user`는 Amazon Linux 또는 RHEL Amazon EC2 인스턴스의 기본 사용자 이름을 나타냅니다. 사용자 지정 AMI를 사용하여 인스턴스를 만든 경우 AMI 소유자가 다른 기본 사용자 이름을 지정했을 수 있습니다.

CodeDeploy 에이전트 설치 프로그램을 다운로드합니다. `latestv2/` 접두사는 버전 2.0.x 설치 관리자를 제공하고 `latest/` 접두사는 버전 1.8.x 설치 관리자를 제공합니다.

버전 2.0.x 이상의 경우:

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latestv2/install
```

버전 1.8.x 이하의 경우:

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
```

*bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름이며, *region-identifier*는 리전의 식별자입니다.

예를 들어 버전 2.0.x 이상의 경우:

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latestv2/install`

버전 1.8.x 이하의 경우:

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install`

버킷 이름 및 리전 식별자 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

`install` 파일에 대한 실행 권한을 설정합니다.

```
chmod +x ./install
```

최신 버전의 CodeDeploy 에이전트를 설치하려면 다음을 수행합니다.

- ```
  sudo ./install auto
  ```

특정 버전의 CodeDeploy 에이전트를 설치하려면 다음을 수행합니다.

- 해당 리전에서 사용 가능한 버전을 나열합니다.

  ```
  aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier
   | grep '\.rpm$'
  ```

- 다음 버전 중 하나를 설치합니다.

  버전 2.0.x 이상의 경우 패키지는 아키텍처별로 다릅니다.

  ```
  sudo ./install auto -v releases/codedeploy-agent-version.arch.rpm
  ```
  ```

여기서 *arch*는 인스턴스 아키텍처에 aarch64 따라 x86_64 또는 입니다.

버전 1.8.x 이하의 경우 패키지는 noarch를 사용합니다.

```
sudo ./install auto -v releases/codedeploy-agent-version.noarch.rpm
```

최신 릴리스 버전은 섹션을 참조하세요CodeDeploy 에이전트의 버전 기록.

서비스가 실행 중인지 확인하려면 다음 명령을 실행합니다.

```
systemctl status codedeploy-agent
```

버전 2.0.x 이상의 경우 에이전트가 실행 중인 경우 다음과 같은 메시지가 표시됩니다. The AWS CodeDeploy agent is running.

버전 1.8.x 이하의 경우 에이전트가 실행 중인 경우 다음과 같은 메시지가 표시되어야 합니다. The AWS CodeDeploy agent is running.

에이전트가 실행되지 않는 경우 서비스를 시작하고 상태를 확인합니다.

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

Ubuntu Server용 CodeDeploy 에이전트 설치

Ubuntu Server에 CodeDeploy 에이전트를 설치하려면

1. 인스턴스에 로그인합니다.
2. 다음 명령을 차례로 입력합니다.

   ```
   sudo apt update
   ```

   버전 2.0.x 이상의 경우:

   ```
   sudo apt install wget
   ```

버전 1.8.x 이하의 경우:

```
sudo apt install ruby-full wget
```

3. 다음 명령을 입력합니다.

```
cd /home/ubuntu
```

*/home/Ubuntu*는 Ubuntu Server 인스턴스의 기본 사용자 이름을 나타냅니다. 사용자 지정 AMI를 사용하여 인스턴스를 만든 경우 AMI 소유자가 다른 기본 사용자 이름을 지정했을 수 있습니다.

4. 다음 명령을 입력합니다. `latestv2/` 접두사는 버전 2.0.x 설치 관리자를 제공하고 `latest/` 접두사는 버전 1.8.x 설치 관리자를 제공합니다.

버전 2.0.x 이상의 경우:

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latestv2/install
```

버전 1.8.x 이하의 경우:

```
wget https://bucket-name.s3.region-identifier.amazonaws.com/latest/install
```

*bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름이며, *region-identifier*는 리전의 식별자입니다.

예를 들어 버전 2.0.x 이상의 경우:

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latestv2/install`

버전 1.8.x 이하의 경우:

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install`

버킷 이름 및 리전 식별자 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

5. 다음 명령을 입력합니다.

```
chmod +x ./install
```

6. 다음 중 하나를 수행하세요.

   - 최신 버전의 CodeDeploy 에이전트를 설치하려면 다음을 수행합니다.

     ```
     sudo ./install auto
     ```

   - 특정 버전의 CodeDeploy 에이전트를 설치하려면 다음을 수행합니다.
     - 해당 리전에서 사용 가능한 버전을 나열합니다.

       ```
       aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier | grep '\.deb$'
       ```

     - 다음 버전 중 하나를 설치합니다.

       버전 2.0.x 이상의 경우 패키지는 아키텍처별로 다릅니다.

       ```
       sudo ./install auto -v releases/codedeploy-agent_version_arch.deb
       ```

       여기서 *arch*는 인스턴스 아키텍처에 `arm64` 따라 `amd64` 또는 입니다.

       버전 1.8.x 이하의 경우 패키지는 다른 명명 규칙을 사용합니다.

       ```
       sudo ./install auto -v releases/codedeploy-agent_version_all.deb
       ```

       최신 릴리스 버전은 섹션을 참조하세요CodeDeploy 에이전트의 버전 기록.

서비스가 실행 중인지 확인하려면

1. 다음 명령을 입력합니다.

   ```
   systemctl status codedeploy-agent
   ```

   CodeDeploy 에이전트가 설치되어 실행 중인 경우:

   버전 2.0.x 이상에서는 다음이 표시됩니다. `The AWS CodeDeploy agent is running.`

   버전 1.8.x 이하의 경우 다음이 표시되어야 합니다. `The AWS CodeDeploy agent is running.`

2. 에이전트가 실행되지 않는 경우 서비스를 시작하고 상태를 확인합니다.

```
systemctl start codedeploy-agent
```

```
systemctl status codedeploy-agent
```

Windows Server용 CodeDeploy 에이전트 설치

Windows Server 인스턴스에서 다음 방법 중 하나를 사용하여 CodeDeploy 에이전트를 다운로드하고 설치할 수 있습니다.

- 사용 AWS Systems Manager (권장)
- 일련의 Windows PowerShell 명령 실행
- 직접 다운로드 링크 선택
- Amazon S3 복사 명령을 실행합니다.

> **Note**
>
> CodeDeploy 에이전트가 설치된 폴더는 `C:\ProgramData\Amazon\CodeDeploy`입니다. 이 경로에 디렉터리 정션 또는 symlink를 배치하지 마십시오. 버전 2.0.x 이상의 경우 에이전트는이 요구 사항을 적용하고 서비스 바이너리가 미만이 아닌 경우 시작을 거부합니다`C:\ProgramData\Amazon\CodeDeploy\`.

주제

- Systems Manager 사용
- Windows PowerShell 사용
- 직접 링크 사용
- Amazon S3 복사 명령 사용

Systems Manager 사용

CodeDeploy 에이전트를 설치하려면 를 사용하여 CodeDeploy 에이전트 설치 AWS Systems Manager의 지침을 따릅니다.

Windows PowerShell 사용

인스턴스에 로그인하여 Windows PowerShell에서 다음 명령을 실행합니다.

1. 인터넷에서 다운로드한 모든 스크립트와 구성 파일에 신뢰할 수 있는 게시자의 서명이 있어야 합니다. 실행 정책을 변경하라는 메시지가 나타나면 "**Y**"를 입력합니다.

   ```
   Set-ExecutionPolicy RemoteSigned
   ```

2. 를 로드합니다 AWS Tools for Windows PowerShell.

   ```
   Import-Module AWSPowerShell
   ```

3. CodeDeploy 에이전트 설치 파일이 다운로드될 디렉터리를 생성합니다.

   ```
   New-Item -Path "c:\temp" -ItemType "directory" -Force
   ```

4. `Set-AWSCredential` 및 `Initialize-AWSDefaultConfiguration` 명령을 사용하여 AWS 자격 증명을 구성합니다. 자세한 내용은 AWS Tools for PowerShell 사용 설명서의 AWS 자격 증명 사용을 참조하세요.

5. CodeDeploy 에이전트 설치 파일을 다운로드합니다.

   > **Note**
   >
   > 최신 릴리스 버전은 섹션을 참조하세요CodeDeploy 에이전트의 버전 기록.

   최신 버전의 CodeDeploy 에이전트를 설치합니다. `latestv2/` 접두사는 버전 2.x(2.0.x 이상) 설치 관리자를 제공하고 접`latest/`두사는 버전 1.8.x 설치 관리자를 제공합니다.

   - 버전 2.0.x 이상의 경우:

     ```
     powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latestv2/
     codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
     ```

     버전 1.8.x 이하의 경우:

     ```
     powershell.exe -Command Read-S3Object -BucketName bucket-name -Key latest/
     codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
     ```

특정 버전의 CodeDeploy 에이전트를 설치하려면

- ```
  powershell.exe -Command Read-S3Object -BucketName bucket-name -Key releases/
  codedeploy-agent-###.msi -File c:\temp\codedeploy-agent.msi
  ```

*bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름입니다. 예를 들어 미국 동부(오하이오) 리전의 경우 *bucket-name*을 aws-codedeploy-us-east-2로 바꿉니다. 버킷 이름 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

6. CodeDeploy 에이전트 설치 파일을 실행합니다.

```
c:\temp\codedeploy-agent.msi /quiet /l c:\temp\host-agent-install-log.txt
```

서비스가 실행 중인지 확인하려면 다음 명령을 실행합니다.

```
powershell.exe -Command Get-Service -Name codedeployagent
```

CodeDeploy 에이전트를 방금 설치했고 아직 시작하지 않은 경우 Get-Service 명령을 실행한 후 상태에 **Start...**이 표시되어야 합니다.

```
Status     Name               DisplayName
------     ----               -----------
Start...   codedeployagent    CodeDeploy Host Agent Service
```

CodeDeploy 에이전트가 이미 실행 중인 경우 Get-Service 명령을 실행한 후 상태에 **Running**이 표시되어야 합니다.

```
Status     Name               DisplayName
------     ----               -----------
Running    codedeployagent    CodeDeploy Host Agent Service
```
```

직접 링크 사용

Windows Server 인스턴스의 브라우저 보안 설정이 `https://s3.*.amazonaws.com` 등에 권한을 제공하는 경우, 리전의 직접 링크를 사용하여 CodeDeploy 에이전트를 다운로드한 후 설치 관리자를 수동으로 실행할 수 있습니다.

`latestv2/` 접두사는 버전 2.x(2.0.x 이상) 설치 관리자를 제공하고 접`latest/`두사는 버전 1.8.x 설치 관리자를 제공합니다.

버전 2.0.x 이상의 경우 링크는 다음과 같습니다.

```
https://s3.region.amazonaws.com/aws-codedeploy-region/latestv2/codedeploy-agent.msi
```

버전 1.8.x 이하의 경우:

```
https://s3.region.amazonaws.com/aws-codedeploy-region/latest/codedeploy-agent.msi
```

여기서 *##*은 애플리케이션을 배포하는 AWS 리전입니다.

예를 들어 버전 2.0.x 이상의 경우:

```
https://s3.af-south-1.amazonaws.com/aws-codedeploy-af-south-1/latestv2/codedeploy-
agent.msi
```

> **Important**
>
> CodeDeploy 애플리케이션과 같은 리전에 있는 `.msi` 파일을 가져옵니다. 다른 리전을 선택하면 `.msi` 파일을 실행할 때 `codedeploy-agent-log` 파일에 `inconsistent region` 오류가 발생할 수 있습니다.

Amazon S3 복사 명령 사용

AWS CLI 가 인스턴스에 설치된 경우 Amazon S3 cp 명령을 사용하여 CodeDeploy 에이전트를 다운로드한 다음 설치 관리자를 수동으로 실행할 수 있습니다. 자세한 내용은 Microsoft Windows AWS Command Line Interface 에 설치를 참조하세요.

`latestv2/` 접두사는 버전 2.x(2.0.x 이상) 설치 관리자를 제공하고 접`latest/`두사는 버전 1.8.x 설치 관리자를 제공합니다.

버전 2.0.x 이상의 경우 Amazon S3 명령은 다음과 같습니다.

```
aws s3 cp s3://aws-codedeploy-region/latestv2/codedeploy-agent.msi codedeploy-agent.msi
 --region region
```

버전 1.8.x 이하의 경우:

```
aws s3 cp s3://aws-codedeploy-region/latest/codedeploy-agent.msi codedeploy-agent.msi
 --region region
```

여기서 *##*은 애플리케이션을 배포하는 AWS 리전입니다.

예를 들어 버전 2.0.x 이상의 경우:

```
aws s3 cp s3://aws-codedeploy-af-south-1/latestv2/codedeploy-agent.msi codedeploy-
agent.msi --region af-south-1
```

macOS용 CodeDeploy 에이전트 설치

인스턴스에 로그인하여 다음 명령을 한 번에 하나씩 실행합니다.

홈 디렉터리로 이동합니다.

```
cd ~
```

CodeDeploy 에이전트 설치 관리자 패키지를 다운로드합니다.

```
curl -O https://bucket-name.s3.region-identifier.amazonaws.com/latestv2/codedeploy-
agent.aarch64.pkg
```

*bucket-name*은 해당 리전의 CodeDeploy 리소스 키트 파일이 포함되어 있는 Amazon S3 버킷의 이름이며, *region-identifier*는 리전의 식별자입니다.

예제:

`https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latestv2/codedeploy-agent.aarch64.pkg`

버킷 이름 및 리전 식별자 목록은 리전별 리소스 키트 버킷 이름 단원을 참조하세요.

최신 버전의 CodeDeploy 에이전트를 설치하려면 다음을 수행합니다.

- ```
  sudo installer -pkg codedeploy-agent.aarch64.pkg -target /
  ```

특정 버전의 CodeDeploy 에이전트를 설치하려면 다음을 수행합니다.

- 해당 리전에서 사용 가능한 버전을 나열합니다.

  ```
  aws s3 ls s3://aws-codedeploy-region-identifier/releases/ --region region-identifier
   | grep '\.pkg$'
  ```

- 버전 중 하나를 다운로드하여 설치합니다.

  ```
  aws s3 cp s3://aws-codedeploy-region-identifier/releases/codedeploy-
  agent-version.aarch64.pkg codedeploy-agent.aarch64.pkg --region region-identifier
  ```

  ```
  sudo installer -pkg codedeploy-agent.aarch64.pkg -target /
  ```

  최신 릴리스 버전은 섹션을 참조하세요CodeDeploy 에이전트의 버전 기록.

서비스가 실행 중인지 확인하려면 다음 명령을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent status
```

에이전트가 실행 중인 경우 다음과 같은 메시지가 표시됩니다. `The AWS CodeDeploy agent is running.`

에이전트가 실행되지 않는 경우 서비스를 시작하고 상태를 확인합니다.

```
sudo launchctl kickstart system/com.amazon.aws.codedeploy-agent
```

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent status
```

CodeDeploy 에이전트 업데이트

AWS Systems Manager를 사용하여 지원되는 모든 운영 체제에서 CodeDeploy 에이전트의 자동 예약 업데이트를 구성할 수 있습니다. 또한 에이전트에서 명령을 실행하여 지원되는 모든 운영 체제에 대한 업데이트를 강제로 수행할 수도 있습니다.
```

주제

- Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 업데이트
- Ubuntu Server에서 CodeDeploy 에이전트 업데이트
- Windows Server에서 CodeDeploy 에이전트 업데이트
- macOS에서 CodeDeploy 에이전트 업데이트

Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 업데이트

를 사용하여 CodeDeploy 에이전트의 예약된 자동 업데이트를 구성하려면 를 사용하여 CodeDeploy 에이전트 설치 AWS Systems Manager의 단계를 AWS Systems Manager따릅니다.

버전 2.0.x 이상의 경우 업데이트를 강제 적용하려면 다음을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent update
```

또는 리전 `latestv2/install` 스크립트를 다시 다운로드하여 실행합니다. 설치 지침은 Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치을 확인하세요.

버전 1.8.x 이하의 경우 업데이트를 강제 적용하려면 다음을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/install auto
```

> **Important**
>
> 현재 버전 1.8.x에서 2.0.0으로 자동 업데이트 경로가 없습니다. 2.0.0으로 업그레이드하려면 `AWSCodeDeployAgentV2` Systems Manager Distributor 패키지를 사용하여 설치하거나 리전 `latestv2/install` 스크립트를 수동으로 실행합니다. 지침은 CodeDeploy 에이전트 설치 섹션을 참조하세요.

Ubuntu Server에서 CodeDeploy 에이전트 업데이트

를 사용하여 CodeDeploy 에이전트의 예약된 자동 업데이트를 구성하려면 를 사용하여 CodeDeploy 에이전트 설치 AWS Systems Manager의 단계를 AWS Systems Manager따릅니다.

버전 2.0.x 이상의 경우 업데이트를 강제 적용하려면 다음을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent update
```

또는 리전 `latestv2/install` 스크립트를 다시 다운로드하여 실행합니다. 설치 지침은 Ubuntu Server용 CodeDeploy 에이전트 설치을 확인하세요.

버전 1.8.x 이하의 경우 업데이트를 강제 적용하려면 다음을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/install auto
```

> **Important**
>
> 현재 버전 1.8.x에서 2.0.0으로 자동 업데이트 경로가 없습니다. 2.0.0으로 업그레이드하려면 `AWSCodeDeployAgentV2` Systems Manager Distributor 패키지를 사용하여 설치하거나 리전 `latestv2/install` 스크립트를 수동으로 실행합니다. 지침은 CodeDeploy 에이전트 설치 섹션을 참조하세요.

Windows Server에서 CodeDeploy 에이전트 업데이트

를 사용하여 CodeDeploy 에이전트의 자동 업데이트를 활성화할 수 있습니다 AWS Systems Manager. Systems Manager에서 Systems Manager 상태 관리자와의 연결을 생성하여 Amazon EC2 또는 온프레미스 인스턴스에 업데이트 일정을 구성할 수 있습니다. 또한 현재 버전을 제거하고 최신 버전을 설치하여 CodeDeploy 에이전트를 수동으로 업데이트할 수도 있습니다.

주제

- 를 사용하여 자동 CodeDeploy 에이전트 업데이트 설정 AWS Systems Manager
- CodeDeploy 에이전트 수동 업데이트

를 사용하여 자동 CodeDeploy 에이전트 업데이트 설정 AWS Systems Manager

Systems Manager를 구성하고 CodeDeploy 에이전트의 자동 업데이트를 활성화하려면를 사용하여 CodeDeploy 에이전트 설치의 AWS Systems Manager 지침을 따릅니다.

CodeDeploy 에이전트 수동 업데이트

CodeDeploy 에이전트를 수동으로 업데이트하려는 경우 CLI 또는 Systems Manager를 사용하여 최신 버전을 설치할 수 있습니다. CodeDeploy 에이전트 설치의 지침을 따릅니다. CodeDeploy 에이전트 설치 제거 지침에 따라 이전 버전의 CodeDeploy 에이전트를 제거하는 것이 좋습니다.

macOS에서 CodeDeploy 에이전트 업데이트

를 사용하여 CodeDeploy 에이전트의 예약된 자동 업데이트를 구성하려면의 단계를 AWS Systems Manager따르고 `AWSCodeDeployAgentV2` 패키지를 를 사용하여 CodeDeploy 에이전트 설치 AWS Systems Manager 선택합니다.

업데이트를 강제로 실행하려면 다음을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/codedeploy-agent update
```

또는 최신 `.pkg` 파일을 다운로드하여 기존 에이전트 위에 설치합니다. 설치 지침은 macOS용 CodeDeploy 에이전트 설치을 확인하세요.

CodeDeploy 에이전트 설치 제거

더 이상 필요가 없거나 새로 설치하려는 경우 인스턴스에서 CodeDeploy 에이전트를 제거할 수 있습니다.

Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치 제거

CodeDeploy 에이전트를 제거하려면 인스턴스에 로그인하여 다음 명령을 실행합니다.

```
sudo yum erase codedeploy-agent
```

Ubuntu Server에 CodeDeploy 에이전트를 설치 제거하려면

CodeDeploy 에이전트를 제거하려면 인스턴스에 로그인하여 다음 명령을 실행합니다.

```
sudo dpkg --purge codedeploy-agent
```

Windows Server에서 CodeDeploy 에이전트를 설치 제거합니다.

CodeDeploy 에이전트를 제거하려면 인스턴스에 로그인하여 다음 3가지 명령을 한 번에 하나씩 실행합니다.

```
wmic
product where name="CodeDeploy Host Agent" call uninstall /nointeractive
exit
```

> **Note**
>
> Windows Server 2025 이상에서는 `wmic`가 더 이상 사용되지 않습니다. 대신 다음 PowerShell 명령을 사용합니다.
>
> ```
> Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall*' |
>   Where-Object DisplayName -eq 'CodeDeploy Host Agent' |
>   ForEach-Object { Start-Process msiexec.exe -ArgumentList "/x
>  $($_.PSChildName) /quiet" -Wait }
> ```

또한 인스턴스에 로그인하여 제어판에서 프로그램 및 기능을 열고 CodeDeploy 호스트 에이전트를 선택한 후 제거를 선택할 수도 있습니다.

macOS에서 CodeDeploy 에이전트 제거

CodeDeploy 에이전트를 제거하려면 인스턴스에 로그인하여 다음 명령을 실행합니다.

```
sudo /opt/codedeploy-agent/bin/uninstall.sh
```

CloudWatch로 CodeDeploy 에이전트 로그 전송

통합 CloudWatch 에이전트, 줄여서 CloudWatch 에이전트를 사용하여 CodeDeploy 에이전트 지표 및 로그 데이터를 CloudWatch로 전송할 수 있습니다.

CodeDeploy 에이전트와 함께 사용할 CloudWatch 에이전트를 설치하고 구성하려면 다음 지침을 따르세요.

사전 조건

시작하기 전에 다음 작업을 완료하세요.

- CodeDeploy 에이전트를 설치하고 실행 중인지 확인합니다. 자세한 내용은 CodeDeploy 에이전트 설치 및 CodeDeploy 에이전트가 실행 중인지 확인 섹션을 참조하세요.
- CloudWatch 에이전트를 설치합니다. 자세한 내용은 CloudWatch 에이전트 설치를 참조하세요.
- CodeDeploy IAM 인스턴스 프로파일에 다음 권한을 추가합니다.
  - CloudWatchLogsFullAccess

- CloudWatchAgentServerPolicy

CodeDeploy 인스턴스 프로파일에 대한 자세한 정보는 CodeDeploy 시작하기의 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 섹션을 참조하세요.

CodeDeploy 로그를 수집하도록 CloudWatch 에이전트 구성

마법사를 단계별로 진행하거나 구성 파일을 수동으로 생성 또는 편집하여 CloudWatch 에이전트를 구성할 수 있습니다.

마법사를 사용하여 CloudWatch 에이전트 구성(Linux)

1. CloudWatch 에이전트 구성 마법사 실행에 설명된 대로 마법사를 실행합니다.
2. 마법사에서 “Do you want to monitor any log files?”라는 질문이 표시되면 **1**을 입력합니다.
3. 다음과 같이 CodeDeploy 에이전트 로그 파일을 지정합니다.
    1. Log file path에 CodeDeploy 로그 파일의 경로를 입력합니다(예: **/var/log/aws/codedeploy-agent/codedeploy-agent.log**).
    2. Log group name에 로그 그룹 이름을 입력합니다(예: **codedeploy-agent-log**).
    3. Log stream name에 로그 스트림 이름을 입력합니다(예: **{instance_id}-codedeploy-agent-log**).
4. “Do you want to specify any additional log files?”라는 질문이 표시되면 **1**을 입력합니다.
5. 다음과 같이 CodeDeploy 에이전트 배포 로그를 지정합니다.
    1. Log file path에 CodeDeploy 배포 로그 파일의 경로를 입력합니다(예: **/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log**).
    2. Log group name에 로그 그룹 이름을 입력합니다(예: **codedeploy-agent-deployment-log**).
    3. Log stream name에 로그 스트림 이름을 입력합니다(예: **{instance_id}-codedeploy-agent-deployment-log**).
6. “Do you want to specify any additional log files?”라는 질문이 표시되면 **1**을 입력합니다.
7. 다음과 같이 CodeDeploy 에이전트 업데이트 도구 로그를 지정합니다.

1. 에 CodeDeploy 업데이터 로그 파일의 경로를 `Log file path`입력합니다. 경로는 에이전트 버전에 따라 다릅니다.
   - 버전 2.0.x 이상에서는를 사용합니다**/var/log/aws/codedeploy-agent/codedeploy-agent-updater.log**.
   - 버전 1.8.x 이하의 경우를 사용합니다**/tmp/codedeploy-agent.update.log**.
2. `Log group name`에 로그 그룹 이름을 입력합니다(예: **codedeploy-agent-updater-log**).
3. `Log stream name`에 로그 스트림 이름을 입력합니다(예: **{instance_id}-codedeploy-agent-updater-log**).

마법사를 사용하여 CloudWatch 에이전트 구성(Windows)

1. CloudWatch 에이전트 구성 마법사 실행에 설명된 대로 마법사를 실행합니다.
2. 마법사에서 "`Do you want to monitor any customized log files?`"라는 질문이 표시되면 **1**을 입력합니다.
3. 다음과 같이 CodeDeploy 로그 파일을 지정합니다.
   1. 에 CodeDeploy 에이전트 로그 파일의 경로를 `Log file path`입력합니다. 경로는 에이전트 버전에 따라 다릅니다.
      - 버전 2.0.x 이상에서는를 사용합니다**C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent.log**.
      - 버전 1.8.x 이하의 경우를 사용합니다**C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt**.
   2. `Log group name`에 로그 그룹 이름을 입력합니다(예: **codedeploy-agent-log**).
   3. `Log stream name`에 로그 스트림 이름을 입력합니다(예: **{instance_id}-codedeploy-agent-log**).
4. "`Do you want to specify any additional log files?`"라는 질문이 표시되면 **1**을 입력합니다.
5. 다음과 같이 CodeDeploy 에이전트 배포 로그를 지정합니다.
   1. 에 CodeDeploy 배포 로그 파일의 경로를 `Log file path`입력합니다**C:\ProgramData\Amazon\CodeDeploy\deployment-logs\codedeploy-agent-deployments.log**.
   2. `Log group name`에 로그 그룹 이름을 입력합니다(예: **codedeploy-agent-deployment-log**).

3. Log stream name에 로그 스트림 이름을 입력합니다(예: **{instance_id}-codedeploy-agent-deployment-log**).

구성 파일을 수동으로 생성 또는 편집하여 CloudWatch 에이전트 구성(Linux)

1. 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집에 설명된 대로 CloudWatch 에이전트 구성 파일을 생성 또는 편집합니다.
2. 파일 이름이 /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json이고 다음 코드를 포함해야 합니다.

   업데이트 프로그램 로그 경로는 버전에 따라 다릅니다. 버전 2.0.x 이상에서는를 사용합니다/var/log/aws/codedeploy-agent/codedeploy-agent-updater.log. 버전 1.8.x 이하의 경우를 사용합니다/tmp/codedeploy-agent.update.log.

```
...
"logs": {
    "logs_collected": {
        "files": {
            "collect_list": [
                {
                    "file_path": "/var/log/aws/codedeploy-agent/codedeploy-
agent.log",
                    "log_group_name": "codedeploy-agent-log",
                    "log_stream_name": "{instance_id}-agent-log"
                },
                {
                    "file_path": "/opt/codedeploy-agent/deployment-root/deployment-
logs/codedeploy-agent-deployments.log",
                    "log_group_name": "codedeploy-agent-deployment-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-deployment-
log"
                },
                {
                    "file_path": "/var/log/aws/codedeploy-agent/codedeploy-agent-
updater.log",
                    "log_group_name": "codedeploy-agent-updater-log",
                    "log_stream_name": "{instance_id}-codedeploy-agent-updater-log"
                }
            ]
        }
    }
```

```
}
...
```

구성 파일을 수동으로 생성 또는 편집하여 CloudWatch 에이전트 구성(Windows)

1. 수동으로 CloudWatch 에이전트 구성 파일 생성 또는 편집에 설명된 대로 CloudWatch 에이전트 구성 파일을 생성 또는 편집합니다.
2. 파일 이름이 `C:\ProgramData\Amazon\AmazonCloudWatchAgent\amazon-cloudwatch-agent.json`이고 다음 코드를 포함해야 합니다.

   버전 2.0.x 이상의 경우 에이전트 로그 파일 이름은 입니다`codedeploy-agent.log`. 버전 1.8.x 이하의 경우 에이전트 로그 파일 이름은 입니다`codedeploy-agent-log.txt`. 배포 로그 경로는 두 버전에서 동일합니다.

```
...
"logs": {
        "logs_collected": {
            "files": {
                "collect_list": [
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\\log\
\codedeploy-agent.log",
                        "log_group_name": "codedeploy-agent-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-log"
                    },
                    {
                        "file_path": "C:\\ProgramData\\Amazon\\CodeDeploy\
\deployment-logs\\codedeploy-agent-deployments.log",
                        "log_group_name": "codedeploy-agent-deployment-log",
                        "log_stream_name": "{instance_id}-codedeploy-agent-
deployment-log"
                    }
                ]
            },
            ...
        }
    },
...
```

CloudWatch 에이전트 재시작

변경한 후 CloudWatch 에이전트 시작에 설명된 대로 CloudWatch 에이전트를 다시 시작합니다.

CodeDeploy용 인스턴스 작업

CodeDeploy는 Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux(RHEL) 및 Windows Server를 실행하는 인스턴스에 대한 배포를 지원합니다.

CodeDeploy를 사용하여 Amazon EC2 인스턴스와 온프레미스 인스턴스 모두에 배포할 수 있습니다. 온프레미스 인스턴스는 CodeDeploy 에이전트를 실행하고 퍼블릭 AWS 서비스 엔드포인트에 연결할 수 있는 Amazon EC2 인스턴스가 아닌 모든 물리적 디바이스입니다. CodeDeploy를 사용하여 애플리케이션을 클라우드의 Amazon EC2 인스턴스와 사무실의 데스크톱 PC 또는 자체 데이터 센터의 서버에 동시에 배포할 수 있습니다.

Amazon EC2 인스턴스와 온프레미스 인스턴스 비교

다음 표에서는 Amazon EC2 인스턴스 및 온프레미스 인스턴스를 비교합니다.

Subject	Amazon EC2 인스턴스	온프레미스 인스턴스
인스턴스에서 실행 중인 운영 체제와 호환되는 CodeDeploy 에이전트 버전을 설치 및 실행해야 합니다.	예	예
인스턴스가 CodeDeploy에 연결할 수 있어야 합니다.	예	예
IAM 인스턴스 프로파일을 인스턴스에 연결해야 합니다. IAM 인스턴스 프로파일에는 CodeDeploy 배포에 참여할 수 있는 권한이 있어야 합니다. 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 단원을 참조하세요.	예	아니요
인스턴스를 인증하고 등록하려면 다음 중 하나를 수행해야 합니다.	아니요	예

Subject	Amazon EC2 인스턴스	온프레미스 인스턴스
• 각 인스턴스에서 IAM 사용자가 가정할 수 있는 IAM 역할을 생성하여 AWS Security Token Service을 통해 생성된 주기적으로 새로 고쳐진 임시 자격 증명을 검색합니다. • 각 인스턴스에 대해 IAM 사용자를 생성하고 IAM 사용자의 계정 자격 증명을 인스턴스에 일반 텍스트로 저장합니다.		
각 인스턴스에 배포하려면 먼저 CodeDeploy에 각 인스턴스를 등록해야 합니다.	아니요	예
CodeDeploy가 인스턴스에 배포할 수 있으려면 먼저 각 인스턴스에 태그를 지정해야 합니다.	예	예
CodeDeploy 배포의 일부로 Amazon EC2 Auto Scaling 및 Elastic Load Balancing 시나리오에 참여할 수 있습니다.	예	아니요
Amazon S3 버킷 및 GitHub 리포지토리에서 배포할 수 있습니다.	예	예
배포 또는 인스턴스에서 지정된 이벤트가 발생할 때 SMS 또는 이메일 알림을 보내도록 요청하는 트리거를 지원할 수 있습니다.	예	예

Subject	Amazon EC2 인스턴스	온프레미스 인스턴스
연결된 배포에 대해 요금이 청구될 수 있습니다.	아니요	예

CodeDeploy에 대한 인스턴스 작업

배포에 사용할 인스턴스를 시작하거나 구성하려면 다음 지침 중에서 선택합니다.

새 Amazon Linux 또는 Windows Server Amazon EC2 인스턴스를 시작하려고 합니다.	최소한의 작업으로 Amazon EC2 인스턴스를 시작하려면 CodeDeploy용 Amazon EC2 인스턴스 생성(CloudFormation 템플릿) 단원을 참조하세요. 주로 사용자가 직접 Amazon EC2 인스턴스를 시작하려면 CodeDeploy용 Amazon EC2 인스턴스 생성(AWS CLI 또는 Amazon EC2 콘솔) 단원을 참조하세요.
새로운 Ubuntu Server 또는 RHEL Amazon EC2 인스턴스를 시작하려고 합니다.	CodeDeploy용 Amazon EC2 인스턴스 생성(AWS CLI 또는 Amazon EC2 콘솔)을(를) 참조하세요.
Amazon Linux, Windows Server, Ubuntu Server 또는 RHEL Amazon EC2 인스턴스를 구성하려고 합니다.	CodeDeploy 작업을 위한 Amazon EC2 인스턴스 구성을(를) 참조하세요.
Windows 서버, Ubuntu Server 또는 RHEL 온프레미스 인스턴스(Amazon EC2 인스턴스가 아닌 물리적 디바이스)를 구성하려고 합니다.	Working with On-Premises Instances을(를) 참조하세요.
CodeDeploy가 블루/그린 배포 중에 대체 인스턴스 집합을 프로비저닝하기를 원합니다.	CodeDeploy에서 배포 작업을(를) 참조하세요.

Amazon EC2 Auto Scaling 그룹에서 Amazon EC2 인스턴스를 준비하려면 몇 가지 추가 단계를 수행해야 합니다. 자세한 내용은 Amazon EC2 Auto Scaling과 CodeDeploy 통합 단원을 참조하십시오.

주제

- Tagging Instances for Deployments
- Working with Amazon EC2 Instances
- Working with On-Premises Instances
- View Instance Details
- Instance Health

CodeDeploy에서 배포 그룹에 대한 인스턴스에 태그 지정

태그를 사용하여 각 리소스에 고유의 메타데이터를 할당하면 Amazon EC2 인스턴스 및 온프레미스 인스턴스를 손쉽게 관리할 수 있습니다. 태그를 사용하면 인스턴스를 다양한 방식(예: 용도, 소유자 또는 환경을 기준으로)으로 분류할 수 있습니다. 이는 인스턴스가 많을 때 유용합니다. 인스턴스에 할당한 태그를 기준으로 인스턴스 또는 인스턴스 그룹을 손쉽게 식별할 수 있습니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다. 자세한 내용은 Amazon EC2 리소스에 태그 지정을 참조하세요.

CodeDeploy 배포 그룹에 포함시킬 인스턴스를 지정하려면 하나 이상의 태그 그룹에서 태그를 지정합니다. 태그 기준을 충족하는 인스턴스는 해당 배포 그룹에 대한 배포가 만들어질 때 최신 애플리케이션 개정이 설치된 인스턴스입니다.

> **Note**
>
> 배포 그룹에 Amazon EC2 Auto Scaling 그룹을 포함시킬 수도 있지만, 인스턴스에 적용된 태그가 아닌 이름으로 식별됩니다. 자세한 내용은 Amazon EC2 Auto Scaling과 CodeDeploy 통합 단원을 참조하세요.

배포 그룹의 인스턴스 기준은 한 태그 그룹의 태그 한 개만큼 단순하거나, 최대 세 태그 그룹의 각 10개의 태그만큼 복잡할 수도 있습니다.

한 태그 그룹을 사용하는 경우 그룹의 하나 이상의 태그로 식별되는 모든 인스턴스가 배포 그룹에 포함됩니다. 여러 태그 그룹을 사용하는 경우 각 태그 그룹의 하나 이상의 태그로 식별되는 인스턴스만 포함됩니다.

다음 예에서는 태그와 태그 그룹을 사용하여 배포 그룹에 대한 인스턴스를 선택하는 방법을 보여 줍니다.

주제

- 예 1: 한 태그 그룹, 한 태그
- 예 2: 한 태그 그룹, 여러 태그
- 예 3: 여러 태그 그룹, 한 태그
- 예 4: 여러 태그 그룹, 여러 태그

예 1: 한 태그 그룹, 한 태그

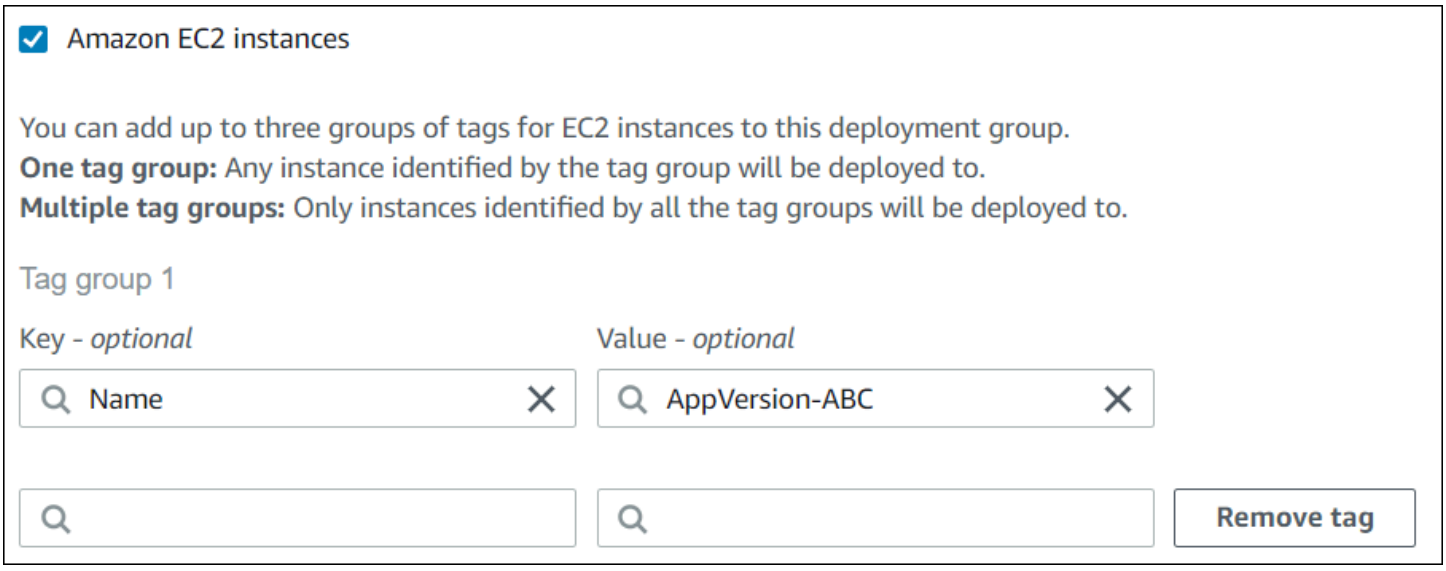
한 태그 그룹에서 한 태그를 지정할 수 있습니다.

태그 그룹 1

Key(키)	값
이름	AppVersion-ABC

`Name=AppVersion-ABC`로 태그 지정된 각 인스턴스는 다른 태그가 적용되었을지라도 배포 그룹의 일부입니다.

CodeDeploy 콘솔 설정 보기:



JSON 구조:

```
"ec2TagSet": {
```

```
        "ec2TagSetList": [
           [
              {
                 "Type": "KEY_AND_VALUE",
                 "Key": "Name",
                 "Value": "AppVersion-ABC"
              }
           ]
        ]
     },
```

예 2: 한 태그 그룹, 여러 태그

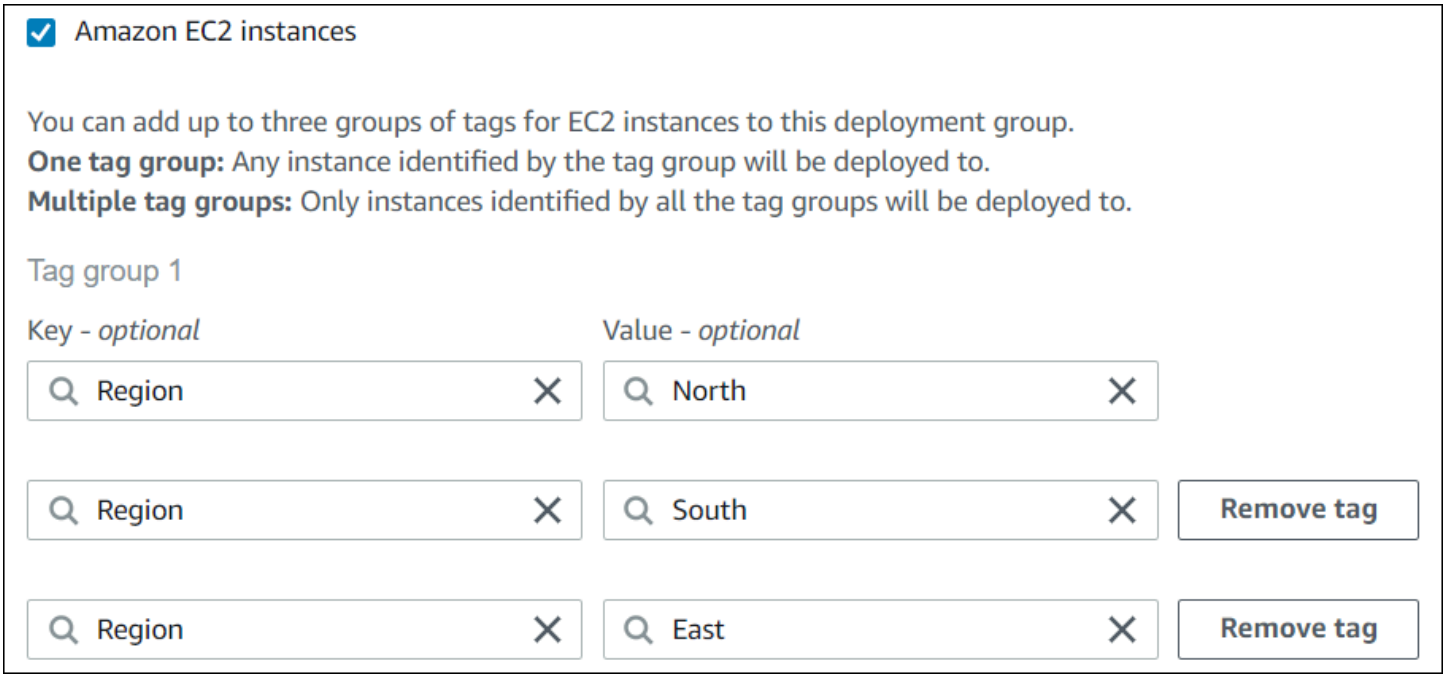
한 태그 그룹에서 여러 태그를 지정할 수도 있습니다.

태그 그룹 1

Key(키)	값
리전	North
리전	South
리전	East

이러한 세 개의 태그 중 하나로 태그 지정된 인스턴스는 다른 태그가 적용되었을지라도 배포 그룹의 일부입니다. 예를 들어, `Region=West`로 태그가 지정된 인스턴스는 배포 그룹에 포함되지 않습니다.

CodeDeploy 콘솔 설정 보기:



JSON 구조:

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               },
                           {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "South"
               },
                           {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "East"
               }
            ]
         ]
      },
```

예 3: 여러 태그 그룹, 한 태그

각각에 키-값 페어 한 개가 있는 여러 태그 그룹 세트를 사용하여 배포 그룹의 인스턴스에 대한 기준을 지정할 수도 있습니다. 배포 그룹에서 여러 태그 그룹을 사용하는 경우 모든 태그 그룹으로 식별되는 인스턴스만 배포 그룹에 포함됩니다.

태그 그룹 1

Key(키)	값
이름	AppVersion-ABC

태그 그룹 2

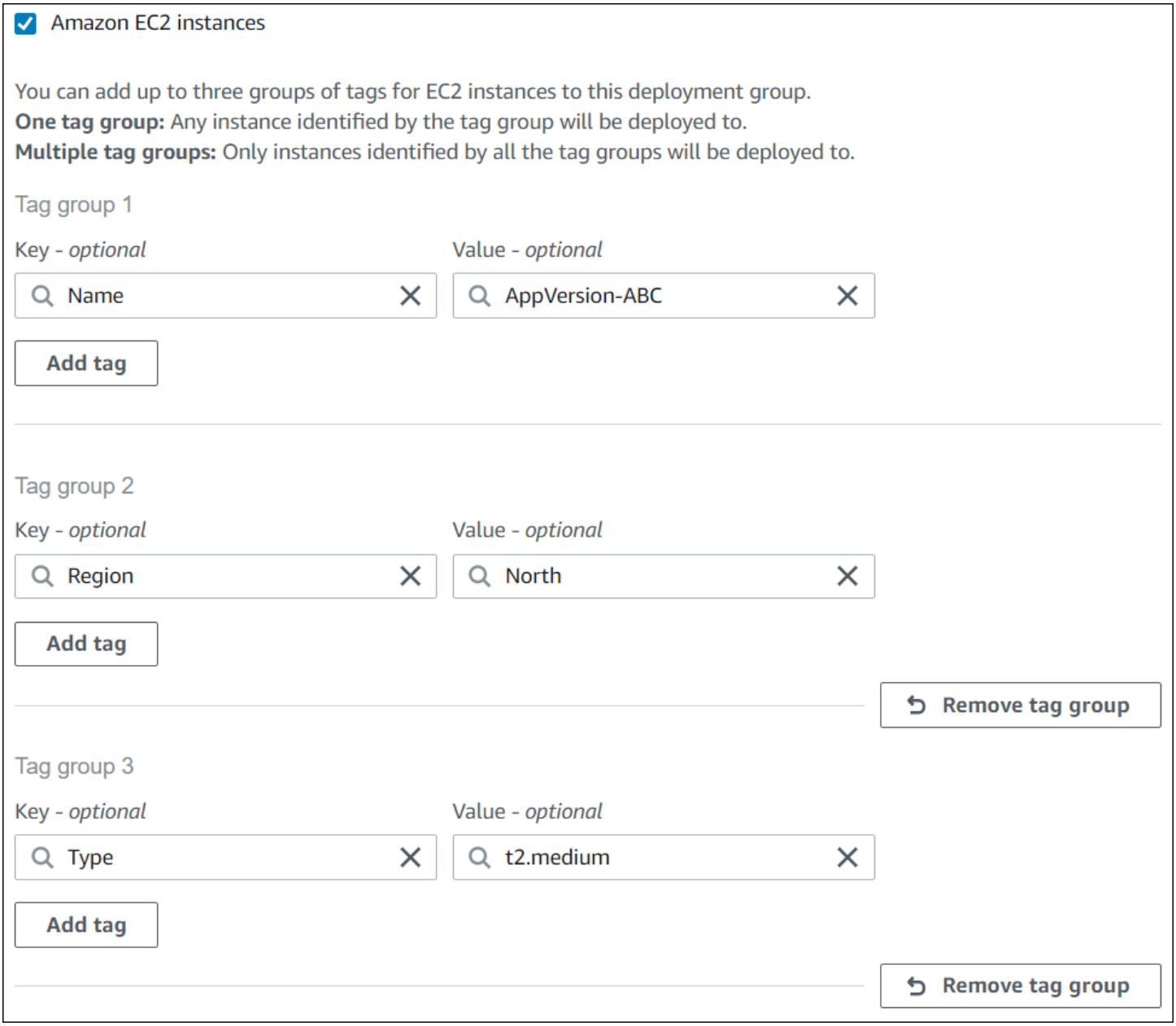
Key(키)	값
리전	North

태그 그룹 3

Key(키)	값
유형	t2.medium

여러 리전에 인스턴스가 있고 로 태그가 지정된 다양한 인스턴스 유형이 있을 수 있습니다Name=AppVersion-ABC. 이 예에서는 Region=North 및 Type=t2.medium으로도 태그 지정된 인스턴스만 배포 그룹의 일부입니다.

CodeDeploy 콘솔 설정 보기:



JSON 구조:

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Name",
                  "Value": "AppVersion-ABC"
               }
            ],
```

```
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.medium"
               }
            ],
         ]
      },
```

예 4: 여러 태그 그룹, 여러 태그

하나 이상의 그룹에서 여러 태그가 있는 여러 태그 그룹을 사용하는 경우, 인스턴스는 각 그룹의 태그 중 적어도 하나와 일치해야 합니다.

태그 그룹 1

Key(키)	값
환경	베타
환경	Staging

태그 그룹 2

Key(키)	값
리전	North
리전	South
리전	East

태그 그룹 3

Key(키)	값
유형	t2.medium
Type	t2.large

이 예에서 배포 그룹에 포함되려면 인스턴스는 (1) `Environment=Beta` 또는 `Environment=Staging`, (2) `Region=North`, `Region=South` 또는 `Region=East`, (3) `Type=t2.medium` 또는 `Type=t2.large`로 태그 지정되어야 합니다.

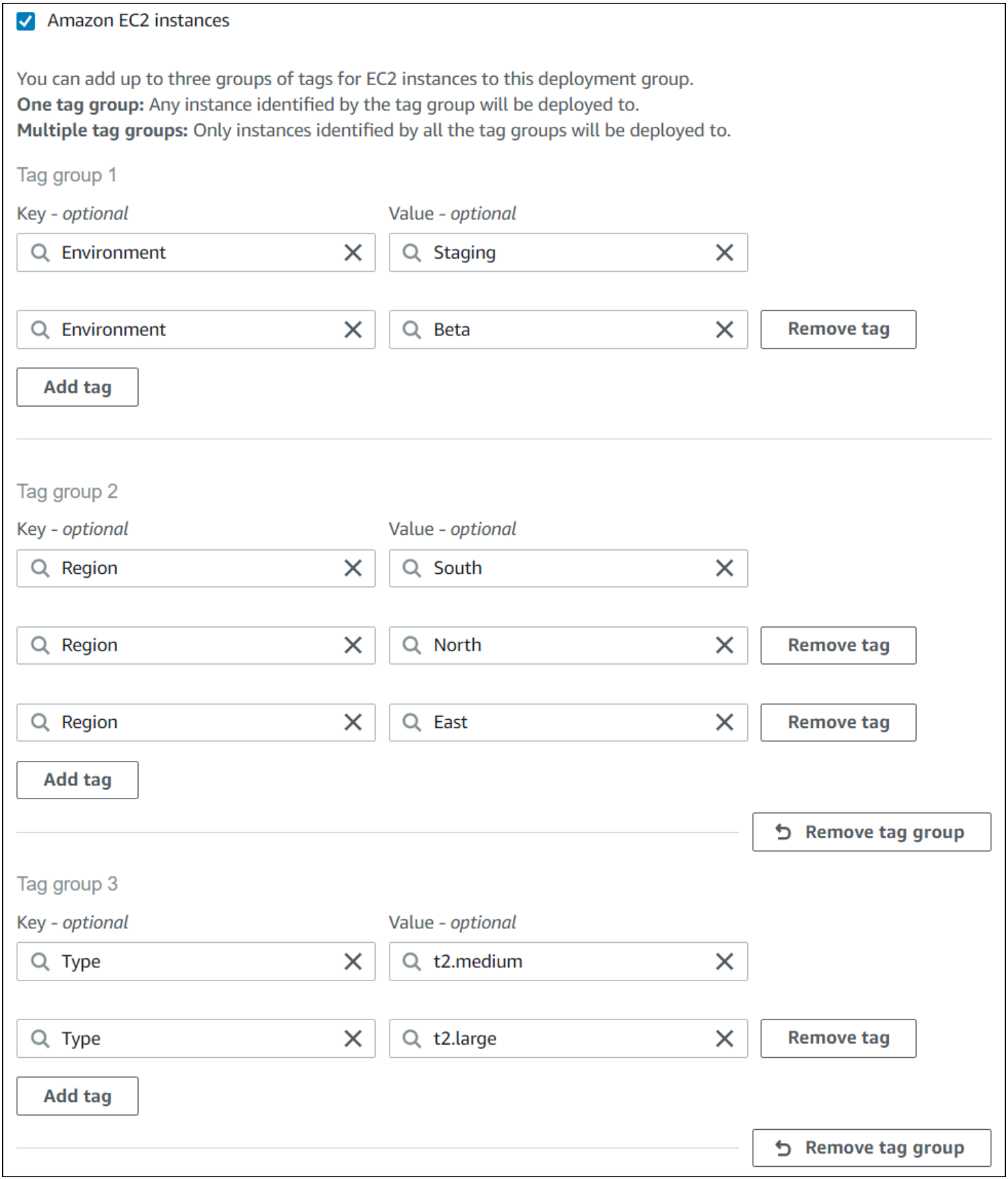
예를 들어, 다음 태그 그룹이 있는 인스턴스는 배포 그룹에 포함되는 인스턴스입니다.

- `Environment=Beta`, `Region=North`,`Type=t2.medium`
- `Environment=Staging`,`Region=East`,`Type=t2.large`
- `Environment=Staging`,`Region=South`,`Type=t2.large`

다음 태그 그룹이 있는 인스턴스는 배포 그룹에 포함되지 않습니다. 강조 표시된 키 값으로 인해 인스턴스가 제외됩니다.

- `Environment=Beta`, 리전=West,`Type=t2.medium`
- `Environment=Staging`, `Region=East`, 유형=t2.micro
- 환경=Production,`Region=South`,`Type=t2.large`

CodeDeploy 콘솔 설정 보기:


Amazon EC2 instances
You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.
Tag group 1
Key - optional
Value - optional
Environment
Staging
Environment
Beta
Remove tag
Add tag
Tag group 2
Key - optional
Value - optional
Region
South
Region
North
Remove tag
Region
East
Remove tag
Add tag
Remove tag group
Tag group 3
Key - optional
Value - optional
Type
t2.medium
Type
t2.large
Remove tag
Add tag
Remove tag group

JSON 구조:

```
"ec2TagSet": {
         "ec2TagSetList": [
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Beta"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Environment",
                  "Value": "Staging"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "North"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "South"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Region",
                  "Value": "East"
               }
            ],
            [
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.medium"
               },
               {
                  "Type": "KEY_AND_VALUE",
                  "Key": "Type",
                  "Value": "t2.large"
               }
```

```
            ],
        ]
    },
```

CodeDeploy용 Amazon EC2 인스턴스 작업

Amazon EC2 인스턴스는 Amazon Elastic Compute Cloud를 사용하여 생성하고 구성하는 가상 컴퓨팅 환경입니다. Amazon EC2는 AWS 클라우드에서 확장 가능한 컴퓨팅 용량을 제공합니다. Amazon EC2를 사용하여 CodeDeploy 배포에 필요한 만큼 많거나 적은 수의 가상 서버를 시작할 수 있습니다.

Amazon EC2에 대한 자세한 내용은 Amazon EC2 시작 가이드를 참조하세요.

다음 지침에서는 CodeDeploy 배포에 사용하도록 구성된 Amazon EC2 인스턴스를 생성하고 구성하는 방법을 안내합니다.

주제

- CodeDeploy용 Amazon EC2 인스턴스 생성(AWS CLI 또는 Amazon EC2 콘솔)
- CodeDeploy용 Amazon EC2 인스턴스 생성(CloudFormation 템플릿)
- CodeDeploy 작업을 위한 Amazon EC2 인스턴스 구성

CodeDeploy용 Amazon EC2 인스턴스 생성(AWS CLI 또는 Amazon EC2 콘솔)

다음 지침은 CodeDeploy 배포에 사용하도록 구성된 새 Amazon EC2 인스턴스를 시작하는 방법을 안내합니다.

CloudFormation 템플릿을 사용하여 CodeDeploy 배포에 사용하도록 이미 구성된 Amazon Linux 또는 Windows Server를 실행하는 Amazon EC2 인스턴스를 시작할 수 있습니다. Ubuntu Server 또는 Red Hat Enterprise Linux(RHEL)를 실행하는 Amazon EC2 인스턴스에는 CloudFormation 템플릿을 제공하지 않습니다. 템플릿 사용 대신 적용할 수 있는 방법은 CodeDeploy용 인스턴스 작업 단원을 참조하세요.

Amazon EC2 콘솔 AWS CLI또는 Amazon EC2 APIs를 사용하여 Amazon EC2 인스턴스를 시작할 수 있습니다.

Amazon EC2 인스턴스 시작(콘솔)

사전 조건

아직 수행하지 않은 경우의 지침에 따라 CodeDeploy 시작하기를 설정 및 구성 AWS CLI 하고 IAM 인스턴스 프로파일을 생성합니다.

Amazon EC2 인스턴스 시작하기

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 인스턴스를 선택한 후 인스턴스 시작을 선택합니다.
3. 1단계: Amazon Machine Image(AMI) 선택 페이지의 빠른 시작 탭에서 사용하려는 운영 체제와 버전을 찾은 다음 선택을 선택합니다. CodeDeploy에서 지원하는 Amazon EC2 AMI 운영 체제를 선택해야 합니다. 자세한 내용은 CodeDeploy 에이전트에서 지원하는 운영 체제 단원을 참조하십시오.
4. 2단계: 인스턴스 유형 선택(Step 2: Choose an Instance Type) 페이지에서 사용 가능한 Amazon EC2 인스턴스 유형을 선택한 후 다음: 인스턴스 세부 정보 구성(Next: Configure Instance Details)을 선택합니다.
5. 3단계: 인스턴스 세부 정보 구성(Step 3: Configure Instance Details) 페이지의 IAM 역할(IAM role) 목록에서 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기에서 생성한 IAM 인스턴스 역할을 선택합니다. 제안된 역할 이름을 사용한 경우 **`CodeDeployDemo-EC2-Instance-Profile`**을 선택합니다. 고유의 역할 이름을 생성한 경우 해당 이름을 선택합니다.

   > **Note**
   >
   > 기본 Virtual Private Cloud(VPC)가 Network(네트워크) 목록에 표시되지 않는 경우 Amazon VPC 및 서브넷을 선택하거나 생성해야 합니다. 새 VPC 생성이나 새 서브넷 생성 또는 둘 다 선택합니다. 자세한 내용은 VPC 및 서브넷을 참조하세요.

6. 다음: 스토리지 추가를 선택합니다.
7. 4단계: 스토리지 추가 페이지의 설정을 그대로 두고 다음: 태그 추가를 선택합니다.
8. 5단계: 태그 추가 페이지에서 태그 추가를 선택합니다.
9. 키 상자에 **`Name`**을 입력합니다. 값 상자에 **`CodeDeployDemo`**를 입력합니다.

> **Important**
>
> 키 및 값 상자의 내용은 대/소문자를 구분합니다.

10. 다음: 보안 그룹 구성(Next: Configure Security Group)을 선택합니다.
11. 6단계: 보안 그룹 구성 페이지에서 새 보안 그룹 생성 옵션을 선택된 채로 둡니다.

    SSH 역할 기본값은 Amazon Linux, Ubuntu Server 또는 RHEL을 실행하는 Amazon EC2 인스턴스로 구성되어 있습니다. RDP 역할 기본값은 Windows Server를 실행하는 Amazon EC2 인스턴스로 구성되어 있습니다.
12. HTTP 포트를 열려면 규칙 추가 버튼을 선택하고 유형 드롭다운 목록에서 **HTTP**를 선택합니다. 소스의 기본값 사용자 지정 0.0.0.0/0을 그대로 두고, 검토 및 시작을 선택합니다.

> **Note**
>
> 프로덕션 환경에서는 위치 무관 0.0.0.0/0(Anywhere 0.0.0.0/0)을 지정하는 대신 SSH, RDP 및 HTTP 포트에 대한 액세스를 제한하는 것이 좋습니다. CodeDeploy는 무제한 포트 액세스와 HTTP 액세스가 필요하지 않습니다. 자세한 내용은 Amazon EC2 인스턴스의 보안 유지를 위한 팁을 참조하세요.

    범용(SSD)에서 부팅 대화 상자가 표시되면 해당 지침을 따른 후 다음을 선택합니다.
13. 7단계: 인스턴스 시작 검토 페이지의 설정을 그대로 두고 시작을 선택합니다.
14. 기존 키 페어 선택 또는 새 키 페어 생성 대화 상자에서 기존 키 페어 선택 또는 새 키 페어 생성을 선택합니다. Amazon EC2 인스턴스 키 페어를 이미 구성했다면 여기에서 선택할 수 있습니다.

    아직 Amazon EC2 인스턴스 키 페어가 없다면 새 키 페어 생성(Create a new key pair)을 선택하고 식별 가능한 이름을 지정합니다. 키 페어 다운로드(Download Key Pair)를 선택하여 컴퓨터에 Amazon EC2 인스턴스 키 페어를 다운로드합니다.

> **Important**
>
> SSH 또는 RDP를 사용하여 Amazon EC2 인스턴스에 액세스하려면 키 페어가 있어야 합니다.

15. 인스턴스 시작(Launch Instances)을 선택합니다.

16. Amazon EC2 인스턴스의 ID를 선택합니다. 인스턴스가 시작되어 모든 검사를 통과할 때까지 기다리세요.

CodeDeploy 에이전트 설치

CodeDeploy 배포에서 CodeDeploy 에이전트를 사용하기 전에 Amazon EC2 인스턴스에 에이전트를 설치해야 합니다. 자세한 내용은 CodeDeploy 에이전트 설치 단원을 참조하십시오.

> **Note**
>
> 콘솔에서 배포 그룹을 생성할 때 CodeDeploy 에이전트의 자동 설치 및 업데이트를 구성할 수 있습니다.

Amazon EC2 인스턴스 시작하기(CLI)

사전 조건

아직 수행하지 않은 경우의 지침에 따라 CodeDeploy 시작하기를 설정 및 구성 AWS CLI 하고 IAM 인스턴스 프로파일을 생성합니다.

Amazon EC2 인스턴스 시작하기

1. Windows Server만 해당: Windows Server를 실행하여 Amazon EC2 인스턴스를 만드는 경우 create-security-group 및 authorize-security-group-ingress 명령을 호출하여 RDP 액세스(기본값으로 허용되지 않음) 또는 HTTP 액세스를 허용하는 보안 그룹을 만듭니다. 예를 들어 CodeDeployDemo-Windows-Security-Group이라는 보안 그룹을 만들려면 다음 명령을 한 번에 하나씩 실행합니다.

```
aws ec2 create-security-group --group-name CodeDeployDemo-Windows-Security-Group --
description "For launching Windows Server images for use with CodeDeploy"
```

```
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
Security-Group --to-port 3389 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port
 3389
```

```
aws ec2 authorize-security-group-ingress --group-name CodeDeployDemo-Windows-
Security-Group --to-port 80 --ip-protocol tcp --cidr-ip 0.0.0.0/0 --from-port 80
```

> **Note**
>
> 데모용으로 이 명령은 포트 3389를 통한 RDP 또는 포트 80을 통한 HTTP에 대한 무제한 액세스를 허용하는 보안 그룹을 만듭니다. RDP 및 HTTP 포트에 대한 액세스를 제한하는 것을 모범 사례로 권장합니다. CodeDeploy는 무제한 포트 액세스와 HTTP 액세스가 필요하지 않습니다. 자세한 내용은 Amazon EC2 인스턴스의 보안 유지를 위한 팁을 참조하세요.

2. run-instances 명령을 호출하여 Amazon EC2 인스턴스를 생성하고 시작합니다.

   이 명령을 호출하기 전에 다음을 수집해야 합니다.

   - 인스턴스에 사용할 Amazon Machine Image(AMI)의 ID(*ami-id*). ID를 확인하려면 적합한 AMI 찾기를 참조하세요.
   - 생성할 Amazon EC2 인스턴스 유형(*instance-type*)의 이름(예: `t1.micro`). Amazon EC2 인스턴스 유형에서 목록을 확인하세요.
   - 리전의 CodeDeploy 에이전트 설치 파일이 저장된 Amazon S3 버킷에 액세스할 수 있는 권한이 있는 IAM 인스턴스 프로파일의 이름입니다.

     IAM 인스턴스 프로파일의 생성에 대한 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 섹션을 참조하세요.
   - Amazon Linux, Ubuntu Server를 실행하는 Amazon EC2 인스턴스에 대한 SSH 액세스 또는 Windows Server를 실행하는 Amazon EC2 인스턴스에 대한 RHEL 또는 RDP 액세스를 가능하게 하는 Amazon EC2 인스턴스 키 페어의 이름(*key-name*).

     > **Important**
     >
     > 키 페어 파일 확장명은 제외하고 키 페어 이름만 입력합니다. 예를 들어 my-keypair.pem이 아니라 my-keypair로 입력합니다.

     키 페어 이름을 확인하려면 https://console.aws.amazon.com/ec2에서 Amazon EC2 콘솔을 엽니다. 탐색 창의 네트워크 및 보안에서 키 페어를 선택하고 목록에 표시되는 키 페어의 이름을 기록해 둡니다.

키 페어를 생성하려면 Amazon EC2를 사용하여 키 페어 만들기를 참조하세요. 리전에 나열된 리전 중 하나와의 엔드포인트에서 키 페어를 생성해야 합니다AWS 일반 참조. 그렇지 않으면 CodeDeploy에서 Amazon EC2 인스턴스 키 페어를 사용할 수 없습니다.

Amazon Linux, RHEL 및 Ubuntu Server용

run-instances 명령을 호출하여 Amazon Linux, Ubuntu Server 또는 RHEL을 실행하는 Amazon EC2 인스턴스를 시작하고 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기에서 생성된 IAM 인스턴스 프로파일을 연결합니다. 예:

```
aws ec2 run-instances \
  --image-id ami-id \
  --key-name key-name \
  --count 1 \
  --instance-type instance-type \
  --iam-instance-profile Name=iam-instance-profile
```

> **Note**
>
> 이 명령은 Amazon EC2 인스턴스에 대해 여러 포트에 대한 액세스를 허용하는 보안 그룹 기본값을 만듭니다. 여기에는 SSH의 경우 포트 22, HTTP의 경우 포트 80을 통한 무제한 액세스가 포함됩니다. RDP 및 HTTP 포트에 대한 액세스를 제한하는 것을 모범 사례로 권장합니다. CodeDeploy에는 무제한 포트 액세스와 HTTP 포트 액세스가 필요하지 않습니다. 자세한 내용은 Amazon EC2 인스턴스의 보안 유지를 위한 팁을 참조하세요.

Windows Server용

run-instances 명령을 호출하여 Windows Server를 실행하는 Amazon EC2 인스턴스를 시작하고 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기에서 생성한 IAM 인스턴스 프로파일을 연결하며, 1단계에서 생성한 보안 그룹의 이름을 지정합니다. 예:

```
aws ec2 run-instances --image-id ami-id --key-name key-name --count 1 --instance-
type instance-type --iam-instance-profile Name=iam-instance-profile --security-
groups CodeDeploy-Windows-Security-Group
```

이 명령은 지정된 IAM 인스턴스 프로파일에 따라 지정된 AMI, 키 페어, 인스턴스 유형으로 단일 Amazon EC2 인스턴스를 시작하고, 시작 시 지정된 스크립트를 실행합니다.

3. 출력에서 `InstanceID` 값을 기록해 둡니다. 이 값을 잊어버린 경우 Amazon EC2 인스턴스 키 페어에 대해 describe-instances 명령을 호출하여 나중에 확인할 수 있습니다.

```
aws ec2 describe-instances --filters "Name=key-name,Values=keyName" --query
 "Reservations[*].Instances[*].[InstanceId]" --output text
```

인스턴스 ID를 사용하여 create-tags 명령을 호출합니다. 이 명령은 Amazon EC2 인스턴스에 태그를 지정하여 이후 배포 중에 CodeDeploy가 인스턴스를 찾을 수 있도록 합니다. 다음 예제에서 태그 이름이 **CodeDeployDemo**지만, 원하는 대로 Amazon EC2 인스턴스 태그를 지정할 수 있습니다.

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=CodeDeployDemo
```

하나의 인스턴스에 태그 여러 개를 동시에 적용할 수 있습니다. 예:

```
aws ec2 create-tags --resources instance-id --tags Key=Name,Value=testInstance
 Key=Region,Value=West Key=Environment,Value=Beta
```

Amazon EC2 인스턴스가 시작되어 모든 검사를 통과했는지 확인하려면, 인스턴스 ID를 사용하여 describe-instance-status 명령을 호출합니다.

```
aws ec2 describe-instance-status --instance-ids instance-id --query
 "InstanceStatuses[*].InstanceStatus.[Status]" --output text
```

인스턴스가 시작되어 모든 검사를 통과한 경우 출력에 ok가 표시됩니다.

CodeDeploy 에이전트 설치

CodeDeploy 배포에서 CodeDeploy 에이전트를 사용하기 전에 Amazon EC2 인스턴스에 에이전트를 설치해야 합니다. 자세한 내용은 CodeDeploy 에이전트 설치 단원을 참조하십시오.

> **Note**
>
> 콘솔에서 배포 그룹을 생성할 때 CodeDeploy 에이전트의 자동 설치 및 업데이트를 구성할 수 있습니다.

CodeDeploy용 Amazon EC2 인스턴스 생성(CloudFormation 템플릿)

CloudFormation 템플릿을 사용하여 Amazon Linux 또는 Windows Server를 실행하는 Amazon EC2 인스턴스를 빠르게 시작할 수 있습니다. AWS CLI, CodeDeploy 콘솔 또는 AWS APIs를 사용하여 템플릿으로 인스턴스를 시작할 수 있습니다. 템플릿은 인스턴스를 시작하는 것 외에도 다음을 수행합니다.

- 인스턴스에 CodeDeploy 배포에 참여할 수 있는 권한을 부여 CloudFormation 하도록 지시합니다.
- 배포 중에 CodeDeploy가 인스턴스를 찾을 수 있도록 인스턴스에 태그를 지정합니다.
- 인스턴스에 CodeDeploy 에이전트를 설치하고 실행합니다.

Amazon EC2 인스턴스를 설정 CloudFormation 하기 위해를 사용할 필요는 없습니다. CodeDeploy용 인스턴스 작업에서 다른 방법을 확인해 보세요.

Ubuntu Server 또는 Red Hat Enterprise Linux(RHEL)를 실행하는 Amazon EC2 인스턴스에는 CloudFormation 템플릿을 제공하지 않습니다.

주제

- 시작하기 전 준비 사항
- CloudFormation 템플릿을 사용하여 Amazon EC2 인스턴스 시작(콘솔)
- CloudFormation 템플릿을 사용하여 Amazon EC2 인스턴스 시작(AWS CLI)

시작하기 전 준비 사항

CloudFormation 템플릿을 사용하여 Amazon EC2 인스턴스를 시작하려면 먼저 다음 단계를 완료해야 합니다.

1. 1단계: 설정에 설명된 대로 관리 사용자를 생성했는지 확인합니다. 사용자에게 다음과 같은 최소 권한이 있는지 다시 확인하고 누락된 권한을 추가합니다.

   - cloudformation:*

- codedeploy:*
- ec2:*
- iam:AddRoleToInstanceProfile
- iam:CreateInstanceProfile
- iam:CreateRole
- iam:DeleteInstanceProfile
- iam:DeleteRole
- iam:DeleteRolePolicy
- iam:GetRole
- iam:DeleteRolePolicy
- iam:PutRolePolicy
- iam:RemoveRoleFromInstanceProfile

2. Amazon Linux를 실행하는 Amazon EC2 인스턴스에 대한 SSH 액세스 또는 Windows 서버를 실행하는 인스턴스에 대한 RDP 액세스를 활성화하려면 인스턴스 키 페어가 있는지 확인합니다.

   키 페어 이름을 확인하려면 https://console.aws.amazon.com/ec2에서 Amazon EC2 콘솔을 엽니다. 탐색 창의 네트워크 및 보안에서 키 페어를 선택하고 목록에 표시되는 키 페어의 이름을 기록해 둡니다.

   새 키 페어를 생성하려면 Amazon EC2를 사용하여 키 페어 생성을 참조하세요. 키 페어가의 리전 및 엔드포인트에 나열된 리전 중 하나에 생성되었는지 확인합니다AWS 일반 참조. 그렇지 않으면 CodeDeploy에서 인스턴스 키 페어를 사용할 수 없습니다.

CloudFormation 템플릿을 사용하여 Amazon EC2 인스턴스 시작(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/cloudformation CloudFormation 콘솔을 엽니다.

   > **Important**
   >
   > 에서 사용한 AWS Management Console 것과 동일한 계정으로에 로그인합니다CodeDeploy 시작하기. 탐색 모음의 리전 선택기에서 리전에 나열된 리전 중 하나와의 엔드포인트를 선택합니다AWS 일반 참조. CodeDeploy는 이러한 리전만 지원합니다.

2. 스택 생성을 선택합니다.

3. 템플릿 선택(Choose a template)에서 Amazon S3 템플릿 URL 지정(Specify an Amazon S3 template URL)을 선택합니다. 상자에 해당 리전의 CloudFormation 템플릿 위치를 입력한 후 다음을 선택합니다.

리전	CloudFormation 템플릿 위치
미국 동부(오하이오) 리전	`http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/templates/latest/CodeDeploy_SampleCF_Template.json`
미국 동부(버지니아 북부) 리전	`http://s3.amazonaws.com/aws-codedeploy-us-east-1/templates/latest/CodeDeploy_SampleCF_Template.json`
미국 서부(캘리포니아 북부) 리전	`http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/templates/latest/CodeDeploy_SampleCF_Template.json`
미국 서부(오리건) 리전	`http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/templates/latest/CodeDeploy_SampleCF_Template.json`
캐나다(중부) 리전	`http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/templates/latest/CodeDeploy_SampleCF_Template.json`
유럽(아일랜드) 리전	`http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/templates/latest/CodeDeploy_SampleCF_Template.json`

리전	CloudFormation 템플릿 위치
Europe (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/templates/latest/CodeDeploy_SampleCF_Template.json
Europe (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/templates/latest/CodeDeploy_SampleCF_Template.json
유럽(프랑크푸르트) 리전	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
이스라엘(텔아비브) 리전	http://s3-il-central-1.amazonaws.com/aws-codedeploy-il-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
Asia Pacific (Hong Kong) Region	http://s3-ap-east-1.amazonaws.com/aws-codedeploy-ap-east-1/templates/latest/CodeDeploy_SampleCF_Template.json
Asia Pacific (Tokyo) Region	http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/templates/latest/CodeDeploy_SampleCF_Template.json

리전	CloudFormation 템플릿 위치
아시아 태평양(서울) 리전	`http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/templates/latest/CodeDeploy_SampleCF_Template.json`
아시아 태평양(싱가포르) 리전	`http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/templates/latest/CodeDeploy_SampleCF_Template.json`
Asia Pacific (Sydney) Region	`http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json`
Asia Pacific (Melbourne) Region	`https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json`
아시아 태평양(뭄바이) 리전	`http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/templates/latest/CodeDeploy_SampleCF_Template.json`
남아메리카(상파울루) 리전	`aws-codedeploy-ap-northeast-1.s3.sa-east-1.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json`

4. 스택 이름(Stack name) 상자에 스택 이름(예: **`CodeDeployDemoStack`**)을 입력합니다.

5. 파라미터(Parameters)에 다음을 입력한 후 다음(Next)을 선택합니다.

   - InstanceCount에 시작하려는 인스턴스의 수를 입력합니다. (기본값 1을 그대로 두는 것을 권장합니다.)
   - InstanceType에 시작할 인스턴스 유형을 입력합니다(또는 기본값 t1.micro를 그대로 유지).
   - KeyPairName에 인스턴스 키 페어 이름을 입력합니다. 키 페어 파일 확장명은 제외하고 키 페어 이름만 입력합니다.
   - OperatingSystem 상자에 **`Windows`**를 입력하여 Windows 서버를 실행하는 인스턴스를 시작합니다(또는 기본값 Linux를 그대로 유지).
   - SSHLocation에서, SSH 또는 RDP를 사용하여 인스턴스에 연결하는 데 사용할 IP 주소 범위를 입력합니다(또는 기본값 0.0.0.0/0을 그대로 유지).

     > **Important**
     >
     > 기본값 **`0.0.0.0/0`**은 데모용으로만 제공됩니다. CodeDeploy에서는 Amazon EC2 인스턴스에 포트에 대한 무제한 액세스를 요구하지 않습니다. SSH(및 HTTP) 포트에 대한 액세스를 제한할 것을 모범 사례로 권장합니다. 자세한 내용은 Amazon EC2 인스턴스의 보안 유지를 위한 팁을 참조하세요.

   - TagKey에 CodeDeploy가 배포 중에 인스턴스를 식별하는 데 사용할 인스턴스 태그 키를 입력합니다(또는 기본값 이름(Name)을 그대로 둠).
   - TagValue에 CodeDeploy가 배포 중에 인스턴스를 식별하는 데 사용할 인스턴스 태그 값을 입력합니다(또는 기본값 CodeDeployDemo를 그대로 유지).

6. 옵션(Options) 페이지에서 옵션 상자를 비워 두고 다음(Next)을 선택합니다.

   > **Important**
   >
   > CloudFormation 태그는 CodeDeploy 태그와 다릅니다.는 태그를 CloudFormation 사용하여 인프라 관리를 간소화합니다. CodeDeploy에서는 태그를 사용하여 Amazon EC2 인스턴스를 식별합니다. 파라미터 지정(Specify Parameters) 페이지에서 CodeDeploy 태그를 지정했습니다.

7. 검토 페이지의 기능에서 가 IAM 리소스를 생성할 CloudFormation 수 있음을 승인합니다 상자를 선택한 다음 생성을 선택합니다.

CloudFormation 가 스택을 생성하고 Amazon EC2 인스턴스를 시작한 후 콘솔에서 CloudFormation CREATE_COMPLETE가 상태 열에 표시됩니다. 이 프로세스는 몇 분 정도 걸릴 수 있습니다.

Amazon EC2 인스턴스에서 CodeDeploy 에이전트가 실행 중인지 확인하려면 CodeDeploy 에이전트 작업 관리를 확인한 다음 CodeDeploy를 사용하여 애플리케이션 생성로 진행합니다.

CloudFormation 템플릿을 사용하여 Amazon EC2 인스턴스 시작(AWS CLI)

1. create-stack 명령을 호출할 때 CloudFormation 템플릿을 사용합니다. 이 스택으로 CodeDeploy 에이전트가 설치된 새 Amazon EC2 인스턴스를 시작합니다.

   Amazon Linux를 실행하는 Amazon EC2 인스턴스 시작하기:

   ```
   aws cloudformation create-stack \
     --stack-name CodeDeployDemoStack \
     --template-url templateURL \
     --parameters ParameterKey=InstanceCount,ParameterValue=1
    ParameterKey=InstanceType,ParameterValue=t1.micro \
       ParameterKey=KeyPairName,ParameterValue=keyName
    ParameterKey=OperatingSystem,ParameterValue=Linux \
       ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0
    ParameterKey=TagKey,ParameterValue=Name \
       ParameterKey=TagValue,ParameterValue=CodeDeployDemo \
     --capabilities CAPABILITY_IAM
   ```

   Windows Server를 실행하는 Amazon EC2 인스턴스 시작:

   ```
   aws cloudformation create-stack --stack-name CodeDeployDemoStack --template-
   url template-url --parameters ParameterKey=InstanceCount,ParameterValue=1
    ParameterKey=InstanceType,ParameterValue=t1.micro
    ParameterKey=KeyPairName,ParameterValue=keyName
    ParameterKey=OperatingSystem,ParameterValue=Windows
    ParameterKey=SSHLocation,ParameterValue=0.0.0.0/0
    ParameterKey=TagKey,ParameterValue=Name
    ParameterKey=TagValue,ParameterValue=CodeDeployDemo --capabilities CAPABILITY_IAM
   ```

   *keyName*은 인스턴스 키 페어의 이름입니다. 키 페어 파일 확장명은 제외하고 키 페어 이름만 입력합니다.

*template-url*은 해당 리전의 CloudFormation 템플릿 위치입니다.

리전	CloudFormation 템플릿 위치
미국 동부(오하이오) 리전	`http://s3-us-east-2.amazonaws.com/aws-codedeploy-us-east-2/templates/latest/CodeDeploy_SampleCF_Template.json`
미국 동부(버지니아 북부) 리전	`http://s3.amazonaws.com/aws-codedeploy-us-east-1/templates/latest/CodeDeploy_SampleCF_Template.json`
미국 서부(캘리포니아 북부) 리전	`http://s3-us-west-1.amazonaws.com/aws-codedeploy-us-west-1/templates/latest/CodeDeploy_SampleCF_Template.json`
미국 서부(오리건) 리전	`http://s3-us-west-2.amazonaws.com/aws-codedeploy-us-west-2/templates/latest/CodeDeploy_SampleCF_Template.json`
캐나다(중부) 리전	`http://s3-ca-central-1.amazonaws.com/aws-codedeploy-ca-central-1/templates/latest/CodeDeploy_SampleCF_Template.json`
유럽(아일랜드) 리전	`http://s3-eu-west-1.amazonaws.com/aws-codedeploy-eu-west-1/templates/latest/CodeDeploy_SampleCF_Template.json`

리전	CloudFormation 템플릿 위치
Europe (London) Region	http://s3-eu-west-2.amazonaws.com/aws-codedeploy-eu-west-2/templates/latest/CodeDeploy_SampleCF_Template.json
Europe (Paris) Region	http://s3-eu-west-3.amazonaws.com/aws-codedeploy-eu-west-3/templates/latest/CodeDeploy_SampleCF_Template.json
유럽(프랑크푸르트) 리전	http://s3-eu-central-1.amazonaws.com/aws-codedeploy-eu-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
이스라엘(텔아비브) 리전	http://s3-il-central-1.amazonaws.com/aws-codedeploy-il-central-1/templates/latest/CodeDeploy_SampleCF_Template.json
Asia Pacific (Hong Kong) Region	http://s3-ap-east-1.amazonaws.com/aws-codedeploy-ap-east-1/templates/latest/CodeDeploy_SampleCF_Template.json
Asia Pacific (Tokyo) Region	http://s3-ap-northeast-1.amazonaws.com/aws-codedeploy-ap-northeast-1/templates/latest/CodeDeploy_SampleCF_Template.json

리전	CloudFormation 템플릿 위치
아시아 태평양(서울) 리전	http://s3-ap-northeast-2.amazonaws.com/aws-codedeploy-ap-northeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
아시아 태평양(싱가포르) 리전	http://s3-ap-southeast-1.amazonaws.com/aws-codedeploy-ap-southeast-1/templates/latest/CodeDeploy_SampleCF_Template.json
Asia Pacific (Sydney) Region	http://s3-ap-southeast-2.amazonaws.com/aws-codedeploy-ap-southeast-2/templates/latest/CodeDeploy_SampleCF_Template.json
Asia Pacific (Melbourne) Region	https://aws-codedeploy-ap-southeast-4.s3.ap-southeast-4.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json
아시아 태평양(뭄바이) 리전	http://s3-ap-south-1.amazonaws.com/aws-codedeploy-ap-south-1/templates/latest/CodeDeploy_SampleCF_Template.json
남아메리카(상파울루) 리전	aws-codedeploy-ap-northeast-1.s3.sa-east-1.amazonaws.com/templates/latest/CodeDeploy_SampleCF_Template.json

이 명령은 지정된 Amazon S3 버킷의 CloudFormation 템플릿을 **CodeDeployDemoStack**사용하여 라는 CloudFormation 스택을 생성합니다. Amazon EC2 인스턴스는 t1.micro 인스턴스 유형을 따르지만 모든 유형을 사용할 수 있습니다. **CodeDeployDemo** 값으로 태그 지정되어 있지만 모든 값을 태그 지정할 수 있습니다. 키 페어가 적용된 지정된 인스턴스가 있습니다.

2. describe-stacks 명령을 호출하여 라는 CloudFormation 스택이 성공적으로 생성**CodeDeployDemoStack**되었는지 확인합니다.

```
aws cloudformation describe-stacks --stack-name CodeDeployDemoStack --query
 "Stacks[0].StackStatus" --output text
```

CREATE_COMPLETE 값이 반환될 때까지 진행하지 마세요.

Amazon EC2 인스턴스에서 CodeDeploy 에이전트가 실행 중인지 확인하려면 CodeDeploy 에이전트 작업 관리 섹션을 확인한 다음 CodeDeploy를 사용하여 애플리케이션 생성로 진행합니다.

CodeDeploy 작업을 위한 Amazon EC2 인스턴스 구성

이러한 지침은 CodeDeploy에 사용하기 위해 Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux(RHEL) 또는 Windows Server를 실행하는 Amazon EC2 인스턴스 구성 방법을 안내합니다.

> **Note**
>
> Amazon EC2 인스턴스가 없는 경우 CloudFormation 템플릿을 사용하여 Amazon Linux 또는 Windows Server를 실행하는 인스턴스를 시작할 수 있습니다. Ubuntu Server 또는 RHEL용 템플릿은 제공되지 않습니다.

1단계: IAM 인스턴스 프로파일이 Amazon EC2 인스턴스에 연결되어 있는지 확인

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 인스턴스에서 인스턴스를 선택합니다.
3. 목록에서 Amazon EC2 인스턴스를 찾아서 선택합니다.
4. 세부 정보 창의 설명(Description) 탭에서 IAM 역할(IAM role) 필드의 값을 기록해 둔 후 다음 섹션으로 넘어갑니다.

필드가 비어 있는 경우 IAM 인스턴스 프로파일을 인스턴스에 연결할 수 있습니다. 자세한 내용은 IAM 역할을 인스턴스에 연결을 참조하세요.

2단계: 연결된 IAM 인스턴스 프로파일에 올바른 액세스 권한이 있는지 확인

1. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 이전 섹션의 4단계에서 기록해 둔 IAM 역할 이름을 찾아 선택합니다.

> **Note**
>
> 의 지침에 따라 생성한 서비스 역할 대신 CloudFormation 템플릿에서 생성한 서비스 역할을 사용하려면 다음 사항에 2단계: CodeDeploy에 대한 서비스 역할 생성유의하세요. CloudFormation 템플릿의 일부 버전에서는 생성되어 Amazon EC2 인스턴스에 연결된 IAM 인스턴스 프로파일의 표시 이름이 IAM 콘솔의 표시 이름과 동일하지 않습니다. 예를 들어 IAM 인스턴스 프로파일의 표시 이름은 `CodeDeploySampleStack-expnyi6-InstanceRoleInstanceProfile-IK8J8A9123EX`인 반면 IAM 콘솔의 IAM 인스턴스 프로파일의 표시 이름은 `CodeDeploySampleStack-expnyi6-InstanceRole-C5P33V1L64EX`일 수 있습니다.
> IAM 콘솔에서 인스턴스 프로파일을 식별하기 위해서는 두 가지 표시 이름의 `CodeDeploySampleStack-expnyi6-InstanceRole` 접두사가 동일한 것을 확인합니다. 표시 이름이 다를 수 있는 이유에 대해서는 인스턴스 프로파일에서 자세한 내용을 확인하세요.

4. 신뢰 관계(Trust Relationships) 탭을 선택합니다. 신뢰할 수 있는 엔터티(Trusted Entities)에 The identity provider(s) ec2.amazonaws.com이라는 항목이 없으면 이 Amazon EC2 인스턴스를 사용할 수 없습니다. 중지한 후, CodeDeploy용 인스턴스 작업의 정보를 사용하여 Amazon EC2 인스턴스를 생성합니다

   The identity provider(s) ec2.amazonaws.com이라는 항목이 있고 GitHub 리포지토리에만 애플리케이션을 저장하는 경우는 3단계: Amazon EC2 인스턴스에 태그 지정으로 진행합니다.

   The identity provider(s) ec2.amazonaws.com이라는 항목이 있고 애플리케이션을 Amazon S3 버킷에 저장하는 경우 권한(Permissions) 탭을 선택합니다.
5. 권한 정책(Permissions policies) 영역에 정책이 있는 경우 정책을 확장한 다음 정책 편집(Edit policy)을 선택합니다.

6. JSON 탭을 선택합니다. Amazon S3 버킷에 애플리케이션을 저장하는 경우 "s3:Get*" 및 "s3:List*"가 지정된 작업 목록에 있어야 합니다.

   다음과 같을 것입니다.

```
{"Statement":[{"Resource":"*","Action":[
  ... Some actions may already be listed here ...
  "s3:Get*","s3:List*"
  ... Some more actions may already be listed here ...
  ],"Effect":"Allow"}]}
```

   또는 다음과 같을 것입니다.

   JSON

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

   "s3:Get*" 및 "s3:List*"가 지정된 작업 목록에 없으면 편집(Edit)을 선택하여 추가한 다음 저장(Save)을 선택합니다. ("s3:Get*" 또는 "s3:List*"가 목록의 마지막 작업이 아닌 경우 정책 문서의 유효성을 검사할 수 있도록 작업 뒤에 쉼표를 추가해야 합니다.)

> **Note**
>
> Amazon EC2 인스턴스에서 액세스해야 하는 Amazon S3 버킷으로만 이 정책 적용을 제한하는 것이 좋습니다. CodeDeploy 에이전트가 포함된 Amazon S3 버킷에 대한 액세스 권한을 제공해야 합니다. 그렇지 않으면 인스턴스에 CodeDeploy 에이전트를 설치 또는 업데이트할 때 오류가 발생할 수 있습니다. Amazon S3의 일부 CodeDeploy 리소스 키트

버킷에만 IAM 인스턴스 프로파일 액세스 권한을 부여하려면 다음 정책을 사용하되 액세스를 차단하려는 버킷 라인을 제거합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::aws-codedeploy-us-east-2/*",
        "arn:aws:s3:::aws-codedeploy-us-east-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
```

```
    ]
}
```

3단계: Amazon EC2 인스턴스에 태그 지정

배포 중에 CodeDeploy가 Amazon EC2 인스턴스를 찾을 수 있도록 태그를 지정하는 방법은 콘솔에서 태그 작업하기를 참조한 다음 이 페이지로 돌아오세요.

> **Note**
>
> Amazon EC2 인스턴스에 원하는 키와 값으로 태깅할 수 있습니다. 배포할 때 해당 키와 값을 지정했는지 확인합니다.

4단계: Amazon EC2 인스턴스에 AWS CodeDeploy 에이전트 설치

Amazon EC2 인스턴스에서 CodeDeploy 에이전트를 설치하고 에이전트가 실행 중인지 확인하는 지침을 CodeDeploy 에이전트 작업 관리 섹션에서 확인한 다음 CodeDeploy를 사용하여 애플리케이션 생성으로 진행합니다.

CodeDeploy용 온프레미스 인스턴스 작업

온프레미스 인스턴스는 CodeDeploy 에이전트를 실행하고 퍼블릭 AWS 서비스 엔드포인트에 연결할 수 있는 Amazon EC2 인스턴스가 아닌 모든 물리적 디바이스입니다.

온프레미스 인스턴스에 CodeDeploy 애플리케이션 개정 버전을 배포하는 데는 두 가지 주요 단계가 필요합니다.

- 1단계 – 각 온프레미스 인스턴스를 구성하고 CodeDeploy에 등록한 다음 태그를 지정합니다.
- 2단계 - 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포합니다.

> **Note**
>
> 올바르게 구성되고 등록된 온프레미스 인스턴스에 샘플 애플리케이션 개정 버전을 만들고 배포하는 방법을 실험하려면 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux) 단원을 참

조하세요. 온프레미스 인스턴스와 이러한 인스턴스가 CodeDeploy와 작동하는 방식에 대한 자세한 내용은 Working with On-Premises Instances 단원을 참조하세요.

온프레미스 인스턴스를 배포에서 더 이상 사용하지 않으려면 배포 그룹에서 온프레미스 인스턴스 태그를 제거할 수 있습니다. 보다 강력한 접근 방식을 위해 인스턴스에서 온프레미스 인스턴스 태그를 제거합니다. 또한 온프레미스 인스턴스를 명시적으로 등록 취소할 수 있으므로 더 이상 배포에서 사용할 수 없게 됩니다. 자세한 내용은 CodeDeploy에서 온프레미스 인스턴스 작업 관리 단원을 참조하십시오.

이 섹션의 지침은 온프레미스 인스턴스를 구성한 다음 배포에 사용할 수 있도록 CodeDeploy에 등록하고 태그를 지정하는 방법을 보여 줍니다. 이 섹션에서는 CodeDeploy를 사용하여 온프레미스 인스턴스에 대한 정보를 가져오고 더 이상 배포할 계획이 없는 온프레미스 인스턴스를 등록 취소하는 방법도 설명합니다.

주제

- 온프레미스 인스턴스 구성을 위한 사전 요구 사항
- CodeDeploy에 온프레미스 인스턴스 등록
- CodeDeploy에서 온프레미스 인스턴스 작업 관리

온프레미스 인스턴스 구성을 위한 사전 요구 사항

온프레미스 인스턴스를 등록하려면 다음 사전 요구 사항을 충족해야 합니다.

> **Important**
>
> register-on-premises-instance 명령을 사용하고 AWS Security Token Service (AWS STS)로 생성된 임시 자격 증명을 주기적으로 새로 고치는 경우 다른 사전 조건이 있습니다. 자세한 내용은 IAM 세션 ARN 등록 필수 조건 단원을 참조하세요.

디바이스 요구 사항

준비하고, CodeDeploy에 등록하고, 온프레미스 인스턴스로 태그를 지정하려는 디바이스는 지원되는 운영 체제에서 실행 중이어야 합니다. 목록을 보려면 CodeDeploy 에이전트에서 지원하는 운영 체제 단원을 참조하세요.

운영 체제가 지원되지 않는 경우 필요에 따라 조정할 수 있도록 CodeDeploy 에이전트는 오픈 소스로 제공됩니다. 자세한 내용은 GitHub에서 CodeDeploy 에이전트 리포지토리를 참조하세요.

아웃바운드 통신

온프레미스 인스턴스는 CodeDeploy와 통신하기 위해 퍼블릭 AWS 서비스 엔드포인트에 연결할 수 있어야 합니다.

CodeDeploy 에이전트는 포트 443을 통해 HTTPS를 사용하여 아웃바운드 통신을 수행합니다.

관리 제어

온프레미스 인스턴스를 구성하기 위해 온프레미스 인스턴스에 사용하는 로컬 또는 네트워크 계정은 `sudo` 또는 `root`(Ubuntu Server의 경우) 또는 관리자(Windows Server의 경우)로 실행해야 합니다.

IAM 권한

온프레미스 인스턴스를 등록하는 데 사용하는 IAM 자격 증명에는 등록을 완료할 수 있는 권한(및 필요에 따라 온프레미스 인스턴스를 등록 취소할 수 있는 권한)이 있어야 합니다.

3단계: CodeDeploy 사용자 권한 제한에서 설명하는 정책 이외에도 IAM 자격 증명 호출에 다음과 같은 추가 정책을 연결해야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

IAM 정책을 연결하는 방법에 대한 자세한 내용은 IAM 정책 관리를 참조하세요.

CodeDeploy에 온프레미스 인스턴스 등록

온프레미스 인스턴스를 등록하려면 IAM 자격 증명을 사용하여 요청을 인증해야 합니다. IAM 자격 증명 및 등록 방법에 대해 다음 옵션 중에서 선택할 수 있습니다.

- IAM 역할 ARN을 사용하여 요청을 인증합니다.
  - register-on-premises-instance 명령을 사용하고 AWS Security Token Service (AWS STS)로 생성된 임시 자격 증명을 주기적으로 새로 고쳐 대부분의 등록 옵션을 수동으로 구성합니다. 이 옵션은 제한 시간이 있고 주기적으로 새로 고쳐야 하는 임시 토큰을 사용하여 인증이 수행되므로 가장 높은 수준의 보안을 제공합니다. 이 옵션은 규모와 상관없이 프로덕션 배포에 권장됩니다. 자세한 내용은 register-on-premises-instance 명령(IAM 세션 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다. 단원을 참조하세요.
- (권장되지 않음) IAM 사용자 ARN을 사용하여 요청을 인증합니다.
  - 대부분의 자동화된 등록 프로세스에 register 명령을 사용합니다. 이 옵션은 보안의 중요성이 비교적 낮은 비프로덕션 배포에만 사용해야 합니다. 이 옵션은 인증에 정적(영구) 보안 인증 정보를 사용하므로 덜 안전합니다. 이 옵션은 단일 온프레미스 인스턴스 등록에 가장 적합합니다. 자세한 내용은 레지스터 명령(IAM 사용자 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다. 단원을 참조하세요.
  - register-on-premises-instance 명령을 사용하여 대부분의 등록 옵션을 수동으로 구성합니다. 소수의 온프레미스 인스턴스를 등록하는 데 적합합니다. 자세한 내용은 register-on-premises-instance 명령(IAM 사용자 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다. 섹션을 참조하세요.

주제

- register-on-premises-instance 명령(IAM 세션 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다.
- 레지스터 명령(IAM 사용자 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다.
- register-on-premises-instance 명령(IAM 사용자 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다.

register-on-premises-instance 명령(IAM 세션 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다.

온프레미스 인스턴스의 인증 및 등록을 최대한 제어하려면 register-on-premises-instance 명령을 사용하고 AWS Security Token Service ()로 생성된 임시 자격 증명을 주기적으로 새로 고칠 수 있습니다AWS STS. 인스턴스에 대한 정적 IAM 역할은 새로 고쳐진 자격 AWS STS 증명의 역할을 수임하여 CodeDeploy 배포 작업을 수행합니다.

이 방법은 많은 수의 인스턴스를 등록해야 하는 경우에 가장 유용합니다. CodeDeploy를 사용하여 등록 프로세스를 자동화할 수 있습니다. 자체 자격 증명 및 인증 시스템을 사용하여 온프레미스 인스턴스를 인증하고 서비스에서 인스턴스에 IAM 세션 자격 증명을 배포하여 CodeDeploy와 함께 사용할 수 있습니다.

> **Note**
>
> 또는 모든 온프레미스 인스턴스에 분산된 공유 IAM 사용자를 사용하여 AWS STS AssumeRole API를 호출하여 온프레미스 인스턴스에 대한 세션 자격 증명을 검색할 수 있습니다. 이 방법은 안전성이 떨어지며 프로덕션 환경이나 미션 크리티컬 환경에는 권장되지 않습니다.

다음 주제의 정보를 사용하여에서 생성된 임시 보안 자격 증명을 사용하여 온프레미스 인스턴스를 구성합니다 AWS STS.

주제

- IAM 세션 ARN 등록 필수 조건
- 1단계: 온프레미스 인스턴스에서 가정할 IAM 역할 만들기
- 2단계:를 사용하여 개별 인스턴스에 대한 임시 자격 증명 생성 AWS STS
- 3단계: 온프레미스 인스턴스에 구성 파일 추가
- 4단계: CodeDeploy 배포를 위한 온프레미스 인스턴스 준비
- 5단계: CodeDeploy에 온프레미스 인스턴스 등록
- 6단계: 온프레미스 인스턴스에 태그 지정
- 7단계 : 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포합니다.
- 8단계: 온프레미스 인스턴스에 대한 배포 추적

IAM 세션 ARN 등록 필수 조건

온프레미스 인스턴스 구성을 위한 사전 요구 사항에 나열된 사전 필수 조건에 더하여 다음 추가 요구 사항을 충족해야 합니다.

IAM 권한

온프레미스 인스턴스를 등록하는 데 사용하는 IAM 자격 증명에는 CodeDeploy 작업을 수행할 수 있는 권한이 부여되어야 합니다. AWSCodeDeployFullAccess 관리형 정책이 IAM 자격 증명에 연결되도록 해야 합니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

임시 보안 인증 정보를 새로 고치는 시스템

IAM 세션 ARN을 사용하여 온프레미스 인스턴스를 등록하는 경우 임시 자격 증명을 주기적으로 새로 고칠 수 있는 시스템이 있어야 합니다. 자격 증명이 생성될 때 더 짧은 기간을 지정하면 임시 자격 증명이 1시간 후에 만료됩니다. 자격 증명 새로 고치는 방법은 두 가지입니다.

- 방법 1: ID 및 인증 시스템을 주기적으로 폴링하고 최신 세션 자격 증명을 인스턴스에 복사하는 CRON 스크립트와 함께 회사 네트워크에 있는 자격 증명 및 인증 시스템을 사용합니다. 이를 통해 조직에서 사용하는 인증 유형을 지원하기 위해 CodeDeploy 에이전트 또는 서비스를 변경할 필요 AWS 없이 인증 및 자격 증명 구조를와 통합할 수 있습니다.
- 방법 2: 인스턴스에서 CRON 작업을 주기적으로 실행하여 AWS STS AssumeRole 작업을 호출하고 CodeDeploy 에이전트가 액세스할 수 있는 파일에 세션 자격 증명을 씁니다. 이 방법을 사용하려면 IAM 사용자를 사용하고 온프레미스 인스턴스에 자격 증명을 복사해야 하지만 온프레미스 인스턴스 전체에서 동일한 IAM 사용자 및 자격 증명을 재사용할 수 있습니다.

자격 AWS STS 증명 생성 및 작업에 대한 자세한 내용은 AWS Security Token Service API 참조 및 임시 보안 자격 증명을 사용하여 AWS 리소스에 대한 액세스 요청을 참조하세요.

1단계: 온프레미스 인스턴스에서 가정할 IAM 역할 만들기

AWS CLI 또는 IAM 콘솔을 사용하여 온프레미스 인스턴스가 CodeDeploy를 인증하고 상호 작용하는 데 사용할 IAM 역할을 생성할 수 있습니다.

IAM 역할은 하나씩만 생성하면 됩니다. 각 온프레미스 인스턴스는 이 역할을 수임하여 이 역할에 부여된 권한을 제공하는 임시 보안 자격 증명을 검색할 수 있습니다.

만드는 역할에 CodeDeploy 에이전트를 설치하는 데 필요한 파일에 액세스하려면 다음 권한이 필요합니다.

JSON

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

온프레미스 인스턴스에서 액세스해야 하는 Amazon S3 버킷으로만 이 정책을 제한하는 것이 좋습니다. 이 정책을 제한하는 경우, CodeDeploy 에이전트가 포함된 Amazon S3 버킷에 대한 액세스 권한도 부여해야 합니다. 그렇지 않으면 온프레미스 인스턴스에 CodeDeploy 에이전트를 설치 또는 업데이트할 때 오류가 발생할 수 있습니다. Amazon S3 버킷 액세스 제어에 대한 자세한 내용은 Amazon S3 리소스에 대한 액세스 권한 관리를 참조하세요.

IAM 역할을 만들려면

1. `--role-name` 옵션을 사용하여 create-role 명령을 호출해 IAM 역할의 이름(예: `CodeDeployInstanceRole`)을 지정하고 `--assume-role-policy-document` 옵션을 사용하여 권한을 제공할 수 있습니다.

   이 인스턴스에 대한 IAM 역할을 생성할 때 `CodeDeployInstanceRole`(이)라는 이름을 지정하고 `CodeDeployRolePolicy.json`(이)라는 파일에 필요한 권한을 포함합니다.

   ```
   aws iam create-role --role-name CodeDeployInstanceRole --assume-role-policy-
   document file://CodeDeployRolePolicy.json
   ```

2. create-role 명령에 대한 호출의 출력에서 ARN 필드의 값을 기록해 둡니다. 예제:

   ```
   arn:aws:iam::123456789012:role/CodeDeployInstanceRole
   ```

AWS STS AssumeRole API를 사용하여 각 인스턴스에 대한 단기 자격 증명을 생성할 때 역할 ARN이 필요합니다.

IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 권한을 위임할 역할 생성을 참조하세요.

기존 역할에 사용 권한을 할당하는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 put-role-policy을 참조하세요.

2단계:를 사용하여 개별 인스턴스에 대한 임시 자격 증명 생성 AWS STS

온프레미스 인스턴스 등록에 사용할 임시 자격 증명을 생성하기 전에 임시 자격 증명을 생성할 IAM 자격 증명(사용자 또는 역할)을 생성하거나 선택해야 합니다. `sts:AssumeRole` 권한은 이 IAM 자격 증명에 대한 정책 설정에 포함되어야 합니다.

IAM 자격 증명에 `sts:AssumeRole` 권한을 부여하는 방법에 대한 자세한 내용은 AWS 서비스에 권한을 위임할 역할 생성 및 AssumeRole을 참조하세요.

임시 자격 증명 생성 방법은 두 가지입니다.

- AWS CLI와(과) 함께 assume-role 명령 사용 예제:

```
aws sts assume-role --role-arn arn:aws:iam::12345ACCOUNT:role/role-arn --role-
session-name session-name
```

위치:

- *12345ACCOUNT*는 조직의 12자리 계정 번호입니다.
- *role-arn*은 1단계: 온프레미스 인스턴스에서 가정할 IAM 역할 만들기에서 생성한 수임할 역할의 ARN입니다.
- *session-name*은 지금 만들려는 역할 세션에 제공할 이름입니다.

> **Note**
>
> 자격 증명 및 인증 시스템을 주기적으로 폴링하고 최신 세션 자격 증명을 인스턴스에 복사하는 CRON 스크립트를 사용하는 경우(에 설명된 임시 자격 증명을 새로 고치는 방법 1IAM 세션 ARN 등록 필수 조건), 지원되는 SDK를 사용하여 AssumeRole을 호출 AWS 할 수 있습니다.

- 에서 제공하는 도구를 사용합니다 AWS.

  aws-codedeploy-session-helper 도구는 AWS STS 자격 증명을 생성하고 인스턴스에 배치한 파일에 기록합니다. 이 도구는 IAM 세션 ARN 등록 필수 조건에 설명된 임시 자격 증명을 새로 고치는 방법 2에 가장 적합합니다. 이 방법에서는 aws-codedeploy-session-helper 도구가 각 인스턴스에 배치되고 IAM 사용자의 권한을 사용하여 명령을 실행합니다. 각 인스턴스는이 도구와 함께 동일한 IAM 사용자의 자격 증명을 사용합니다.

  자세한 내용은 aws-codedeploy-session-helper GitHub 리포지토리를 참조하세요.

  > **Note**
  >
  > IAM 세션 자격 증명을 생성한 후에는 온프레미스 인스턴스의 어느 위치에나 배치합니다. 다음 단계에서는 이 위치의 자격 증명에 액세스하도록 CodeDeploy 에이전트를 구성합니다.

계속하기 전에 임시 자격 증명을 주기적으로 새로 고치는 데 사용할 시스템이 있는지 확인하세요. 임시 자격 증명이 새로 고쳐지지 않으면 온프레미스 인스턴스에 대한 배포가 실패합니다. 자세한 내용은 IAM 세션 ARN 등록 필수 조건의 “임시 자격 증명을 새로 고치는 시스템”을 참조하세요.

3단계: 온프레미스 인스턴스에 구성 파일 추가

루트 또는 관리자 권한을 사용하여 온프레미스 인스턴스에 구성 파일을 추가합니다. 이 구성 파일은 CodeDeploy에 사용할 IAM 자격 증명과 대상 AWS 리전을 선언하는 데 사용됩니다. 온프레미스 인스턴스의 특정 위치에 파일을 추가해야 합니다. 파일에는 IAM 임시 세션 ARN, 보안 키 ID 및 보안 액세스 키, 대상 AWS 리전이 포함되어야 합니다.

구성 파일을 추가하려면

1. 온프레미스 인스턴스의 다음 위치에 `codedeploy.onpremises.yml`(Ubuntu Server 또는 RHEL 온프레미스 인스턴스의 경우) 또는 `conf.onpremises.yml`(Windows Server 온-레미스 인스턴스의 경우)라는 이름의 파일을 만듭니다.

   - Ubuntu Server: `/etc/codedeploy-agent/conf`
   - Windows Server: `C:\ProgramData\Amazon\CodeDeploy`

2. 텍스트 편집기를 사용하여 새로 만든 `codedeploy.onpremises.yml` 파일(Linux) 또는 `conf.onpremises.yml` 파일(Windows)에 다음 정보를 추가하세요.

   ```
   ---
   ```

```
iam_session_arn: iam-session-arn
aws_credentials_file: credentials-file
region: supported-region
```

위치:

- *iam-session-arn*은 2단계:를 사용하여 개별 인스턴스에 대한 임시 자격 증명 생성 AWS STS에서 기록해 둔 IAM 세션 ARN 입니다.
- *credentials-file*은 2단계:를 사용하여 개별 인스턴스에 대한 임시 자격 증명 생성 AWS STS에서 기록해 둔대로 임시 세션 ARN에 대한 자격 증명 파일의 위치입니다.
- *supported-region*은의 리전 및 엔드포인트에 나열된 대로 CodeDeploy가 지원하는 리전 중 하나입니다AWS 일반 참조. https://docs.aws.amazon.com/general/latest/gr/rande.html#codedeploy_region

4단계: CodeDeploy 배포를 위한 온프레미스 인스턴스 준비

설치 및 구성 AWS CLI

온프레미스 인스턴스 AWS CLI 에를 설치하고 구성합니다. (AWS CLI 는 온프레미스 인스턴스에서 CodeDeploy 에이전트를 다운로드하고 설치하는 데 사용됩니다.)

1. 온프레미스 인스턴스 AWS CLI 에를 설치하려면 AWS Command Line Interface 사용 설명서의 로 설정 AWS CLI의 지침을 따르세요.

   > **Note**
   >
   > 온프레미스 인스턴스 작업을 위한 CodeDeploy 명령은 AWS CLI의 버전 1.7.19에서 사용할 수 있게 되었습니다. 버전이 AWS CLI 이미 설치되어 있는 경우를 호출하여 버전을 확인할 수 있습니다aws --version.

2. 온프레미스 인스턴스 AWS CLI 에서를 구성하려면 AWS Command Line Interface 사용 설명서의 구성 AWS CLI의 지침을 따르세요.

   > **Important**
   >
   > 를 구성할 때 AWS CLI (예: aws configure 명령을 호출하여) 최소한에 설명된 권한이 있는 IAM 사용자의 보안 키 ID와 보안 액세스 키를 지정해야 합니다IAM 세션 ARN 등록 필수 조건.

AWS_REGION 환경 변수 설정(Ubuntu Server 및 RHEL만 해당)

온프레미스 인스턴스에서 Ubuntu Server 또는 RHEL을 실행하지 않는 경우 이 단계를 건너뛰고 "CodeDeploy 에이전트 생성"으로 직접 이동합니다.

Ubuntu Server 또는 RHEL 온프레미스 인스턴스에 CodeDeploy 에이전트를 설치하고 새 버전을 사용할 수 있을 때마다 인스턴스가 CodeDeploy 에이전트를 업데이트하도록 설정합니다. 인스턴스의 `AWS_REGION` 환경 변수를 CodeDeploy에서 지원하는 리전 중 하나의 식별자로 설정하면 됩니다. CodeDeploy 애플리케이션, 배포 그룹 및 애플리케이션 개정이 위치한 리전(예: `us-west-2`)으로 값을 설정하는 것이 좋습니다. 리전 목록은의 리전 및 엔드포인트를 참조하세요AWS 일반 참조.

환경 변수를 설정하려면 터미널에서 다음을 호출하세요.

```
export AWS_REGION=supported-region
```

여기서 *`supported-region`*은 리전 식별자입니다(예: `us-west-2`).

CodeDeploy 에이전트 설치

- Ubuntu Server 온프레미스 인스턴스의 경우 Ubuntu Server용 CodeDeploy 에이전트 설치의 지침을 따르고 이 페이지로 돌아옵니다.
- RHEL 온프레미스 인스턴스의 경우 Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치의 지침을 따르고 이 페이지로 돌아옵니다.
- Windows Serve 온프레미스 인스턴스의 경우 Windows Server용 CodeDeploy 에이전트 설치의 지침을 따르고 이 페이지로 돌아옵니다.

5단계: CodeDeploy에 온프레미스 인스턴스 등록

이 단계의 지침은 온프레미스 인스턴스 자체에서 온프레미스 인스턴스를 등록한다고 가정합니다. AWS CLI 설치 및 구성된 별도의 디바이스 또는 인스턴스에서 온프레미스 인스턴스를 등록할 수 있습니다.

AWS CLI 를 사용하여 배포에 사용할 수 있도록 CodeDeploy에 온프레미스 인스턴스를 등록합니다.

를 사용하려면 먼저에서 생성한 임시 세션 자격 증명의 ARN AWS CLI이 필요합니다3단계: 온프레미스 인스턴스에 구성 파일 추가. 예를 들어, `AssetTag12010298EX`(으)로 식별하는 인스턴스의 경우:

```
arn:sts:iam::123456789012:assumed-role/CodeDeployInstanceRole/AssetTag12010298EX
```

register-on-premises-instance 명령을 호출하여 다음을 지정합니다.

- 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-name` 옵션 사용).

> **Important**
>
> 특히 디버깅을 위해 온프레미스 인스턴스를 식별하려면 온프레미스 인스턴스의 일부 고유 특성(예: STS 자격 증명의 session-name과 일련 번호 또는 해당되는 경우 내부 자산 식별자)에 매핑되는 이름을 지정하는 것이 좋습니다. 이름으로 MAC 주소를 지정하는 경우 MAC 주소에는 콜론(:)과 같이 CodeDeploy에서 허용하지 않는 문자가 포함되어 있다는 점에 유의하세요. 허용되는 문자 목록은 CodeDeploy 할당량 단원을 참조하세요.

- 1단계: 온프레미스 인스턴스에서 가정할 IAM 역할 만들기에서 여러 온프레미스 인스턴스를 인증하도록 설정한 IAM 세션 ARN

예:

```
aws deploy register-on-premises-instance --instance-name name-of-instance --iam-
session-arn arn:aws:sts::account-id:assumed-role/role-to-assume/session-name
```

위치:

- *name-of-instance*은 온프레미스 인스턴스를 식별하는 데 사용하는 이름입니다(예: `AssetTag12010298EX`).
- *account-id*는 조직의 12자리 계정 ID입니다(예: 111222333444).
- *role-to-assume*은 인스턴스에 대해 생성한 IAM 역할의 이름입니다(예: `CodeDeployInstanceRole`).
- *session-name*은 2단계:를 사용하여 개별 인스턴스에 대한 임시 자격 증명 생성 AWS STS에서 지정한 세션 역할의 이름입니다..

6단계: 온프레미스 인스턴스에 태그 지정

AWS CLI 또는 CodeDeploy 콘솔을 사용하여 온프레미스 인스턴스에 태그를 지정할 수 있습니다. CodeDeploy는 온프레미스 인스턴스 태그를 사용하여 배포 중 배포 대상을 식별합니다.

온프레미스 인스턴스에 태그 지정(CLI)

- add-tags-to-on-premises-instances 명령을 호출하여 다음을 지정합니다.
  - 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-names` 옵션 사용).

- 사용하려는 온프레미스 인스턴스 태그 키 및 태그 값의 이름(`--tags` 옵션 사용). 이름과 값을 둘 다 지정해야 합니다. CodeDeploy는 값만 있는 온프레미스 인스턴스 태그를 허용하지 않습니다.

  예:

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

온프레미스 인스턴스에 태그 지정(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 온프레미스 인스턴스를 선택합니다.
3. 온프레미스 인스턴스 목록에서 태그를 지정할 온프레미스 인스턴스의 이름을 선택합니다.
4. 태그 목록에서 원하는 태그 키와 태그 값을 선택하거나 입력합니다. 태그 키와 태그 값을 입력하면 다른 행이 나타납니다. 최대 10개의 태그에 대해 이 작업을 반복할 수 있습니다. 태그를 제거하려면 제거를 선택합니다.
5. 태그를 추가한 후 태그 업데이트를 선택합니다.

7단계 : 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포합니다.

등록 및 태그가 지정된 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포할 준비가 되었습니다.

애플리케이션 개정 버전을 Amazon EC2 인스턴스에 배포하는 것과 유사한 방식으로 온프레미스 인스턴스에 배포합니다. 지침은 CodeDeploy에서 배포 만들기 단원을 참조하세요. 이러한 지침은 애플리케이션 만들기, 배포 그룹 만들기 및 애플리케이션 개정 버전 준비를 비롯한 필수 구성 요소에 대한 링크가 포함되어 있습니다. 배포하기 위해 간단한 샘플 애플리케이션 개정 버전이 필요한 경우 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)의 2단계: 샘플 애플리케이션 수정 버전 만들기에 설명된 개정 버전을 생성할 수 있습니다.

> **Important**
>
> 온프레미스 인스턴스를 대상으로 하는 배포 그룹을 만드는 과정에서 CodeDeploy 서비스 역할을 다시 사용하는 경우 서비스 역할 정책 설명의 `Action` 부분에 `Tag:get*`을(를) 포함해야 합니다. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.

8단계: 온프레미스 인스턴스에 대한 배포 추적

등록되고 태그가 지정된 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포한 후 배포의 진행 상황을 추적할 수 있습니다.

Amazon EC2 인스턴스에 대한 배포를 추적하는 것과 유사한 방식으로 온프레미스 인스턴스에 대한 배포를 추적합니다. 지침은 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

레지스터 명령(IAM 사용자 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다.

> **Important**
>
> IAM 사용자를 사용하여 인스턴스를 등록하는 것은 인증에 정적(영구) 보안 인증 정보를 사용하므로 권장되지 않습니다. 보안을 강화하려면 인증을 위한 임시 보안 인증 정보를 사용하여 인스턴스를 등록하는 것이 좋습니다. 자세한 내용은 register-on-premises-instance 명령(IAM 세션 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다. 단원을 참조하십시오.

> **Important**
>
> IAM 사용자의 액세스 키(영구 보안 인증 정보)를 교체할 계획이 있어야 합니다. 자세한 내용은 액세스 키 교체를 참조하세요.

이 섹션에서는 온프레미스 인스턴스를 구성하고 최소한의 작업으로 CodeDeploy에 등록하고 태그를 지정하는 방법에 대해 설명합니다. register 명령은 단일 또는 소규모 온프레미스 인스턴스 집합으로 작업할 때 가장 유용합니다. register 명령은 IAM 사용자 ARN을 사용하여 인스턴스를 인증하는 경우에만 사용할 수 있습니다. register 명령은 인증용 IAM 세션 ARN과 함께 사용합니다.

register 명령을 사용할 때 CodeDeploy에서 다음 작업을 수행할 수 있습니다.

- 명령으로 IAM 사용자를 지정하지 않은 경우 온프레미스 인스턴스에 AWS Identity and Access Management 대해에서 IAM 사용자를 생성합니다.
- IAM 사용자의 자격 증명을 온프레미스 인스턴스 구성 파일에 저장합니다.
- CodeDeploy에 온프레미스 인스턴스 등록
- 명령의 일부로 태그를 지정한 경우 온프레미스 인스턴스에 태그를 추가합니다.

> **Note**
>
> register-on-premises-instance 명령은 register 명령 대신 사용할 수 있습니다. register-on-premises-instance 명령을 사용하여 온프레미스 인스턴스를 구성하고 주로 직접 CodeDeploy에 등록하고 태그를 지정할 수 있습니다. register-on-premises-instance 명령은 IAM 사용자 ARN 대신 IAM 세션 ARN을 사용하여 인스턴스를 등록할 수 있는 옵션도 제공합니다. 이 접근 방식은 대규모 온프레미스 인스턴스 플릿이 있는 경우 큰 이점을 제공합니다. 특히 각 온프레미스 인스턴스에 대해 하나씩 IAM 사용자를 만들지 않고 단일 IAM 세션 ARN 사용하여 여러 인스턴스를 인증할 수 있습니다. 자세한 내용은 register-on-premises-instance 명령(IAM 사용자 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다. 및 register-on-premises-instance 명령(IAM 세션 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다. 섹션을 참조하세요.

주제

- 1단계: 온프레미스 인스턴스 AWS CLI 에 설치 및 구성
- 2단계: 등록 명령 호출
- 3단계: 설치 명령 호출
- 4단계 : 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포합니다.
- 5단계: 온프레미스 인스턴스에 대한 배포 추적

1단계: 온프레미스 인스턴스 AWS CLI 에 설치 및 구성

1. 온프레미스 인스턴스 AWS CLI 에를 설치합니다. AWS Command Line Interface 사용 설명서의 AWS CLI을(를) 이용하여 설치 시작하기에 나와 있는 지침을 따릅니다.

> **Note**
>
> 온프레미스 인스턴스 작업을 위한 CodeDeploy 명령은 AWS CLI 버전 1.7.19 이상에서 사용할 수 있습니다. 가 AWS CLI 이미 설치되어 있는 경우를 호출aws --version하여 버전을 확인합니다.

2. 온프레미스 인스턴스 AWS CLI 에서를 구성합니다. AWS Command Line Interface 사용 설명서의 AWS CLI구성에 나와 있는 지침을 따릅니다.

> **Important**
>
> 를 구성할 때 AWS CLI (예: aws configure 명령을 호출하여) 최소한에 지정된 권한 외에도 다음 액세스 권한이 있는 IAM 사용자의 보안 키 ID와 보안 AWS 액세스 키를 지정해야 합니다온프레미스 인스턴스 구성을 위한 사전 요구 사항. 이렇게 하면 온프레미스 인스턴스에 CodeDeploy 에이전트를 다운로드하고 설치할 수 있습니다. 액세스 권한은 다음과 비슷할 수 있습니다.
>
> JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*",
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
        "iam:GetUser",
        "tag:getTagKeys",
        "tag:getTagValues",
        "tag:GetResources"
      ],
```

```
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

> **Note**
>
> 이전에 표시된 Amazon S3 버킷 중 하나에 액세스하려고 할 때 액세스 거부 오류가 표시되면 버킷의 리소스 ARN에서 `/*` 부분(예: `arn:aws:s3:::aws-codedeploy-sa-east-1`)을 생략해 보세요.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*",
        "iam:CreateAccessKey",
        "iam:CreateUser",
        "iam:DeleteAccessKey",
        "iam:DeleteUser",
        "iam:DeleteUserPolicy",
        "iam:ListAccessKeys",
        "iam:ListUserPolicies",
        "iam:PutUserPolicy",
```

```
        "iam:GetUser",
        "tag:GetResources"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket2/*"
      ]
    }
  ]
}
```

2단계: 등록 명령 호출

이 단계에서는 온프레미스 인스턴스 자체에서 온프레미스 인스턴스를 등록한다고 가정합니다. 이전 단계에서 설명한 대로가 AWS CLI 설치 및 구성된 별도의 디바이스 또는 인스턴스에서 온프레미스 인스턴스를 등록할 수도 있습니다.

를 사용하여 다음을 지정 AWS CLI 하여 register 명령을 호출합니다.

- CodeDeploy에 대한 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-name` 옵션 사용).

  > **Important**
  >
  > 나중에 특히 디버깅을 위해 온프레미스 인스턴스를 식별하려면 온프레미스 인스턴스의 일부 고유 특성(예: 일련 번호 또는 해당되는 경우 고유한 내부 자산 식별자)에 매핑되는 이름을 사용하는 것이 좋습니다. 이름에 MAC 주소를 지정하는 경우 MAC 주소에는 콜론(:)과 같이 CodeDeploy에서 허용하지 않는 문자가 포함되어 있다는 점에 유의하세요. 허용되는 문자 목록은 CodeDeploy 할당량 단원을 참조하세요.

- 선택적으로 이 온프레미스 인스턴스와 연결하려는 기존 IAM 사용자의 ARN(`--iam-user-arn` 옵션 사용). IAM 사용자의 ARN을 가져오려면 get-user 명령을 호출하거나 IAM 콘솔의 사용자 섹션에

서 IAM 사용자 이름을 선택한 다음 요약 섹션에서 사용자 ARN 값을 찾습니다. 이 옵션을 지정하지 않으면 CodeDeploy는 AWS 계정에서 사용자를 대신하여 IAM 사용자를 생성하고 이를 온프레미스 인스턴스와 연결합니다.

> **Important**
>
> `--iam-user-arn` 옵션을 지정하는 경우 4단계: 온프레미스 인스턴스에 구성 파일 추가에 설명된 대로 온프레미스 인스턴스 구성 파일도 수동으로 생성해야 합니다.
> 하나의 IAM 사용자만 하나의 온프레미스 인스턴스에만 연결할 수 있습니다. 단일 IAM 사용자를 여러 온프레미스 인스턴스와 연결하려고 하면 오류가 발생하거나, 이러한 온프레미스 인스턴스에 배포하지 못하거나, 영구 보류 상태로 유지된 온프레미스 인스턴스에 배포될 수 있습니다.

- 온프레미스 인스턴스 태그 집합(`--tags`옵션)을 사용하여 CodeDeploy Amazon EC2 인스턴스 집합을 식별하는 데 사용할 수 있습니다. `Key=`*`tag-key`*`,Value=`*`tag-value`*을(를) 사용하여 각 태그를 지정합니다(예: `Key=Name,Value=Beta  Key=Name,Value=WestRegion`). 이 옵션을 지정하지 않으면 태그가 등록되지 않습니다. 나중에 태그를 등록하려면 add-tags-to-on-premises-instances 명령을 호출하세요.
- 선택적으로 온프레미스 인스턴스가 CodeDeploy에 등록될 AWS 리전입니다(`--region` 옵션 사용). 이는의 리전 및 엔드포인트에 나열된 지원되는 리전 중 하나여야 합니다AWS 일반 참조(예: `us-west-2`). 이 옵션을 지정하지 않으면 호출하는 IAM 사용자와 연결된 기본 AWS 리전이 사용됩니다.

예제:

```
aws deploy register --instance-name AssetTag12010298EX --iam-user-
arn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem --tags
 Key=Name,Value=CodeDeployDemo-OnPrem --region us-west-2
```

register 명령은 다음 작업을 수행합니다.

1. 기존 IAM 사용자가 지정되지 않은 경우 IAM 사용자를 생성하고 필요한 권한을 연결하며 해당 보안 키와 보안 액세스 키를 생성합니다. 온프레미스 인스턴스는 이 IAM 사용자와 해당 권한 및 자격 증명을 사용하여 CodeDeploy를 인증하고 상호 작용합니다.
2. CodeDeploy에 온프레미스 인스턴스 등록
3. 지정된 경우 CodeDeploy에서 `--tags` 옵션으로 지정된 태그를 등록된 온프레미스 인스턴스 이름과 연결합니다.

4. IAM 사용자가 생성된 경우 register 명령이 호출된 동일한 디렉터리에 필요한 구성 파일도 생성합니다.

이 명령에 오류가 발생하면 나머지 단계를 수동으로 완료하는 방법을 설명하는 오류 메시지가 나타납니다. 그렇지 않으면 다음 단계에 나열된 대로 install 명령을 호출하는 방법을 설명하는 성공 메시지가 나타납니다.

3단계: 설치 명령 호출

온프레미스 인스턴스에서를 사용하여 다음을 지정 AWS CLI 하여 설치 명령을 호출합니다.

- 구성 파일의 경로(`--config-file` 옵션 사용).
- 선택적으로, 온프레미스 인스턴스에 이미 존재하는 구성 파일을 대체할지 여부(`--override-config` 옵션 사용). 지정하지 않으면 기존 구성 파일이 대체되지 않습니다.
- 선택적으로 온프레미스 인스턴스가 CodeDeploy에 등록될 AWS 리전입니다(`--region` 옵션 사용). 이는의 리전 및 엔드포인트에 나열된 지원되는 리전 중 하나여야 합니다AWS 일반 참조(예: `us-west-2`). 이 옵션을 지정하지 않으면 호출하는 IAM 사용자와 연결된 기본 AWS 리전이 사용됩니다.
- 선택적으로, CodeDeploy 에이전트를 설치할 사용자 지정 위치(`--agent-installer` 옵션 사용). 이 옵션은 CodeDeploy가 공식적으로 지원하지 않는 CodeDeploy 에이전트의 사용자 지정 버전(예: GitHub의 CodeDeploy 에이전트 리포지토리 기반 사용자 지정 버전)을 설치하는 데 유용합니다. 값은 다음 중 하나를 포함하는 Amazon S3 버킷의 경로여야 합니다.
  - CodeDeploy 에이전트 설치 스크립트(Linux 또는 UNIX 기반 운영 체제의 경우 GitHub의 CodeDeploy 에이전트 리포지토리에 있는 설치 파일과 유사).
  - CodeDeploy 에이전트 설치 관리자 패키지(.msi) 파일(Windows 기반 운영 체제의 경우)

  이 옵션을 지정하지 않으면 CodeDeploy는 자체 위치에서 온프레미스 인스턴스의 운영 체제와 호환되는 공식적으로 지원되는 CodeDeploy 에이전트 버전을 설치하는 것이 가장 좋습니다.

예:

```
aws deploy install --override-config --config-file /tmp/codedeploy.onpremises.yml --
region us-west-2 --agent-installer s3://aws-codedeploy-us-west-2/latest/codedeploy-
agent.msi
```

install 명령은 다음 작업을 수행합니다.

1. 온프레미스 인스턴스가 Amazon EC2 인스턴스인지 확인합니다. 이 경우 오류 메시지가 나타납니다.
2. 파일이 해당 위치에 없는 경우 인스턴스의 지정된 위치에서 CodeDeploy 에이전트가 해당 파일을 찾을 것으로 예상되는 위치로 온프레미스 인스턴스 구성 파일을 복사합니다.

   Ubuntu Server와 Red Hat Enterprise Linux(RHEL)의 경우, `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`.

   Windows Server의 경우 `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`.

   `--override-config` 옵션이 지정되면 파일을 만들거나 덮어씁니다.
3. 온프레미스 인스턴스에 CodeDeploy 에이전트를 설치한 다음 시작합니다.

4단계 : 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포합니다.

등록 및 태그가 지정된 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포할 준비가 되었습니다.

애플리케이션 개정 버전을 Amazon EC2 인스턴스에 배포하는 것과 유사한 방식으로 온프레미스 인스턴스에 배포합니다. 지침은 CodeDeploy에서 배포 만들기 단원을 참조하세요. 이러한 지침은 애플리케이션 만들기, 배포 그룹 만들기 및 애플리케이션 개정 버전 준비를 비롯한 필수 구성 요소에 연결됩니다. 배포하기 위해 간단한 샘플 애플리케이션 개정 버전이 필요한 경우 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)의 2단계: 샘플 애플리케이션 수정 버전 만들기에 설명된 개정 버전을 생성할 수 있습니다.

> **Important**
>
> 온프레미스 인스턴스를 대상으로 하는 배포 그룹을 만드는 과정에서 기존 CodeDeploy 서비스 역할을 다시 사용하는 경우 서비스 역할 정책 설명의 `Action` 부분에 `Tag:get*`을(를) 포함해야 합니다. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.

5단계: 온프레미스 인스턴스에 대한 배포 추적

등록되고 태그가 지정된 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포한 후 배포의 진행 상황을 추적할 수 있습니다.

Amazon EC2 인스턴스에 대한 배포를 추적하는 것과 유사한 방식으로 온프레미스 인스턴스에 대한 배포를 추적합니다. 지침은 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

더 많은 옵션은 CodeDeploy에서 온프레미스 인스턴스 작업 관리을 참조하세요.

register-on-premises-instance 명령(IAM 사용자 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다.

> **Important**
>
> IAM 사용자를 사용하여 인스턴스를 등록하는 것은 인증에 정적(영구) 보안 인증 정보를 사용하므로 권장되지 않습니다. 보안을 강화하려면 인증을 위한 임시 보안 인증 정보를 사용하여 인스턴스를 등록하는 것이 좋습니다. 자세한 내용은 register-on-premises-instance 명령(IAM 세션 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다. 단원을 참조하십시오.

> **Important**
>
> IAM 사용자의 액세스 키(영구 보안 인증 정보)를 교체할 계획이 있어야 합니다. 자세한 내용은 액세스 키 교체를 참조하세요.

다음 지침에 따라 온프레미스 인스턴스를 구성하고 인증에 정적 IAM 사용자 자격 증명을 사용하여 주로 CodeDeploy에 등록하고 태그를 지정합니다.

주제

- 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성
- 2단계: IAM 사용자에게 권한 할당
- 3단계: IAM 사용자 자격 증명을 가져오기
- 4단계: 온프레미스 인스턴스에 구성 파일 추가
- 5단계: 설치 및 구성 AWS CLI
- 6단계: AWS_REGION 환경 변수 설정(Ubuntu Server 및 RHEL만 해당)
- 7단계: CodeDeploy 에이전트 설치
- 8단계: CodeDeploy에 온프레미스 인스턴스 등록
- 9단계: 온프레미스 인스턴스에 태그 지정
- 10단계 : 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포합니다.
- 11단계: 온프레미스 인스턴스에 대한 배포 추적

1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성

온프레미스 인스턴스가 CodeDeploy를 인증하고 상호 작용하는 데 사용할 IAM 사용자를 생성합니다.

> **Important**
>
> 참여하는 각 온프레미스 인스턴스에 대해 별도의 IAM 사용자를 생성해야 합니다. 개별 IAM 사용자를 여러 온프레미스 인스턴스에 재사용하려고 하면 해당 온프레미스 인스턴스를 CodeDeploy로 등록하거나 태그를 지정하지 못할 수 있습니다. 이러한 온프레미스 인스턴스에 대한 배포는 영구 보류 상태로 유지되거나 완전히 실패할 수 있습니다.

IAM 사용자에게 해당 용도를 식별하는 이름(예: CodeDeployUser-OnPrem)을 할당하는 것이 좋습니다.

AWS CLI 또는 IAM 콘솔을 사용하여 IAM 사용자를 생성할 수 있습니다. 자세한 내용은 AWS 계정의 IAM 사용자 생성을 참조하세요.

> **Important**
>
> AWS CLI 또는 IAM 콘솔을 사용하여 새 IAM 사용자를 생성하든 상관없이 사용자에게 제공된 사용자 ARN을 기록해 둡니다. 나중에 4단계: 온프레미스 인스턴스에 구성 파일 추가 및 8단계: CodeDeploy에 온프레미스 인스턴스 등록에서 이 정보가 필요합니다.

2단계: IAM 사용자에게 권한 할당

온프레미스 인스턴스가 Amazon S3 버킷의 애플리케이션 개정 버전을 배포할 경우 IAM 사용자에게 해당 버킷과 상호 작용할 수 있는 권한을 할당해야 합니다. AWS CLI 또는 IAM 콘솔을 사용하여 권한을 할당할 수 있습니다.

> **Note**
>
> GitHub 리포지토리에서만 애플리케이션 개정 버전을 배포하는 경우이 단계를 건너 뛰고 3단계: IAM 사용자 자격 증명을 가져오기(으)로 이동합니다. (1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성에서 생성한 IAM 사용자에 대한 정보가 계속 필요합니다. 이후 단계에서 사용하게 됩니다.)

권한을 할당하려면(CLI)

1. AWS CLI을(를) 호출하는 데 사용하는 Amazon EC2 인스턴스 또는 디바이스에서 다음 정책 내용이 포함된 파일을 생성합니다. 파일의 이름을 **CodeDeploy-OnPrem-Permissions.json**와(과) 같이 지정한 다음 파일을 저장합니다.

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

> **Note**
>
> 온프레미스 인스턴스에서 액세스해야 하는 Amazon S3 버킷으로만 이 정책을 제한하는 것이 좋습니다. 이 정책을 제한하는 경우 AWS CodeDeploy 에이전트가 포함된 Amazon S3 버킷에 대한 액세스 권한도 부여해야 합니다. 그렇지 않으면 연결된 온프레미스 인스턴스에 CodeDeploy 에이전트를 설치 또는 업데이트할 때 오류가 발생할 수 있습니다. 예:
>
> ```
> {
>   "Version": "2012-10-17",
>   "Statement": [
>     {
>       "Effect": "Allow",
>       "Action": [
>         "s3:Get*",
>         "s3:List*"
>       ],
>       "Resource": [
>         "arn:aws:s3:::amzn-s3-demo-bucket/*",
>         "arn:aws:s3:::aws-codedeploy-us-east-2/*",
>         "arn:aws:s3:::aws-codedeploy-us-east-1/*",
> ```

```
        "arn:aws:s3:::aws-codedeploy-us-west-1/*",
        "arn:aws:s3:::aws-codedeploy-us-west-2/*",
        "arn:aws:s3:::aws-codedeploy-ca-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-west-3/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-central-2/*",
        "arn:aws:s3:::aws-codedeploy-eu-north-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-1/*",
        "arn:aws:s3:::aws-codedeploy-eu-south-2/*",
        "arn:aws:s3:::aws-codedeploy-il-central-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-east-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-northeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-2/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-3/*",
        "arn:aws:s3:::aws-codedeploy-ap-southeast-4/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-1/*",
        "arn:aws:s3:::aws-codedeploy-ap-south-2/*",
        "arn:aws:s3:::aws-codedeploy-me-central-1/*",
        "arn:aws:s3:::aws-codedeploy-me-south-1/*",
        "arn:aws:s3:::aws-codedeploy-sa-east-1/*"
      ]
    }
  ]
}
```

2. put-user-policy 명령을 호출하여 IAM 사용자 이름(`--user-name` 옵션 사용), 정책 이름(`--policy-name` 옵션 사용), 새로 생성된 정책 문서의 경로(`--policy-document` 옵션 사용)를 지정합니다. 예를 들어, **`CodeDeploy-OnPrem-Permissions.json`** 파일은 이 명령을 호출하는 것과 동일한 디렉터리(폴더)에 있다고 가정합니다.

> **Important**
>
> 파일 이름 앞에 `file://`를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws iam put-user-policy --user-name CodeDeployUser-OnPrem --policy-name CodeDeploy-
OnPrem-Permissions --policy-document file://CodeDeploy-OnPrem-Permissions.json
```

권한을 할당하려면(콘솔)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택한 후 정책 생성을 선택합니다. (시작하기 버튼이 표시되면 이 버튼을 선택한 후 정책 생성을 선택합니다.)
3. 자체 정책 생성 옆의 선택을 선택하세요.
4. 정책 이름) 상자에 이 정책의 이름(예: **CodeDeploy-OnPrem-Permissions**)을 입력합니다.
5. 정책 문서 상자에 다음 권한 표현식을 입력하거나 붙여 넣습니다. 그러면 AWS CodeDeploy 가 IAM 사용자를 대신하여 정책에 지정된 Amazon S3 버킷의 애플리케이션 개정을 온프레미스 인스턴스에 배포할 수 있습니다.

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

6. 정책 생성을 선택하세요.
7. 탐색 창에서 사용자를 선택합니다.
8. 사용자 목록에서 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성에서 생성한 IAM 사용자의 이름을 찾아서 선택합니다.
9. 권한 탭의 관리형 정책에서 정책 연결을 선택합니다.
10. **CodeDeploy-OnPrem-Permissions**(이)라는 정책을 선택한 후 정책 연결을 선택합니다.

3단계: IAM 사용자 자격 증명을 가져오기

IAM 사용자의 보안 키 ID와 보안 액세스 키를 가져옵니다. 4단계: 온프레미스 인스턴스에 구성 파일 추가을(를) 위해 이 둘이 필요합니다. AWS CLI 또는 IAM 콘솔을 사용하여 보안 키 ID와 보안 액세스 키를 가져올 수 있습니다.

> **Note**
>
> 보안 키 ID와 보안 액세스 키가 이미 있는 경우 이 단계를 건너뛰고 4단계: 온프레미스 인스턴스에 구성 파일 추가으(로) 직접 이동하세요.
> 사용자는 AWS 외부에서와 상호 작용하려는 경우 프로그래밍 방식으로 액세스해야 합니다 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 필요한 사용자	목적	방법
IAM	(권장) 콘솔 자격 증명을 임시 자격 증명으로 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 AWS 로컬 개발을 위한 로그인을 참조하세요. • AWS SDKs 경우 SDK 및 도구 참조 안내서의 AWS 로컬 개발을 위한 로그인을 참조하세요. AWS SDKs
작업 인력 ID (IAM Identity Center에서 관리되는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
		AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center 참조하세요. • AWS SDKs, 도구 및 AWS APIs의 경우 SDK 및 도구 참조 안내서의 IAM Identity Center 인증을 참조하세요. AWS SDKs
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용의 지침을 따릅니다.
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 IAM 사용자 자격 증명을 사용하여 인증을 참조하세요. • AWS SDKs 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용하여 인증을 참조하세요. AWS SDKs • AWS APIs 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

자격 증명을 가져오려면(CLI)

1. list-access-keys 명령을 호출하여 IAM 사용자의 이름(`--user-name` 옵션 사용)을 지정하고 액세스 키 ID만 쿼리합니다(`--query` 및 `--output` 옵션 사용). 예:

```
aws iam list-access-keys --user-name CodeDeployUser-OnPrem --query
 "AccessKeyMetadata[*].AccessKeyId" --output text
```

2. 출력에 키가 나타나지 않거나 출력에 하나의 키에 대한 정보만 나타나는 경우 create-access-key 명령을 호출하여 IAM 사용자의 이름을 지정합니다(`--user-name` 옵션 사용).

```
aws iam create-access-key --user-name CodeDeployUser-OnPrem
```

create-access-key 명령에 대한 호출의 출력에서 `AccessKeyId` 및 `SecretAccessKey` 필드의 값을 기록해 둡니다. 나중에 4단계: 온프레미스 인스턴스에 구성 파일 추가에서 이 정보가 필요합니다.

> **Important**
>
> 지금이 보안 액세스 키에 액세스할 수 있는 유일한 시간입니다. 이 보안 액세스 키에 대한 액세스 권한을 잊어버리거나 분실한 경우 3단계: IAM 사용자 자격 증명을 가져오기의 단계에 따라 새 액세스 키를 생성해야 합니다.

3. 두 개의 액세스 키가 이미 나열되어 있는 경우 delete-access-key 명령을 호출하여 IAM 사용자의 이름(`--user-name` 옵션 사용) 및 삭제할 액세스 키의 ID(`--access-key-id` 옵션 사용)를 지정합니다. 그런 다음 이 단계의 앞부분에서 설명한 대로 create-access-key 명령을 호출합니다. 다음은 delete-access-key 명령 호출 예제입니다.

```
aws iam delete-access-key --user-name CodeDeployUser-OnPrem --access-key-id access-key-ID
```

> **Important**
>
> delete-access-key 명령을 호출하여 이러한 액세스 키 중 하나를 삭제하고 온프레미스 인스턴스가 4단계: 온프레미스 인스턴스에 구성 파일 추가에 설명된 대로 이미 이 액세스 키를 사용하고 있는 경우 4단계: 온프레미스 인스턴스에 구성 파일 추가의 지침에 따라 이 IAM 사용자와 연결된 다른 액세스 키 ID 및 보안 액세스 키를 지정해야 합니다. 그렇지 않

으면 해당 온프레미스 인스턴스에 대한 배포가 영구 보류 상태로 유지되거나 완전히 실패할 수 있습니다.

자격 증명을 가져오려면(콘솔)

1. a. IAM 콘솔(https://console.aws.amazon.com/iam/)을 엽니다.
   b. 사용자 목록이 표시되지 않으면 탐색 창에서 사용자를 선택합니다.
   c. 사용자 목록에서 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성에서 생성한 IAM 사용자의 이름을 찾아서 선택합니다.
2. 보안 자격 증명 탭에서 키가 없거나 키가 하나만 나열되어 있는 경우 액세스 키 생성을 선택합니다.

   두 개의 액세스 키가 나열되는 경우 그 중 하나를 삭제해야 합니다. 액세스 키 중 하나 옆에 있는 삭제를 선택한 다음 액세스 키 생성을 선택합니다.

   > **Important**
   >
   > 이러한 액세스 키 중 하나 옆에 있는삭제를 선택한 경우, 온프레미스 인스턴스가 4단계: 온프레미스 인스턴스에 구성 파일 추가에 설명된 대로 이미 이 액세스 키를 사용하고 있다면 다시 4단계: 온프레미스 인스턴스에 구성 파일 추가의 지침에 따라 이 IAM 사용자와 연결된 다른 액세스 키 ID 및 보안 액세스 키를 지정해야 합니다. 그렇지 않으면 해당 온프레미스 인스턴스에 대한 배포가 영구 보류 상태로 유지되거나 완전히 실패할 수 있습니다.

3. 표시를 선택하고 액세스 키 ID와 보안 액세스 키를 기록해 둡니다. 다음 단계에서 이 정보가 필요합니다. 또는 .csv 파일 다운로드를 선택하여 액세스 키 ID와 보안 액세스 키의 복사본을 저장합니다.

   > **Important**
   >
   > 자격 증명을 기록하거나 다운로드하지 않는 한 지금이 이 보안 액세스 키에 액세스할 수 있는 유일한 시간입니다. 이 보안 액세스 키에 대한 액세스 권한을 잊어버리거나 분실한 경우 3단계: IAM 사용자 자격 증명을 가져오기의 단계에 따라 새 액세스 키를 생성해야 합니다.

4. 닫기를 선택하여 사용자 > ***IAM ### ##*** 페이지로 돌아옵니다.

4단계: 온프레미스 인스턴스에 구성 파일 추가

루트 또는 관리자 권한을 사용하여 온프레미스 인스턴스에 구성 파일을 추가합니다. 이 구성 파일은 CodeDeploy에 사용할 IAM 사용자 자격 증명과 대상 AWS 리전을 선언하는 데 사용됩니다. 온프레미스 인스턴스의 특정 위치에 파일을 추가해야 합니다. 파일에는 IAM 사용자의 ARN, 보안 키 ID, 보안 액세스 키 및 대상 AWS 리전이 포함되어야 합니다. 파일은 특정 형식을 따라야 합니다.

1. 온프레미스 인스턴스의 다음 위치에 `codedeploy.onpremises.yml`(Ubuntu Server 또는 RHEL 온프레미스 인스턴스의 경우) 또는 `conf.onpremises.yml`(Windows Server 온-레미스 인스턴스의 경우)라는 이름의 파일을 만듭니다.

   - Ubuntu Server: `/etc/codedeploy-agent/conf`
   - Windows Server: `C:\ProgramData\Amazon\CodeDeploy`

2. 텍스트 편집기를 사용하여 새로 만든 `codedeploy.onpremises.yml` 또는 `conf.onpremises.yml` 파일에 다음 정보를 추가하세요.

   ```
   ---
   aws_access_key_id: secret-key-id
   aws_secret_access_key: secret-access-key
   iam_user_arn: iam-user-arn
   region: supported-region
   ```

   위치:

   - *`secret-key-id`*는 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성 또는 3단계: IAM 사용자 자격 증명을 가져오기에서 기록해 둔 해당 IAM 사용자의 보안 키 ID입니다.
   - *`secret-access-key`*는 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성 또는 3단계: IAM 사용자 자격 증명을 가져오기에서 기록해 둔 해당 IAM 사용자의 보안 액세스 키입니다.
   - *`iam-user-arn`*는 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성에서 앞서 기록해 둔 IAM 사용자의 ARN 입니다.
   - *`supported-region`*은 CodeDeploy에서 지원하는 리전의 식별자로, CodeDeploy 애플리케이션, 배포 그룹 및 애플리케이션 개정이 위치합니다(예: `us-west-2`). 리전 목록은의 리전 및 엔드포인트를 참조하세요AWS 일반 참조.

> **Important**
>
> 3단계: IAM 사용자 자격 증명을 가져오기에서 이러한 액세스 키 중 하나 옆에 있는 삭제를 선택한 경우, 온프레미스 인스턴스가 이미 연결된 이 액세스 키 ID와 보안 액세스 키를 사용하고 있다면 4단계: 온프레미스 인스턴스에 구성 파일 추가의 지침에 따라 이 IAM 사용자와 연결된 다른 액세스 키 ID 및 보안 액세스 키를 지정해야 합니다. 그렇지 않으면 해당 온프레미스 인스턴스에 대한 배포가 영구 보류 상태로 유지되거나 완전히 실패할 수 있습니다.

5단계: 설치 및 구성 AWS CLI

온프레미스 인스턴스 AWS CLI 에를 설치하고 구성합니다. (AWS CLI 는에서 온프레미스 인스턴스에 CodeDeploy 에이전트를 다운로드하고 설치하는 7단계: CodeDeploy 에이전트 설치 데 사용됩니다.)

1. 온프레미스 인스턴스 AWS CLI 에를 설치하려면 AWS Command Line Interface 사용 설명서의 로 설정 AWS CLI의 지침을 따르세요.

   > **Note**
   >
   > 온프레미스 인스턴스 작업을 위한 CodeDeploy 명령은 AWS CLI의 버전 1.7.19에서 사용할 수 있게 되었습니다. 버전이 AWS CLI 이미 설치되어 있는 경우를 호출하여 버전을 확인할 수 있습니다aws --version.

2. 온프레미스 인스턴스 AWS CLI 에서를 구성하려면 AWS Command Line Interface 사용 설명서의 구성 AWS CLI의 지침을 따르세요.

   > **Important**
   >
   > 를 구성할 때 AWS CLI (예: aws configure 명령을 호출하여) 최소한에 지정된 액세스 권한 외에도 다음 AWS 액세스 권한이 있는 IAM 사용자의 보안 키 ID와 보안 액세스 키를 지정해야 합니다온프레미스 인스턴스 구성을 위한 사전 요구 사항. 이렇게 하면 온프레미스 인스턴스에 CodeDeploy 에이전트를 다운로드하고 설치할 수 있습니다.
   >
   > JSON
   >
   > ```
   > {
   > ```

```
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : "*"
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource" : [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

JSON

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

```
      "Resource" : [
        "*"
      ]
    }
  ]
}
```

이러한 액세스 권한은 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성 또는 다른 IAM 사용자에게 연결할 수 있습니다. 이러한 권한을 IAM 사용자에게 할당하려면 해당 단계의 액세스 권한 대신 이러한 액세스 권한을 사용하여 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성의 지침을 따릅니다.

6단계: AWS_REGION 환경 변수 설정(Ubuntu Server 및 RHEL만 해당)

온프레미스 인스턴스에서 Ubuntu Server 또는 RHEL을 실행하지 않는 경우 이 단계를 건너뛰고 7단계: CodeDeploy 에이전트 설치으(로) 직접 이동합니다.

Ubuntu Server 또는 RHEL 온프레미스 인스턴스에 CodeDeploy 에이전트를 설치하고 새 버전을 사용할 수 있을 때마다 인스턴스가 CodeDeploy 에이전트를 업데이트하도록 설정합니다. 이렇게 하려면 인스턴스의 `AWS_REGION` 환경 변수를 CodeDeploy에서 지원하는 리전 중 하나의 식별자로 설정합니다. CodeDeploy 애플리케이션, 배포 그룹 및 애플리케이션 개정이 위치한 리전(예: `us-west-2`)으로 값을 설정하는 것이 좋습니다. 리전 목록은의 리전 및 엔드포인트를 참조하세요AWS 일반 참조.

환경 변수를 설정하려면 터미널에서 다음을 호출하세요.

```
export AWS_REGION=supported-region
```

여기서 *`supported-region`*은 리전 식별자입니다(예: `us-west-2`).

7단계: CodeDeploy 에이전트 설치

온프레미스 인스턴스에 CodeDeploy 에이전트 설치

- Ubuntu Server 온프레미스 인스턴스의 경우 Ubuntu Server용 CodeDeploy 에이전트 설치의 지침을 따르고 이 페이지로 돌아옵니다.
- RHEL 온프레미스 인스턴스의 경우 Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치의 지침을 따르고 이 페이지로 돌아옵니다.

- Windows Serve 온프레미스 인스턴스의 경우 Windows Server용 CodeDeploy 에이전트 설치의 지침을 따르고 이 페이지로 돌아옵니다.

8단계: CodeDeploy에 온프레미스 인스턴스 등록

이 단계의 지침은 온프레미스 인스턴스 자체에서 온프레미스 인스턴스를 등록한다고 가정합니다. 에 설명된 대로가 AWS CLI 설치 및 구성된 별도의 디바이스 또는 인스턴스에서 온프레미스 인스턴스를 등록할 수 있습니다5단계: 설치 및 구성 AWS CLI.

AWS CLI 를 사용하여 배포에 사용할 수 있도록 CodeDeploy에 온프레미스 인스턴스를 등록합니다.

1. 를 사용하려면 먼저에서 생성한 IAM 사용자의 사용자 ARN AWS CLI이 필요합니다1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성. 아직 사용자 ARN 이 없는 경우 get-user 명령을 호출하여 IAM 사용자의 이름(`--user-name` 옵션 사용)을 지정하고 사용자 ARN만 쿼리합니다(`--query` 및 `--output` 옵션 사용).

   ```
   aws iam get-user --user-name CodeDeployUser-OnPrem --query "User.Arn" --output text
   ```

2. register-on-premises-instance 명령을 호출하여 다음을 지정합니다.

   - 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-name` 옵션 사용).

     > **Important**
     >
     > 특히 디버깅을 위해 온프레미스 인스턴스를 식별하려면 온프레미스 인스턴스의 일부 고유 특성(예: 일련 번호 또는 해당되는 경우 내부 자산 식별자)에 매핑되는 이름을 지정하는 것이 좋습니다. 이름으로 MAC 주소를 지정하는 경우 MAC 주소에는 콜론(`:`)과 같이 CodeDeploy에서 허용하지 않는 문자가 포함되어 있다는 점에 유의하세요. 허용되는 문자 목록은 CodeDeploy 할당량 단원을 참조하세요.

   - 1단계: 온프레미스 인스턴스에 대한 IAM 사용자 생성에서 생성한 IAM 사용자의 사용자 ARN(`--iam-user-arn` 옵션 사용).

     예:

     ```
     aws deploy register-on-premises-instance --instance-name AssetTag12010298EX --
     iam-user-arn arn:aws:iam::444455556666:user/CodeDeployUser-OnPrem
     ```

9단계: 온프레미스 인스턴스에 태그 지정

AWS CLI 또는 CodeDeploy 콘솔을 사용하여 온프레미스 인스턴스에 태그를 지정할 수 있습니다. CodeDeploy는 온프레미스 인스턴스 태그를 사용하여 배포 중 배포 대상을 식별합니다.

온프레미스 인스턴스에 태그 지정(CLI)

- add-tags-to-on-premises-instances 명령을 호출하여 다음을 지정합니다.

  - 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-names` 옵션 사용).
  - 사용하려는 온프레미스 인스턴스 태그 키 및 태그 값의 이름(`--tags` 옵션 사용). 이름과 값을 둘 다 지정해야 합니다. CodeDeploy는 값만 있는 온프레미스 인스턴스 태그를 허용하지 않습니다.

    예:

```
aws deploy add-tags-to-on-premises-instances --instance-names AssetTag12010298EX
 --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

온프레미스 인스턴스에 태그 지정(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. CodeDeploy 메뉴에서 온프레미스 인스턴스를 선택합니다.
3. 온프레미스 인스턴스 목록에서 태그를 지정할 온프레미스 인스턴스 옆에 있는 화살표를 선택합니다.
4. 태그 목록에서 원하는 태그 키와 태그 값을 선택하거나 입력합니다. 태그 키와 태그 값을 입력하면 다른 행이 나타납니다. 최대 10개의 태그에 대해 이 작업을 반복할 수 있습니다. 태그를 제거하려면 삭제 아이콘 ()을 선택합니다.
5. 태그를 추가한 후 태그 업데이트를 선택합니다.

10단계 : 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포합니다.

등록 및 태그가 지정된 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포할 준비가 되었습니다.

애플리케이션 개정 버전을 Amazon EC2 인스턴스에 배포하는 것과 유사한 방식으로 온프레미스 인스턴스에 배포합니다. 지침은 CodeDeploy에서 배포 만들기 단원을 참조하세요. 이러한 지침은 애플리케이션 만들기, 배포 그룹 만들기 및 애플리케이션 개정 버전 준비를 비롯한 필수 구성 요소에 대한 링크가 포함되어 있습니다. 배포하기 위해 간단한 샘플 애플리케이션 개정 버전이 필요한 경우 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)의 2단계: 샘플 애플리케이션 수정 버전 만들기에 설명된 개정 버전을 생성할 수 있습니다.

> **Important**
>
> 온프레미스 인스턴스를 대상으로 하는 배포 그룹을 만드는 과정에서 CodeDeploy 서비스 역할을 다시 사용하는 경우 서비스 역할 정책 설명의 `Action` 부분에 `Tag:get*`을(를) 포함해야 합니다. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.

11단계: 온프레미스 인스턴스에 대한 배포 추적

등록되고 태그가 지정된 온프레미스 인스턴스에 애플리케이션 개정 버전을 배포한 후 배포의 진행 상황을 추적할 수 있습니다.

Amazon EC2 인스턴스에 대한 배포를 추적하는 것과 유사한 방식으로 온프레미스 인스턴스에 대한 배포를 추적합니다. 지침은 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

CodeDeploy에서 온프레미스 인스턴스 작업 관리

CodeDeploy에 등록한 후 온프레미스 인스턴스의 작업을 관리하려면 이 섹션의 지침에 따라 자세한 정보 가져오기, 태그 제거, 온프레미스 인스턴스 제거 및 등록 취소 등의 작업을 관리할 수 있습니다.

주제

- 단일 온프레미스 인스턴스에 대한 정보를 가져옵니다.
- 여러 온프레미스 인스턴스 정보 가져오기
- 온프레미스 인스턴스에서 온프레미스 인스턴스 태그 수동 제거
- CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일 제거
- 온프레미스 인스턴스 등록 자동 취소
- 온프레미스 인스턴스 등록을 수동으로 취소합니다.

단일 온프레미스 인스턴스에 대한 정보를 가져옵니다.

CodeDeploy 배포 세부 정보 보기의 지침에 따라 단일 온프레미스 인스턴스에 대한 정보를 가져오는 데 필요합니다. AWS CLI 또는 CodeDeploy 콘솔을 사용하여 단일 온프레미스 인스턴스에 대한 자세한 정보를 얻을 수 있습니다.

단일 온프레미스 인스턴스에 대한 정보를 가져오려면(CLI)

- get-on-premises-instance 명령을 호출하여 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-name` 옵션 사용)을 지정합니다.

```
aws deploy get-on-premises-instance --instance-name AssetTag12010298EX
```

단일 온프레미스 인스턴스에 대한 정보를 가져오려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 온프레미스 인스턴스를 선택합니다.
3. 온프레미스 인스턴스 목록에서 세부 정보를 표시할 온프레미스 인스턴스의 이름을 선택합니다.

여러 온프레미스 인스턴스 정보 가져오기

CodeDeploy 배포 세부 정보 보기의 지침에 따라 온프레미스 인스턴스에 대한 정보를 가져올 수 있습니다. AWS CLI 또는 CodeDeploy 콘솔을 사용하여 온프레미스 인스턴스에 대한 자세한 정보를 얻을 수 있습니다.

여러 온프레미스 인스턴스에 대한 정보를 가져오려면(CLI)

1. 온프레미스 인스턴스 이름의 목록을 보려면 list-on-premises-instances 명령을 호출하여 다음을 지정합니다.

   - 등록 또는 등록 취소된 모든 온프레미스 인스턴스에 대한 정보를 가져올지 여부(각각 `--registration-status`, `Registered` 또는 `Deregistered` 옵션 사용). 이 옵션을 생략하면

등록된 온프레미스 인스턴스 이름과 등록 취소된 온프레미스 인스턴스 이름이 모두 반환됩니다.

- 특정 온프레미스 인스턴스 태그로 태그가 지정된 온프레미스 인스턴스에 대한 정보만 가져오는지 여부(`--tag-filters` 옵션). 각 온프레미스 인스턴스 태그에 대해 `Key`, `Value`, 및 `Type`(항상`KEY_AND_VALUE`여야 함)을 지정합니다. 여러 온프레미스 인스턴스 태그는 각각 `Key`, `Value`, `Type` 삼중항 사이에 공백으로 구분합니다..

예:

```
aws deploy list-on-premises-instances --registration-status Registered
 --tag-filters Key=Name,Value=CodeDeployDemo-OnPrem,Type=KEY_AND_VALUE
 Key=Name,Value=CodeDeployDemo-OnPrem-Beta,Type=KEY_AND_VALUE
```

2. 자세한 내용은 온프레미스 인스턴스의 이름(`--instance-names` 옵션 사용)과 함께 batch-get-on-premises-instances 명령을 호출합니다.

```
aws deploy batch-get-on-premises-instances --instance-names AssetTag12010298EX
 AssetTag09920444EX
```

여러 온프레미스 인스턴스에 대한 정보를 가져오려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 온프레미스 인스턴스를 선택합니다.

   온프레미스 인스턴스에 대한 자세한 정보가 표시됩니다.

온프레미스 인스턴스에서 온프레미스 인스턴스 태그 수동 제거

일반적으로 해당 태그가 더 이상 사용되지 않는 경우 온프레미스 인스턴스에서 온프레미스 인스턴스 태그를 제거하거나 해당 태그에 의존하는 배포 그룹에서 온프레미스 인스턴스를 제거하려고 합니다.

AWS CLI 또는 AWS CodeDeploy 콘솔을 사용하여 온프레미스 인스턴스에서 온프레미스 인스턴스 태그를 제거할 수 있습니다.

온프레미스 인스턴스를 등록 취소하기 전에 온프레미스 인스턴스에서 인스턴스를 제거할 필요가 없습니다.

온프레미스 인스턴스에서 온프레미스 인스턴스 태그를 수동으로 제거해도 인스턴스가 등록 취소되지 않습니다. 인스턴스에서 CodeDeploy 에이전트를 제거하지는 않습니다. 인스턴스에서 구성 파일은 제거되지 않습니다. 인스턴스와 연결된 IAM 사용자는 삭제되지 않습니다.

온프레미스 인스턴스의 등록을 자동으로 취소하는 방법은 온프레미스 인스턴스 등록 자동 취소 단원을 참조하세요.

온프레미스 인스턴스의 등록을 수동으로 취소하는 방법은 온프레미스 인스턴스 등록을 수동으로 취소합니다. 단원을 참조하세요.

CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일을 제거하려면 CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일 제거 단원을 참조하세요.

온프레미스 인스턴스에서 CodeDeploy 에이전트만 수동으로 제거하려면 CodeDeploy 에이전트 작업 관리 단원을 참조하세요.

연결된 IAM 사용자를 수동으로 삭제하려면 AWS 계정에서 IAM 사용자 삭제를 참조하세요.

온프레미스 인스턴스에서 온프레미스 인스턴스 태그 제거(CLI)

- remove-tags-from-on-premises-instances를 호출하고 다음을 지정합니다.

  - 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-names` 옵션 사용).
  - 제거하고자 하는 태그의 이름과 값(`--tags` 옵션 사용).

    예:

```
aws deploy remove-tags-from-on-premises-instances --instance-names
 AssetTag12010298EX --tags Key=Name,Value=CodeDeployDemo-OnPrem
```

온프레미스 인스턴스에서 온프레미스 인스턴스 태그 제거(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 온프레미스 인스턴스를 선택합니다.
3. 온프레미스 인스턴스 목록에서 태그를 제거할 온프레미스 인스턴스의 이름을 선택합니다.
4. 태그에서 제거할 각 태그 옆의 제거를 선택합니다.
5. 태그를 삭제한 후 태그 업데이트를 선택합니다.

CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일 제거

일반적으로 더 이상 배포할 계획이 없으면 CodeDeploy 에이전트를 제거하고 온프레미스 인스턴스에서 구성 파일을 제거합니다.

> **Note**
>
> CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일을 제거해도 온프레미스 인스턴스의 등록이 취소되지 않습니다. 온프레미스 인스턴스와 연결된 모든 온프레미스 인스턴스 태그의 연결은 해제되지 않습니다. 온프레미스 인스턴스와 연결된 IAM 사용자는 삭제되지 않습니다.
> 온프레미스 인스턴스의 등록을 자동으로 취소하는 방법은 온프레미스 인스턴스 등록 자동 취소 단원을 참조하세요.
> 온프레미스 인스턴스의 등록을 수동으로 취소하는 방법은 온프레미스 인스턴스 등록을 수동으로 취소합니다. 단원을 참조하세요.
> 연결된 온프레미스 인스턴스 태그의 연결을 수동으로 해제하려면 온프레미스 인스턴스에서 온프레미스 인스턴스 태그 수동 제거 단원을 참조하세요.
> 온프레미스 인스턴스에서 CodeDeploy 에이전트를 수동으로 제거하려면 CodeDeploy 에이전트 작업 관리 단원을 참조하세요.
> 연결된 IAM 사용자를 수동으로 삭제하려면 AWS 계정에서 IAM 사용자 삭제를 참조하세요.

온프레미스 인스턴스에서 AWS CLI 를 사용하여 제거 명령을 호출합니다.

예제:

```
aws deploy uninstall
```

uninstall 명령은 다음 작업을 수행합니다.

1. 온프레미스 인스턴스에서 실행 중인 CodeDeploy 에이전트를 중지합니다.
2. 온프레미스 인스턴스에서 CodeDeploy 에이전트를 제거합니다.
3. 온프레미스 인스턴스에서 구성 파일을 제거합니다. (Ubuntu Server와 RHEL의 경우, `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml`. Windows Server의 경우 `C:\ProgramData\Amazon\CodeDeploy\conf.onpremises.yml`.)

온프레미스 인스턴스 등록 자동 취소

일반적으로 온프레미스 인스턴스에 더 이상 배포할 계획이 없는 경우 등록을 취소합니다. 온프레미스 인스턴스의 등록을 취소하면 온프레미스 인스턴스가 배포 그룹의 온프레미스 인스턴스 태그의 일부일 수 있지만 온프레미스 인스턴스는 배포에 포함되지 않습니다. 를 사용하여 온프레미스 인스턴스의 등록 AWS CLI 을 취소할 수 있습니다.

> **Note**
>
> CodeDeploy 콘솔을 사용하여 온프레미스 인스턴스를 등록 취소할 수 없습니다. 또한 온프레미스 인스턴스를 등록 취소하면 온프레미스 인스턴스와 연결된 모든 온프레미스 인스턴스 태그가 제거됩니다. 온프레미스 인스턴스에서 CodeDeploy 에이전트를 제거하지 않습니다. 온프레미스 인스턴스에서 온프레미스 인스턴스 구성 파일은 제거하지 않습니다.
> CodeDeploy 콘솔을 사용하여 이 섹션의 일부 (전부는 아님) 작업을 수행하려면 온프레미스 인스턴스 등록을 수동으로 취소합니다.의 CodeDeploy 콘솔 섹션을 참조하세요.
> 연결된 온프레미스 인스턴스 태그의 연결을 수동으로 해제하려면 온프레미스 인스턴스에서 온프레미스 인스턴스 태그 수동 제거 단원을 참조하세요.
> CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일을 제거하려면 CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일 제거 단원을 참조하세요.
> 온프레미스 인스턴스에서 CodeDeploy 에이전트만 수동으로 제거하려면 CodeDeploy 에이전트 작업 관리 단원을 참조하세요.

를 사용하여 등록 취소 명령을 호출 AWS CLI 하고 다음을 지정합니다.

- CodeDeploy에 대한 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-name` 옵션 사용).
- (선택 사항) 온프레미스 인스턴스와 연결된 IAM 사용자를 삭제할지 여부. 기본 동작은 IAM 사용자를 삭제하는 것입니다. 온프레미스 인스턴스와 연결된 IAM 사용자를 삭제하지 않으려는 경우, 명령에서 `--no-delete-iam-user` 옵션을 지정하세요.
- 선택적으로 온프레미스 인스턴스가 CodeDeploy에 등록된 AWS 리전입니다(`--region` 옵션 사용). 이는의 리전 및 엔드포인트에 나열된 지원되는 리전 중 하나여야 합니다AWS 일반 참조(예: `us-west-2`). 이 옵션을 지정하지 않으면 호출하는 IAM 사용자와 연결된 기본 AWS 리전이 사용됩니다.

인스턴스를 등록 취소하고 사용자를 삭제하는 예는 다음과 같습니다.

```
aws deploy deregister --instance-name AssetTag12010298EX --region us-west-2
```

인스턴스를 등록 취소하고 사용자를 삭제하지 않는 예는 다음과 같습니다.

```
aws deploy deregister --instance-name AssetTag12010298EX --no-delete-iam-user --region
 us-west-2
```

deregister 명령은 다음 작업을 수행합니다.

1. CodeDeploy에서 온프레미스 인스턴스의 등록을 취소합니다.
2. 지정된 경우 온프레미스 인스턴스와 연결된 IAM 사용자를 삭제합니다.

온프레미스 인스턴스 등록을 취소한 후

- 콘솔에 즉시 나타나지 않습니다.
- 같은 이름의 다른 인스턴스를 즉시 만들 수 있습니다.

이 명령에 오류가 발생하면 나머지 단계를 수동으로 완료하는 방법을 설명하는 오류 메시지가 나타납니다. 그렇지 않으면 uninstall 명령을 호출하는 방법을 설명하는 성공 메시지가 나타납니다.

온프레미스 인스턴스 등록을 수동으로 취소합니다.

일반적으로 온프레미스 인스턴스에 더 이상 배포할 계획이 없는 경우 등록을 취소합니다. 를 사용하여 온프레미스 인스턴스의 등록 AWS CLI 을 수동으로 취소할 수 있습니다.

온프레미스 인스턴스를 수동으로 등록 취소해도 CodeDeploy 에이전트가 제거되지는 않습니다. 인스턴스에서 구성 파일은 제거되지 않습니다. 인스턴스와 연결된 IAM 사용자는 삭제되지 않습니다. 인스턴스와 연결된 태그는 제거하지 않습니다.

CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일을 제거하려면 CodeDeploy 에이전트를 자동으로 제거하고 온프레미스 인스턴스에서 구성 파일 제거 단원을 참조하세요.

CodeDeploy 에이전트만 수동으로 제거하려면 CodeDeploy 에이전트 작업 관리 단원을 참조하세요.

연결된 IAM 사용자를 수동으로 삭제하려면 AWS 계정에서 IAM 사용자 삭제를 참조하세요.

연결된 온프레미스 인스턴스 태그만 수동으로 제거하려면 온프레미스 인스턴스에서 온프레미스 인스턴스 태그 수동 제거 단원을 참조하세요.

- deregister-on-premises-instance 명령을 호출하여 온프레미스 인스턴스를 고유하게 식별하는 이름(`--instance-name` 옵션 사용)을 지정합니다.

  ```
  aws deploy deregister-on-premises-instance --instance-name AssetTag12010298EX
  ```

  온프레미스 인스턴스 등록을 취소한 후

  - 콘솔에 즉시 나타나지 않습니다.
  - 같은 이름의 다른 인스턴스를 즉시 만들 수 있습니다.

CodeDeploy에서 인스턴스 정보 보기

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy APIs를 사용하여 배포에 사용되는 인스턴스에 대한 세부 정보를 볼 수 있습니다.

CodeDeploy API 작업을 사용하여 인스턴스를 보는 방법에 대한 자세한 내용은 GetDeploymentInstance, ListDeploymentInstances 및 ListOnPremisesInstances를 참조하세요.

주제

- 인스턴스 정보 보기(콘솔)
- 인스턴스 정보 보기(CLI)

인스턴스 정보 보기(콘솔)

인스턴스 정보를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 배포를 선택합니다.

   > **Note**
   >
   > 항목이 표시되지 않으면 올바른 리전이 선택되어 있는지 확인합니다. 탐색 모음의 리전 선택기에서 AWS 일반 참조의 리전 및 엔드포인트에 나열된 리전 중 하나를 선택합니다. CodeDeploy는 이러한 리전에서만 지원됩니다.

3. 배포 세부 정보를 표시하려면 인스턴스의 배포 ID를 선택합니다.
4. 배포 페이지의 Instance activity 섹션에서 모든 인스턴스를 볼 수 있습니다.
5. 인스턴스의 개별 배포 수명 주기 이벤트에 대한 정보를 보려면 배포 세부 정보 페이지의이벤트 열에서 이벤트 보기를 선택합니다.

   > **Note**
   >
   > 수명 주기 이벤트에 대해 실패함이 표시되면 인스턴스 세부 정보 페이지에서 로그 보기나 EC2에서 보기 또는 둘 다를 선택합니다. 문제 해결 팁은 인스턴스 문제 해결에서 확인할 수 있습니다.

6. Amazon EC2 인스턴스에 대한 자세한 정보를 보려면 인스턴스 ID 열에서 인스턴스의 ID를 선택합니다.

인스턴스 정보 보기(CLI)

AWS CLI 를 사용하여 인스턴스 세부 정보를 보려면 `get-deployment-instance` 명령 또는 `list-deployment-instances` 명령을 호출합니다.

단일 인스턴스에 대한 세부 정보를 보려면 get-deployment-instance 명령을 호출하여, 다음을 지정합니다.

- 고유한 배포 ID입니다. 배포 ID를 가져오려면 list-deployments 명령을 호출합니다.
- 고유한 인스턴스 ID입니다. 인스턴스 ID를 가져오려면 list-deployment-instances 명령을 호출합니다.

배포에 사용된 인스턴스의 ID 목록을 보려면 list-deployment-instances명령을 호출하여 다음을 지정합니다.

- 고유한 배포 ID입니다. 배포 ID를 가져오려면 list-deployments 명령을 호출합니다.
- 선택적으로 배포 상태에 따라 특정 인스턴스 ID만 포함할지 여부를 지정할 수 있습니다. (지정하지 않으면 배포 상태에 관계없이 일치하는 모든 인스턴스 ID가 나열됩니다.)

CodeDeploy 인스턴스 상태

CodeDeploy는 배포 그룹에 있는 인스턴스의 상태를 모니터링합니다. 정상 인스턴스 수가 배포 중에 배포 그룹에 대해 지정된 최소 정상 인스턴스 수 미만으로 떨어지면 배포에 실패합니다. 예를 들어 배포 중에 85%의 인스턴스를 정상 상태로 유지해야 하고 배포 그룹에 10개의 인스턴스가 포함된 경우, 만약 단일 인스턴스의 배포에 실패하면 전체 배포도 실패합니다. 이는 최신 애플리케이션 개정을 설치할 수 있도록 인스턴스를 오프라인으로 전환할 때 사용 가능한 정상 인스턴스 수가 이미 90%로 감소하기 때문입니다. 각각 하나씩의 실패한 인스턴스와 오프라인 인스턴스 때문에 80%의 인스턴스만 정상 상태이면서 사용 가능한 상태입니다. CodeDeploy가 전체 배포에 실패합니다.

전체 배포에 성공하려면 다음 조건이 참이어야 합니다.

- CodeDeploy가 배포 중에 각 인스턴스에 배포할 수 있어야 합니다.
- 하나 이상의 인스턴스에 배포를 성공해야 합니다. 즉, 최소 정상 호스트 값이 0인 경우에도 전체 배포에 성공하려면 하나 이상의 인스턴스에 대한 배포를 성공해야 합니다(즉, 하나 이상의 인스턴스가 정상이어야 함).

주제

- 상태 확인
- 최소 정상 인스턴스 수 정보
- 가용 영역당 최소 정상 인스턴스 수 정보

상태 확인

CodeDeploy는 각 인스턴스에 두 개의 상태 값(수정 버전 상태 및 인스턴스 상태)을 할당합니다.

수정 버전 상태

수정 버전 상태는 인스턴스에 현재 설치되어 있는 애플리케이션 수정 버전을 기반으로 합니다. 다음과 같은 값이 표시될 수 있습니다.

- 현재: 배포 그룹이 마지막으로 배포에 성공한 수정 버전과 인스턴스에 설치된 수정 버전이 일치합니다.
- 이전: 인스턴스에 설치된 수정 버전이 애플리케이션의 이전 버전과 일치합니다.
- 알 수 없음: 애플리케이션의 수정 버전이 인스턴스에 성공적으로 설치되지 않았습니다.

인스턴스 상태

인스턴스 상태는 인스턴스에 대한 배포를 성공했는지 여부에 따라 결정됩니다. 다음과 같은 값을 가질 수 있습니다.

- 정상: 인스턴스에 대한 마지막 배포에 성공했습니다.
- 비정상: 인스턴스에 수정 버전을 배포하려는 시도가 실패했거나 수정 버전이 아직 인스턴스에 배포되지 않았습니다.

CodeDeploy는 수정 버전 상태 및 인스턴스 상태를 사용하여 배포 그룹의 인스턴스에 대한 배포를 다음 순서로 예약합니다.

1. 비정상 인스턴스 상태.
2. 알 수 없음 수정 버전 상태.
3. 이전 수정 버전 상태.
4. 현재 수정 버전 상태.

전체 배포에 성공하면 수정 버전이 업데이트되고 배포 그룹의 상태 값이 최신 배포를 반영하도록 업데이트됩니다.

- 배포에 성공한 모든 현재 인스턴스는 현재 상태로 유지됩니다. 그렇지 않으면 알 수 없음 상태가 됩니다.
- 배포에 성공한 모든 이전 인스턴스 또는 알 수 없음 상태의 모든 인스턴스는 현재 상태가 됩니다. 그렇지 않으면 이전 상태 또는 알 수 없음 상태로 유지됩니다.
- 배포에 성공한 모든 정상 인스턴스는 정상 상태로 유지됩니다. 그렇지 않으면 비정상 상태가 됩니다.

- 배포에 성공한 모든 비정상 인스턴스는 정상 상태가 됩니다. 그렇지 않으면 비정상 상태로 유지됩니다.

전체 배포에 실패하거나 중지된 경우:

- CodeDeploy가 애플리케이션 수정 버전을 배포하려고 시도한 각 인스턴스는 해당 인스턴스에 대한 배포 시도의 성공 또는 실패 여부에 따라 해당 인스턴스 상태가 정상 또는 비정상으로 설정됩니다.
- CodeDeploy가 애플리케이션 수정 버전을 배포하려고 시도하지 않은 각 인스턴스는 현재 인스턴스 상태 값을 유지합니다.
- 배포 그룹의 수정 버전은 동일하게 유지됩니다.

최소 정상 인스턴스 수 정보

필요한 최소 정상 인스턴스 수는 배포 구성의 일부로 정의됩니다.

> **Important**
>
> 블루/그린 배포 시 배포 구성과 최소 정상 호스트 값은 원래 환경의 인스턴스가 아닌 대체 환경의 인스턴스에 적용됩니다. 그러나 원래 환경의 인스턴스가 로드 밸런서에서 등록 취소되는 경우 하나의 원본 인스턴스라도 성공적으로 등록 취소되지 않으면 전체 배포가 실패로 표시됩니다.

CodeDeploy는 일반적으로 최소 정상 호스트 값을 사용하는 세 가지 기본 배포 구성을 제공합니다.

기본 배포 구성 이름	미리 정의된 최소 정상 호스트 값
CodeDeployDefault.OneAtATime	1
CodeDeployDefault.HalfAtATime	50%
CodeDeployDefault.AllAtOnce	0

CodeDeploy에서 배포 구성 작업에서 기본 배포 구성에 대한 자세한 내용을 확인할 수 있습니다.

CodeDeploy에서 사용자 지정 배포 구성을 만들어 고유의 최소 정상 호스트 값을 정의할 수 있습니다. 다음 작업을 사용할 때 이러한 값을 정수 또는 백분율로 정의할 수 있습니다.

- AWS CLI에서 create-deployment-config 명령을 사용할 때 `minimum-healthy-hosts`로.
- CodeDeploy API의 MinimumHealthyHosts 데이터 유형에서 `Value`로.
- CloudFormation 템플릿에서 AWS::CodeDeploy::DeploymentConfig를 사용하는 `MinimumHealthyHosts` 경우와 같습니다.

CodeDeploy를 사용하면 두 가지 주요 목적을 위해 배포에 관한 정상 인스턴스의 최소 개수를 지정할 수 있습니다.

- 전체 배포의 성공 또는 실패 여부를 확인하기 위해. 애플리케이션 수정 버전이 최소 정상 인스턴스 수에 성공적으로 배포된 경우 배포에 성공합니다.
- 배포가 진행되도록 배포 중에 정상 상태를 유지해야 하는 인스턴스 수를 확인하기 위해.

배포 그룹의 최소 정상 인스턴스 수를 인스턴스 수 또는 전체 인스턴스 수의 백분율로 지정할 수 있습니다. 백분율을 지정할 경우 배포 시작 시 CodeDeploy는 이 백분율을 해당 인스턴스 수로 변환하고 소수점 값은 반올림합니다.

CodeDeploy는 배포 프로세스 중에 배포 그룹 인스턴스의 상태를 추적하고 배포를 위해 지정된 최소 수의 정상 인스턴스를 사용하여 배포를 계속할지 여부를 결정합니다. 기본 원칙은 배포 시에 정상 인스턴스 수가 지정한 최소 개수보다 낮아서는 안 된다는 것입니다. 이 규칙의 한 가지 예외는 처음부터 배포 그룹의 정상 인스턴스 수가 지정된 최소 정상 인스턴스 수보다 작은 경우입니다. 이 경우 배포 프로세스는 정상 인스턴스의 수를 더 이상 줄이지 않습니다.

> **Note**
>
> CodeDeploy는 배포 그룹의 모든 인스턴스(현재 중지 상태인 인스턴스 포함)에 배포를 시도합니다. 최소 정상 호스트 계산에서 중지된 인스턴스는 실패한 인스턴스와 동일한 영향을 미칩니다. 중지된 인스턴스가 너무 많아 배포에 실패한 경우 이를 해결하려면 인스턴스를 다시 시작하거나 태그를 변경하여 배포 그룹에서 제외합니다.

CodeDeploy는 배포 그룹의 비정상 인스턴스에 애플리케이션 수정 버전을 배포하려고 시도하여 배포 프로세스를 시작합니다. 배포에 성공할 때마다 CodeDeploy는 인스턴스의 상태를 정상으로 변경하고 배포 그룹의 정상 인스턴스에 추가합니다. 그런 다음 CodeDeploy는 현재 정상 인스턴스 수를 지정된 최소 정상 인스턴스 수와 비교합니다.

- 정상 인스턴스 수가 지정된 최소 정상 인스턴스 수보다 작거나 같으면 CodeDeploy는 배포를 취소하여 더 많은 배포에 따라 정상 인스턴스 수가 줄어들지 않도록 합니다.

- 정상 인스턴스 수가 지정된 최소 정상 인스턴스 수보다 하나 이상 많으면 CodeDeploy는 애플리케이션 수정 버전을 원래 정상 인스턴스 집합에 배포합니다.

정상 인스턴스에 대한 배포에 실패하면 CodeDeploy는 해당 인스턴스의 상태를 비정상으로 변경합니다. 배포가 진행됨에 따라 CodeDeploy는 현재 정상 인스턴스 수를 업데이트하고 지정된 최소 정상 인스턴스 수와 비교합니다. 배포 프로세스상의 어느 지점에서든 정상 인스턴스 수가 지정된 최소 수보다 적아지면 CodeDeploy는 배포를 중지합니다. 이렇게 하면 다음 배포에 실패하여 정상 인스턴스 수가 지정된 최소 수 미만으로 떨어질 가능성을 방지할 수 있습니다.

> **Note**
>
> 지정한 최소 인스턴스 수가 배포 그룹에 있는 전체 인스턴스 수보다 작아야 합니다. 백분율 값을 지정하면 반올림됩니다. 그렇지 않으면 배포가 시작될 때 정상 인스턴스 수가 지정된 최소 정상 인스턴스 수보다 작거나 같으며 CodeDeploy는 전체 배포에 즉시 실패합니다.

또한 CodeDeploy는 지정된 최소 정상 인스턴스 수와 실제 정상 인스턴스 수를 사용하여 애플리케이션 수정 버전을 여러 인스턴스에 배포할지 여부와 방법을 결정합니다. 기본적으로 CodeDeploy는 정상 인스턴스 수가 지정된 최소 정상 인스턴스 수 미만으로 떨어질 위험 없이 가능한 한 많은 인스턴스에 애플리케이션 수정 버전을 배포합니다.

한 번에 배포해야 하는 인스턴스 수를 결정하기 위해 CodeDeploy는 다음 계산을 사용합니다.

```
[total-hosts] - [minimum-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

예제:

- 배포 그룹에 10개의 인스턴스가 있고 최소 정상 인스턴스 수를 9로 설정한 경우 CodeDeploy는 한 번에 하나의 인스턴스에 배포합니다.
- 배포 그룹에 10개의 인스턴스가 있고 최소 정상 인스턴스 수를 3개로 설정한 경우 CodeDeploy는 첫 번째 배치에서 동시에 7개 인스턴스에 배포한 다음 두 번째 배치에서 나머지 3개에 배포합니다.
- 배포 그룹에 10개의 인스턴스가 있고 최소 정상 인스턴스 수를 0으로 설정한 경우 CodeDeploy는 동시에 10개의 인스턴스에 배포됩니다.

예시

다음 예에서는 인스턴스가 10개인 배포 그룹을 가정합니다.

최소 정상 인스턴스: 95%

CodeDeploy는 최소 정상 인스턴스 수를 10개의 인스턴스로 반올림합니다. 이는 정상 인스턴스의 수와 같습니다. 어느 인스턴스에도 수정 버전을 배포하지 않고 전체 배포는 즉시 실패합니다.

최소 정상 인스턴스: 9

CodeDeploy는 한 번에 하나의 인스턴스에 수정 버전을 배포합니다. 어느 하나의 인스턴스에 대한 배포에 실패하면 정상 인스턴스 수가 최소 정상 인스턴스 수와 같아지므로 CodeDeploy의 전체 배포는 즉시 실패합니다. 이 규칙의 예외는 마지막 인스턴스가 실패했는데 전체 배포에 성공하는 경우입니다.

CodeDeploy는 배포에 실패하거나 전체 배포가 완료될 때까지 한 번에 한 인스턴스씩 배포를 계속합니다. 10개의 배포를 모두 성공하면 배포 그룹에는 10개의 정상 인스턴스가 있게 됩니다.

최소 정상 인스턴스: 8

CodeDeploy는 한 번에 두 개의 인스턴스에 수정 버전을 배포합니다. 이러한 배포 중 두 개가 실패하면 CodeDeploy의 전체 배포는 즉시 실패합니다. 이 규칙의 예외는 마지막 인스턴스가 두 번째로 실패한 경우 배포가 계속 성공하는 경우입니다.

최소 정상 인스턴스: 0

CodeDeploy는 수정 버전을 전체 배포 그룹에 한 번에 배포합니다. 전체 배포에 성공하려면 하나 이상의 인스턴스에 대해 배포를 성공해야 합니다. 정상 인스턴스의 수가 0이면 배포에 실패합니다. 이는 전체 배포를 성공으로 표시하려면 최소 정상 인스턴스 값이 0인 경우에도 전체 배포가 완료될 때 적어도 하나의 인스턴스가 정상이어야 한다는 요구 사항 때문입니다.

가용 영역당 최소 정상 인스턴스 수 정보

> **Note**
>
> 이 섹션에서는 인스턴스와 호스트라는 용어를 Amazon EC2 인스턴스를 지칭할 때 혼용하여 사용합니다.

여러 가용 영역의 인스턴스에 배포하는 경우, 선택적으로 영역 구성 기능을 활성화하여 한 번에 하나의 가용 영역에 CodeDeploy를 배포할 수 있습니다.

이 기능을 활성화하면 CodeDeploy는 정상 호스트 수가 '영역당 최소 정상 호스트' 및 '최소 정상 호스트' 값 이상으로 유지되는지 확인합니다. 정상 호스트 수가 두 값 아래로 떨어지면 모든 가용 영역에서 배포가 실패합니다.

한 번에 배포할 호스트 수를 계산하기 위해 CodeDeploy는 '영역당 최소 정상 호스트 수'와 '최소 정상 호스트 수' 값을 모두 사용합니다. CodeDeploy는 [A]와 [B] 중 더 작은 계산값을 사용하며, 여기서 [A]와 [B]는 다음과 같습니다.

```
[A] = [total-hosts] - [min-healthy-hosts] =
        [number-of-hosts-to-deploy-to-at-once]
```

```
[B] = [total-hosts-per-AZ] - [min-healthy-hosts-per-AZ] =
        [number-of-hosts-to-deploy-to-at-once-per-AZ]
```

한 번에 배포할 호스트 수를 결정한 후 CodeDeploy는 해당 수의 호스트에 한 번에 하나의 가용 영역씩 일괄적으로 배포하며, 영역 사이에 일시 중지(또는 '베이크 시간')를 선택할 수 있습니다.

예제

배포가 다음과 같이 구성된 경우:

- [total-hosts]은(는) 200
- [minimum-healthy-hosts]은(는) 160
- [total-hosts-per-AZ]은(는) 100
- [minimum-healthy-hosts-per-AZ]은(는) 50

해당되는 조치

- [A] = 200 - 160 = 40
- [B] = 100 - 50 = 50
- 40은 50 미만

따라서 CodeDeploy는 한 번에 40개의 호스트에 배포됩니다.

이 시나리오에서 배포는 다음과 같이 전개됩니다.

1. CodeDeploy가 첫 번째 가용 영역에 배포됩니다.
   a. CodeDeploy가 처음 40개 호스트에 배포됩니다.
   b. CodeDeploy가 다음 40개 호스트에 배포됩니다.
   c. CodeDeploy가 남은 20개 호스트에 배포됩니다.

이제 첫 번째 가용 영역에 대한 배포가 완료되었습니다.

2. 선택 사항) 첫 번째 영역에 대한 배포가 모니터 지속 시간 또는 첫 번째 영역에 대한 모니터 지속 시간 추가 설정에 정의된 대로 첫 번째 영역에 대한 배포가 '베이크'되는 동안 CodeDeploy가 대기합니다. 문제가 없으면 CodeDeploy가 계속됩니다.
3. CodeDeploy가 두 번째 가용 영역에 배포됩니다.
   a. CodeDeploy가 처음 40개 호스트에 배포됩니다.
   b. CodeDeploy가 다음 40개 호스트에 배포됩니다.
   c. CodeDeploy가 남은 20개 호스트에 배포됩니다.

   이제 두 번째이자 마지막 가용 영역에 대한 배포가 완료되었습니다.

영역 구성 기능 및 가용성 영역당 최소 정상 인스턴스 수를 지정하는 방법에 대해 알아보려면 zonal configuration을 참조하세요.

CodeDeploy에서 배포 구성 작업

배포 구성은 배포 중 CodeDeploy에서 사용하는 규칙과 성공 및 실패 조건 세트입니다. 이러한 규칙과 조건은 EC2/온프레미스 컴퓨팅 플랫폼에 배포하는지, AWS Lambda 컴퓨팅 플랫폼에 배포하는지, Amazon ECS 컴퓨팅 플랫폼에 배포하는지에 따라 다릅니다.

EC2/온프레미스 컴퓨팅 플랫폼에 대한 배포 구성

EC2/온프레미스 컴퓨팅 플랫폼에 배포하는 경우, 배포 구성은 '최소 정상 호스트' 값과 선택적 '영역당 최소 정상 호스트' 값을 사용하여 배포 중 언제든지 사용할 수 있어야 하는 인스턴스의 수 또는 비율을 지정합니다.

에서 제공하는 세 가지 사전 정의된 배포 구성 중 하나를 사용하거나 사용자 지정 배포 구성을 AWS 생성할 수 있습니다. 사용자 지정 배포 구성 만들기에 대한 자세한 내용은 Create a Deployment Configuration 단원을 참조하세요. 배포 구성을 지정하지 않으면 CodeDeplo에서는 CodeDeployDefault.OneAtATime 배포 구성을 사용합니다.

CodeDeplo가 배포 중 인스턴스 상태를 모니터링 및 평가하는 방법에 대한 자세한 내용은 Instance Health 단원을 참조하세요. AWS 계정에 이미 등록된 배포 구성 목록을 보려면 섹션을 참조하세요View Deployment Configuration Details.

EC2/온프레미스 컴퓨팅 플랫폼에 대한 미리 정의된 배포 구성

다음은 미리 정의된 배포 구성이 나열된 표입니다.

> **Note**
>
> 이 zonal configuration 기능(가용 영역당 정상 호스트 수를 지정할 수 있는 기능)을 지원하는 사전 정의된 배포 구성은 없습니다. 이 기능을 사용하려면 배포 구성을 직접 생성해야 합니다.

배포 구성	설명
CodeDeployDefault.AllAtOnce	인 플레이스(in-place) 배포: 한 번에 가급적 많은 수의 인스턴스에 애플리케이션 개정을 배포하기 위한 시도입니다. 하나 이상의 인스턴스에 애플리케이션 개정이 배포되면

배포 구성	설명
	전체 배포 상태가 성공으로 표시됩니다. 어떤 인스턴스에도 애플리케이션 개정이 배포되지 않으면 전체 배포 상태가 실패로 표시됩니다. 아홉 개의 인스턴스가 있는 예제에서 CodeDeployDefault.AllAtOnce는 한 번에 아홉 개의 인스턴스 모두에 배포를 시도합니다. 인스턴스 하나에라도 배포에 성공하면 전체 배포가 성공합니다. 아홉 개 인스턴스 모두에 대한 배포에 실패하는 경우에만 실패합니다. 블루/그린 배포: • 대체 환경으로 배포: 인 플레이스(in-place) 배포의 CodeDeployDefault.AllAtOnce와 동일한 배포 규칙을 따릅니다. • 트래픽 라우팅: 한 번에 대체 환경의 모든 인스턴스로 트래픽을 라우팅합니다. 트래픽이 하나 이상의 인스턴스로 성공적으로 라우팅되면 성공합니다. 모든 인스턴스로 다시 라우팅하지 못하며 실패합니다.

배포 구성	설명
CodeDeployDefault.HalfAtATime	인 플레이스(in-place) 배포: 최대 절반의 인스턴스(분수는 반내림함)에 한번에 배포합니다. 애플리케이션 개정이 인스턴스의 절반 이상(분수는 반올림함)에 배포되면 전체 배포가 성공합니다. 그렇지 않으면 배포가 실패합니다. 아홉 개의 인스턴스가 있는 예제에서는 한 번에 최대 네 개의 인스턴스에 배포합니다. 다섯 개 이상의 인스턴스에 대한 배포에 성공하면 전체 배포가 성공합니다. 그렇지 않으면 배포가 실패합니다. **Note** 여러 Auto Scaling 그룹의 인스턴스에 배포하는 경우 CodeDeploy는 속한 Auto Scaling 그룹과 관계없이 한 번에 최대 절반의 인스턴스에 배포합니다. 예를 들어 Auto Scaling 그룹이 두 개(ASG1, ASG2) 있고 각각 10개의 인스턴스가 있다고 가정해 보겠습니다. 이 시나리오에서 CodeDeploy는 ASG1에 있는 10개의 인스턴스에만 배포하고 성공했다고 간주할 수 있습니다. 인스턴스의 절반 이상에 배포했기 때문입니다. 블루/그린 배포: • 대체 환경으로 배포: 인 플레이스(in-place) 배포의 CodeDeployDefault.HalfAtATime과 동일한 배포 규칙을 따릅니다. • 트래픽 라우팅: 한 번에 대체 환경의 인스턴스 중 최대 절반으로 트래픽을 라우팅합니다. 인스턴스 중 절반 이상으로의 다시 라우팅에 성

배포 구성	설명
	공하는 경우 성공합니다. 그렇지 않으면 가 실패합니다.

배포 구성	설명
CodeDeployDefault.OneAtATime	인 플레이스(in-place) 배포: 한 번에 한 인스턴스에만 애플리케이션 개정을 배포합니다. 인스턴스가 두 개 이상 포함된 배포 그룹의 경우: • 애플리케이션 개정이 모든 인스턴스에 배포되면 전체 배포가 성공입니다. 이 규칙의 예외는 마지막 인스턴스에 대한 배포에 실패했는데 전체 배포에 성공하는 경우입니다. 이는 CodeDeploy가 CodeDeployDefault.OneAtATime 구성을 사용하여 한 번에 한 인스턴스만 오프라인 상태가 되도록 허용하기 때문입니다. • 애플리케이션 개정이 마지막 인스턴스를 제외한 모든 인스턴스에 배포되지 못하면 즉시 전체 배포에 실패합니다. • 아홉 개의 인스턴스가 있는 예제에서는 한 번에 하나의 인스턴스에 배포합니다. 처음 여덟 개 인스턴스에 대한 배포에 성공하면 전체 배포가 성공합니다. 처음 여덟 개 인스턴스 중 하나에 대한 배포에 실패하면 전체 배포가 실패합니다. 인스턴스가 하나뿐인 배포 그룹의 경우, 한 인스턴스에 대한 배포에 성공해야만 전체 배포에 성공합니다. 블루/그린 배포: • 대체 환경으로 배포: 인 플레이스(in-place) 배포의 CodeDeployDefault.OneAtATime과 동일한 배포 규칙을 따릅니다.

배포 구성	설명
	• 트래픽 라우팅: 한 번에 대체 환경의 인스턴스 하나로 트래픽을 라우팅합니다. 트래픽이 모든 대체 인스턴스로 성공적으로 라우팅되면 성공합니다. 맨 처음으로 다시 라우팅하지 못하면 실패합니다. 이 규칙의 예외는 마지막 인스턴스 등록에 실패했는데 전체 배포에 성공하는 경우입니다.

Amazon ECS 컴퓨팅 플랫폼에 대한 배포 구성

Amazon ECS 컴퓨팅 플랫폼에 배포할 때 배포 구성은 트래픽이 업데이트된 Amazon ECS 작업 세트로 이동되는 방법을 지정합니다. 카나리(Canary), 리니어(Linear) 또는 한 번에 모두(All-at-once) 배포 구성을 사용하여 트래픽을 이동할 수 있습니다. 자세한 내용은 배포 구성 단원을 참조하십시오.

Canary 또는 선형 배포 구성을 직접 사용자 지정하여 만들 수도 있습니다. 자세한 내용은 Create a Deployment Configuration 단원을 참조하십시오.

Amazon ECS 컴퓨팅 플랫폼에 대해 미리 정의된 배포 구성

다음 표에는 Amazon ECS 배포에 사용할 수 있는 미리 정의된 구성이 나열되어 있습니다.

> **Note**
>
> Network Load Balancer를 사용하는 경우 미리 정의된 배포 구성인 `CodeDeployDefault.ECSAllAtOnce`만이 지원됩니다.

배포 구성	설명
CodeDeployDefault.ECSLinear10PercentEvery1Minutes	모든 트래픽이 전환될 때까지 트래픽의 10%가 매분마다 전환됩니다.
CodeDeployDefault.ECSLinear10PercentEvery3Minutes	모든 트래픽이 이동될 때까지 트래픽의 10%가 3분마다 이동됩니다.

배포 구성	설명
CodeDeployDefault.ECSCanary10Percent5Minutes	첫 번째 증분에 트래픽의 10%가 전환됩니다. 나머지 90%는 5분 이후 배포됩니다.
CodeDeployDefault.ECSCanary10Percent15Minutes	첫 번째 증분에 트래픽의 10%가 전환됩니다. 나머지 90%는 15분 후에 배포됩니다.
CodeDeployDefault.ECSAllAtOnce	모든 트래픽을 업데이트된 Amazon ECS 컨테이너로 한 번에 이동합니다.

CloudFormation 블루/그린 배포를 위한 배포 구성(Amazon ECS)

CloudFormation 블루/그린 배포를 통해 Amazon ECS 컴퓨팅 플랫폼에 배포하는 경우 배포 구성은 트래픽이 업데이트된 Amazon ECS 컨테이너로 이동하는 방법을 지정합니다. 카나리(Canary), 리니어(Linear) 또는 한 번에 모두(All-at-once) 배포 구성을 사용하여 트래픽을 이동할 수 있습니다. 자세한 내용은 배포 구성 단원을 참조하십시오.

CloudFormation 블루/그린 배포에서는 자체 사용자 지정 canary 또는 선형 배포 구성을 생성할 수 없습니다. 를 사용하여 Amazon ECS 블루/그린 배포 CloudFormation 를 관리하는 방법에 대한 step-by-step 지침은 CloudFormation 사용 설명서의 를 사용하여 CodeDeploy를 통해 ECS 블루/그린 배포 자동화 CloudFormation를 참조하세요.

> **Note**
>
> 유럽(밀라노), 아프리카(케이프타운) 및 아시아 태평양(오사카) 리전에서는를 사용하여 Amazon ECS 블루/그린 배포를 관리할 수 CloudFormation 없습니다.

AWS Lambda 컴퓨팅 플랫폼의 배포 구성

AWS Lambda 컴퓨팅 플랫폼에 배포할 때 배포 구성은 트래픽이 애플리케이션의 새 Lambda 함수 버전으로 이동하는 방법을 지정합니다. 카나리(Canary), 리니어(Linear) 또는 한 번에 모두(All-at-once) 배포 구성을 사용하여 트래픽을 이동할 수 있습니다. 자세한 내용은 배포 구성 단원을 참조하십시오.

Canary 또는 선형 배포 구성을 직접 사용자 지정하여 만들 수도 있습니다. 자세한 내용은 Create a Deployment Configuration 단원을 참조하십시오.

AWS Lambda 컴퓨팅 플랫폼에 대해 미리 정의된 배포 구성

다음 표에는 AWS Lambda 배포에 사용할 수 있는 미리 정의된 구성이 나열되어 있습니다.

배포 구성	설명
CodeDeployDefault.LambdaCanary10Percent5Minutes	첫 번째 증분에 트래픽의 10%가 전환됩니다. 나머지 90%는 5분 이후 배포됩니다.
CodeDeployDefault.LambdaCanary10Percent10Minutes	첫 번째 증분에 트래픽의 10%가 전환됩니다. 나머지 90%는 10분 이후 배포됩니다.
CodeDeployDefault.LambdaCanary10Percent15Minutes	첫 번째 증분에 트래픽의 10%가 전환됩니다. 나머지 90%는 15분 후에 배포됩니다.
CodeDeployDefault.LambdaCanary10Percent30Minutes	첫 번째 증분에 트래픽의 10%가 전환됩니다. 나머지 90%는 30분 이후 배포됩니다.
CodeDeployDefault.LambdaLinear10PercentEvery1Minute	모든 트래픽이 전환될 때까지 트래픽의 10%가 매분마다 전환됩니다.
CodeDeployDefault.LambdaLinear10PercentEvery2Minutes	모든 트래픽이 이동될 때까지 트래픽의 10%가 2분마다 이동됩니다.
CodeDeployDefault.LambdaLinear10PercentEvery3Minutes	모든 트래픽이 이동될 때까지 트래픽의 10%가 3분마다 이동됩니다.
CodeDeployDefault.LambdaLinear10PercentEvery10Minutes	모든 트래픽이 전환될 때까지 트래픽의 10%가 10분마다 전환됩니다.
CodeDeployDefault.LambdaAllAtOnce	업데이트된 Lambda 함수로 모든 트래픽을 한번에 전환합니다.

주제

- Create a Deployment Configuration
- View Deployment Configuration Details
- Delete a Deployment Configuration

CodeDeploy에서 배포 구성 만들기

CodeDeploy와 함께 제공되는 기본 배포 구성 중 하나를 사용하고 싶지 않은 경우 다음 지침을 사용하여 직접 만들 수 있습니다.

CodeDeploy 콘솔, AWS CLI CodeDeploy APIs 또는 CloudFormation 템플릿을 사용하여 사용자 지정 배포 구성을 생성할 수 있습니다.

CloudFormation 템플릿을 사용하여 배포 구성을 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요CloudFormation CodeDeploy 참조용 템플릿.

주제

- 배포 구성 생성(콘솔)
- CodeDeploy에서 배포 구성 생성(AWS CLI)

배포 구성 생성(콘솔)

AWS 콘솔을 사용하여 배포 구성을 만들려면 다음 지침을 따르세요.

콘솔을 사용하여 CodeDeploy에서 배포 구성을 생성하려면 다음을 수행하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포 구성을 선택합니다.

   기본 제공 배포 구성 목록이 나타납니다.
3. 배포 구성 만들기를 선택합니다.
4. 배포 구성 이름에 배포 구성 이름을 입력합니다. 예를 들어 **`my-deployment-config`**입니다.
5. 컴퓨팅 플랫폼에서 다음 중 하나를 선택합니다.

   - EC2/온프레미스
   - AWS Lambda

- Amazon ECS

6. 다음 중 하나를 수행하세요.

   - EC2/온프레미스를 선택한 경우:

     1. 최소 정상 호스트에서 배포 중 언제든지 사용 가능한 상태로 유지되어야 하는 인스턴스의 수 또는 비율을 지정합니다. CodeDeplo가 배포 중 인스턴스 상태를 모니터링 및 평가하는 방법에 대한 자세한 내용은 Instance Health 단원을 참조하세요.
     2. (선택 사항) 영역 구성에서 영역 구성 활성화를 선택하여 CodeDeploy가 AWS 리전 내에서 한 번에 하나의 가용 영역에 애플리케이션을 배포하도록 합니다. 한 번에 하나의 가용 영역에 배포하면 배포의 성능과 실행 가능성에 대한 신뢰가 높아짐에 따라 점차 더 많은 대상에게 배포를 노출할 수 있습니다. 역 구성을 활성화하지 않으면 CodeDeploy는 리전 전체에서 무작위로 선택한 호스트에 애플리케이션을 배포합니다.

        영역 구성 기능을 활성화하는 경우 다음 사항에 유의하세요.

        - 영역 구성 기능은 Amazon EC2 인스턴스에 대한 현재 위치 배포에서만 지원됩니다. (블루/그린 배포 및 온프레미스 인스턴스는 지원되지 않음) 현재 위치 배포에 대한 자세한 내용은 배포 유형 단원을 참조하세요.
        - 영역 구성 기능은 미리 정의된 배포 구성에서는 지원되지 않습니다. 영역 구성을 사용하려면 여기에 설명된 대로 사용자 지정 배포 구성을 만들어야 합니다.
        - CodeDeploy가 배포를 롤백해야 하는 경우 CodeDeploy는 임의의 호스트에서 롤백 작업을 수행합니다. (예상대로 CodeDeploy는 한 번에 한 영역씩 롤백하지 않음) 이 롤백 동작은 성능상의 이유로 선택되었습니다. 롤백에 대한 자세한 내용은 CodeDeploy를 사용하여 재배포 및 배포 롤백을 참조하세요.

     3. 영역 구성 활성화 확인란을 선택한 경우 선택적으로 다음 옵션을 지정합니다.

        - (선택 사항) 모니터 지속 시간에서 가용 영역에 대한 배포를 완료한 후 CodeDeploy가 대기해야 하는 기간(초)을 지정합니다. CodeDeploy는 다음 가용 영역으로의 배포를 시작하기 전에 이 시간 동안 대기합니다. 다음 가용 영역에 배포하기 전에 한 가용 영역에서 배포가 스스로를 증명(또는 '베이크')할 시간을 주기 위해 모니터 지속 시간을 추가하는 것을 고려하세요. 모니터 지속 시간을 지정하지 않으면 CodeDeploy는 즉시 다음 가용 영역으로 배포를 시작합니다. 모니터 지속 시간 설정의 작동 방식에 대한 자세한 내용은 가용 영역당 최소 정상 인스턴스 수 정보를 참조하세요.
        - (선택 사항) 첫 번째 영역에 대한 모니터 지속 시간 추가를 선택하여 첫 번째 가용성 영역에만 적용되는 모니터 지속 시간을 설정합니다. 첫 번째 가용 영역에 대해 추가 베이크 시간을 허용하려는 경우 이 옵션을 설정할 수 있습니다. 첫 번째 영역 모니터 지속 시간 추가에

값을 지정하지 않으면 CodeDeploy는 첫 번째 가용성 영역의 모니터 지속 시간 값을 사용합니다.

- (선택 사항) 영역당 최소 정상 호스트에서 배포 중 가용 영역별로 사용 가능한 상태로 유지되어야 하는 인스턴스의 수 또는 백분율을 지정합니다. 백분율을 지정하려면 FLEET_PERCENT를 선택하고, 숫자를 지정하려면 HOST_COUNT를 선택합니다. 이 필드는 최소 정상 호스트 필드와 함께 작동합니다. 자세한 내용은 가용 영역당 최소 정상 인스턴스 수 정보 단원을 참조하십시오.

  영역당 최소 정상 호스트에서 값을 지정하지 않으면 CodeDeploy는 기본값인 0%를 사용합니다.

- AWS Lambda 또는 Amazon ECS를 선택한 경우:
  1. 유형에서 선형 또는 캐너리를 선택합니다.
  2. 단계 및 간격 필드에서 다음 중 하나를 수행하세요.
     - 단계에 캐너리를 선택한 경우, 이동할 트래픽의 백분율(1에서 99 사이)을 입력합니다. 이 값은 첫 번째 증분에서 이동되는 트래픽의 백분율입니다. 나머지 트래픽은 선택한 간격 후에 두 번째 증분으로 이동합니다.

       간격에 첫 번째와 두 번째 트래픽 이동 사이의 시간(분)을 입력합니다.
     - 단계에 선형을 선택한 경우, 이동할 트래픽의 백분율(1에서 99 사이)을 입력합니다. 이 값은 각 간격이 시작될 때 이동되는 트래픽의 비율입니다.

       간격에 각 증분 이동 사이의 시간(분)을 입력합니다.

7. 배포 구성 만들기를 선택합니다.

   이제 배포 그룹과 연결할 수 있는 배포 구성이 생겼습니다.

CodeDeploy에서 배포 구성 생성(AWS CLI)

AWS CLI 를 사용하여 배포 구성을 생성하려면 create-deployment-config 명령을 호출합니다.

다음 예에서는 배포 중 정상 상태를 유지하기 위해 대상 인스턴스의 75%를 필요로 하는 ThreeQuartersHealthy라는 EC2/온프레미스 배포 구성을 생성합니다.

```
aws deploy create-deployment-config --deployment-config-name ThreeQuartersHealthy --
minimum-healthy-hosts type=FLEET_PERCENT,value=75
```

다음 예에서는 배포당 총 300개의 대상 인스턴스가 정상 상태를 유지해야 하고 가용 영역당 50개의 대상 인스턴스가 정상 상태를 유지해야 하는 `300Total50PerAZ`라는 EC2/온프레미스 배포 구성을 생성합니다. 또한 모니터 지속 시간을 1시간으로 설정합니다.

```
aws deploy create-deployment-config --deployment-config-name 300Total50PerAZ
 --minimum-healthy-hosts type=HOST_COUNT,value=300 --zonal-config
 '{"monitorDurationInSeconds":3600,"minimumHealthyHostsPerZone":
{"type":"HOST_COUNT","value":50}}'
```

다음 예제에서는 라는 AWS Lambda 배포 구성을 생성합니다`Canary25Percent45Minutes`. 이 구성은 Canary 트래픽 이동을 사용하여 첫 증분에서 트래픽의 25퍼센트를 이동합니다. 나머지 75%는 45분 후에 이동됩니다.

```
aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
 --traffic-routing-config
 "type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --
compute-platform Lambda
```

다음 예에서는 `Canary25Percent45Minutes`라는 Amazon ECS 배포 구성을 생성합니다. 이 구성은 Canary 트래픽 이동을 사용하여 첫 증분에서 트래픽의 25퍼센트를 이동합니다. 나머지 75%는 45분 후에 이동됩니다.

```
aws deploy create-deployment-config --deployment-config-name Canary25Percent45Minutes
 --traffic-routing-config
 "type="TimeBasedCanary",timeBasedCanary={canaryPercentage=25,canaryInterval=45}" --
compute-platform ECS
```

CodeDeploy의 배포 구성 세부 정보 보기

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy APIs를 사용하여 AWS 계정과 연결된 배포 구성에 대한 세부 정보를 볼 수 있습니다. 미리 정의된 CodeDeploy 배포 구성에 대한 설명은 EC2/온프레미스 컴퓨팅 플랫폼에 대한 미리 정의된 배포 구성 단원을 참조하세요.

주제

- 배포 구성 세부 정보 보기(콘솔)

- 배포 구성 보기(CLI)

배포 구성 세부 정보 보기(콘솔)

CodeDeploy 콘솔을 사용하여 배포 구성 이름 목록을 보려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 배포 구성을 선택합니다.

   여기에서 각 배포 구성에 대한 배포 구성 이름과 기준을 볼 수 있습니다.

   > **Note**
   >
   > 항목이 표시되지 않으면 올바른 리전이 선택되어 있는지 확인합니다. 탐색 모음의 리전 선택기에서 AWS 일반 참조의 리전 및 엔드포인트에 나열된 리전 중 하나를 선택합니다. CodeDeploy는 이러한 리전에서만 지원됩니다.

배포 구성 보기(CLI)

AWS CLI 를 사용하여 배포 구성 세부 정보를 보려면 `get-deployment-config` 명령 또는 `list-deployment-configs` 명령을 호출합니다.

단일 배포 구성에 대한 세부 정보를 보려면 get-deployment-config 명령을 호출하여 고유한 배포 구성 이름을 지정합니다.

여러 배포 구성에 대한 세부 정보를 보려면 list-deployments 명령을 호출합니다.

CodeDeploy에서 배포 구성 삭제

AWS CLI 또는 CodeDeploy APIs를 사용하여 AWS 계정과 연결된 사용자 지정 배포 구성을 삭제할 수 있습니다. `CodeDeployDefault.AllAtOnce`, `CodeDeployDefault.HalfAtATime` 및 `CodeDeployDefault.OneAtATime`과 같은 기본 제공 배포 구성은 삭제할 수 없습니다.

> **Warning**
>
> 사용 중인 사용자 지정 배포 구성은 삭제할 수 없습니다. 사용되지 않는 사용자 지정 배포 구성을 삭제하면 더 이상 새 배포 및 새 배포 그룹과 연결할 수 없습니다. 이 작업은 실행을 취소할 수 없습니다.

AWS CLI 를 사용하여 배포 구성을 삭제하려면 배포 구성 이름을 지정하여 delete-deployment-config 명령을 호출합니다. 배포 구성 이름 목록을 보려면 list-deployment-configs 명령을 호출하세요.

다음 예에서는 ThreeQuartershealthy라는 이름의 배포 구성을 삭제합니다.

```
aws deploy delete-deployment-config --deployment-config-name ThreeQuartersHealthy
```

CodeDeploy의 애플리케이션 작업

인스턴스를 구성한 후 개정을 배포하려면 먼저 CodeDeploy에서 애플리케이션을 만들어야 합니다. 애플리케이션은 CodeDeploy에서 배포 중에 올바른 개정, 배포 구성 및 배포 그룹이 참조되는지 확인하기 위해 사용하는 이름 또는 컨테이너입니다.

다음 표의 정보를 사용하여 다음 단계를 살펴보세요.

컴퓨팅 플랫폼	시나리오	다음 단계에 대한 정보
EC2/온프레미스	아직 인스턴스를 만들지 않았습니다.	CodeDeploy용 인스턴스 작업을(를) 확인한 후 이 페이지로 돌아옵니다.
EC2/온프레미스	인스턴스를 만들었지만 태그 지정을 완료하지 않았습니다.	Tagging Instances for Deployments을(를) 확인한 후 이 페이지로 돌아옵니다.
EC2/온프레미스,AWS Lambda, 및Amazon ECS	아직 애플리케이션을 만들지 않았습니다.	CodeDeploy를 사용하여 애플리케이션 생성을(를) 참조하세요.
EC2/온프레미스,AWS Lambda, 및Amazon ECS	이미 애플리케이션을 만들었지만 배포 그룹을 만들지 않았습니다.	CodeDeploy에서 배포 그룹 만들기을(를) 참조하세요.
EC2/온프레미스,AWS Lambda, 및Amazon ECS	이미 애플리케이션 및 배포 그룹을 만들었지만 애플리케이션 개정을 만들지 않았습니다.	CodeDeploy의 애플리케이션 개정 작업을(를) 참조하세요.
EC2/온프레미스,AWS Lambda, 및Amazon ECS	이미 애플리케이션과 배포 그룹을 만들었으며 애플리케이션 개정을 이미 업로드했습니다. 배포 준비가 되었습니다.	CodeDeploy에서 배포 만들기을(를) 참조하세요.

주제

- CodeDeploy를 사용하여 애플리케이션 생성

- CodeDeploy로 애플리케이션 세부 정보 보기
- 알림 규칙 생성
- CodeDeploy 애플리케이션 이름 바꾸기
- CodeDCodeDeploy에서 애플리케이션을 삭제합니다.

CodeDeploy를 사용하여 애플리케이션 생성

애플리케이션은 CodeDeploy에서 사용하는 단순한 이름 또는 컨테이너로, 배포 중에 올바른 개정, 배포 구성 및 배포 그룹이 참조되도록 합니다. CodeDeploy 콘솔, AWS CLI, CodeDeploy APIs 또는 CloudFormation 템플릿을 사용하여 애플리케이션을 생성할 수 있습니다.

코드 또는 애플리케이션 개정은 배포라는 프로세스를 통해 인스턴스에 설치됩니다. CodeDeploy는 두 종류의 배포를 지원합니다.

- 현재 위치 배포: 배포 그룹의 각 인스턴스에 있는 애플리케이션이 중지되고 최신 애플리케이션 개정 버전이 설치되며 애플리케이션의 새 버전이 시작되고 유효성이 검사됩니다. 로드 밸런서를 사용하면 배포가 진행될 때 각 인스턴스를 등록 취소한 후 배포가 완료된 후 서비스로 복원할 수 있습니다. EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포만 인 플레이스 배포를 사용할 수 있습니다. 현재 위치 배포에 대한 자세한 내용은 인 플레이스 배포 개요 단원을 참조하세요.
- Blue/Green 배포: 배포 동작은 사용하는 컴퓨팅 플랫폼에 따라 다릅니다.
  - EC2/온프레미스 컴퓨팅 플랫폼에서의 블루/그린 배포: 배포 그룹(원래 환경)의 인스턴스가 다음 단계를 거쳐 인스턴스의 다른 집합(대체 환경)으로 대체됩니다.
    - 인스턴스는 대체 환경을 위해 프로비저닝됩니다.
    - 최신 애플리케이션 수정은 대체 인스턴스에 설치됩니다.
    - 애플리케이션 테스트 및 시스템 검증과 같은 활동에 선택적 대기 시간이 발생합니다.
    - 대체 환경의 인스턴스가 하나 이상의 Elastic Load Balancing 로드 밸런서에 등록되고 트래픽이 이러한 인스턴스로 라우팅됩니다. 원래 환경의 인스턴스는 등록이 취소되고 종료되거나 다른 용도로 계속 실행될 수 있습니다.

    > **Note**
    >
    > EC2/온프레미스 컴퓨팅 플랫폼을 사용할 경우 블루/그린 배포는 Amazon EC2 인스턴스에서만 작동합니다.

  - AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼의 블루/그린: 트래픽은 canary, 선형 또는 all-at-once 배포 구성에 따라 증분으로 이동합니다.

- 를 통한 블루/그린 배포 CloudFormation: CloudFormation 스택 업데이트의 일부로 트래픽이 현재 리소스에서 업데이트된 리소스로 이동합니다. 현재는 ECS 블루/그린 배포만 지원됩니다.

Blue/Green 배포에 대한 자세한 내용은 블루/그린 배포 개요 단원을 참조하세요.

CodeDeploy 콘솔을 사용하여 애플리케이션을 만드는 경우 동시에 이 애플리케이션의 첫 번째 배포 그룹을 구성합니다. AWS CLI 를 사용하여 애플리케이션을 생성할 때 별도의 단계에서 첫 번째 배포 그룹을 생성합니다.

AWS 계정에 이미 등록된 애플리케이션 목록을 보려면 섹션을 참조하세요CodeDeploy로 애플리케이션 세부 정보 보기. CloudFormation 템플릿을 사용하여 애플리케이션을 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요CloudFormation CodeDeploy 참조용 템플릿.

두 배포 유형 모두 모든 대상에 적용되는 것은 아닙니다. 다음 표에는 세 가지 유형의 배포 대상에 대한 배포와 함께 작동하는 배포 유형이 나열되어 있습니다.

배포 대상	현재 위치	블루/그린
Amazon EC2	예	예
온프레미스	예	아니요
서버리스 AWS Lambda 함수	아니요	예
Amazon ECS 애플리케이션	아니요	예

주제

- 현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔)
- Blue/Green 배포를 위한 애플리케이션 생성(콘솔)
- Amazon ECS 서비스 배포를 위한 애플리케이션 생성 (콘솔)
- AWS Lambda 함수 배포를 위한 애플리케이션 생성(콘솔)
- 애플리케이션(CLI) 생성

현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔)

CodeDeploy 콘솔을 사용하여 현재 위치 배포를 위한 애플리케이션을 생성하려면

> **Warning**
>
> 다음과 같은 경우 아래 단계를 수행하지 마세요.
>
> - CodeDeploy 배포에서 사용할 인스턴스를 아직 준비하지 않은 경우. 인스턴스를 설정하려면 CodeDeploy용 인스턴스 작업 섹션의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 사용자 지정 배포 구성을 사용하는 애플리케이션을 만들고 싶지만 아직 배포 구성을 만들지 못한 경우. Create a Deployment Configuration의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 필요한 최소한의 신뢰 및 권한으로 CodeDeploy를 신뢰하는 서비스 역할이 없는 경우. 필요한 권한이 있는 서비스 역할을 만들고 구성하려면 2단계: CodeDeploy에 대한 서비스 역할 생성의 지침을 수행한 다음 이 주제의 단계로 돌아갑니다.
> - 현재 위치 배포에 대해 Elastic Load Balancer에서 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer를 선택하려고 하지만 아직 생성하지 않았습니다.

CodeDeploy 콘솔을 사용하여 현재 위치 배포를 위한 애플리케이션을 생성하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 시작하기를 선택합니다.
3. 애플리케이션 생성을 선택합니다.
4. 애플리케이션 이름에 애플리케이션의 이름을 입력합니다.
5. 컴퓨팅 플랫폼에서 EC2/온프레미스를 선택합니다.
6. 애플리케이션 생성을 선택합니다.
7. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다.
8. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.

> **Note**
>
> (배포 그룹 이름, 태그, Amazon EC2 Auto Scaling 그룹 이름 또는 둘 다 및 배포 구성을 비롯하여) 다른 배포 그룹에서 사용되는 것과 동일한 설정을 사용하려면 이 페이지에서 지정합니다. 새 배포 그룹과 기존 배포 그룹의 이름이 같더라도 각 배포 그룹은 개별 애플리케이션과 연결되므로 CodeDeploy에서는 이러한 배포 그룹을 개별 배포 그룹으로 취급합니다.

9. 서비스 역할에서 대상 인스턴스에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할을 선택합니다.
10. 배포 유형에서 In-place(현재 위치)를 선택합니다.
11. 환경 구성에서 다음 중 하나를 선택합니다.

    a. Amazon EC2 Auto Scaling 그룹: 애플리케이션 개정을 배포할 Amazon EC2 Auto Scaling 그룹의 이름을 입력하거나 선택합니다. 새 Amazon EC2 인스턴스가 Amazon EC2 Auto Scaling 그룹의 일부로 시작되면 CodeDeploy에서는 새 인스턴스로 개정을 자동으로 배포할 수 있습니다. 배포 그룹당 최대 10개의 Amazon EC2 Auto Scaling 그룹을 추가할 수 있습니다.

    b. Amazon EC2 인스턴스 또는 온프레미스 인스턴스: 키 및 값 필드에서 인스턴스에 태그를 지정하는 데 사용하는 키-값 페어의 값을 입력합니다. 한 태그 그룹에서 최대 10개의 키-값 페어에 태그를 지정할 수 있습니다.

        i. 값 필드에서 와일드카드를 사용해 특정 패턴으로 태그가 지정된 모든 인스턴스를 식별할 수 있습니다(예: Amazon EC2 인스턴스, 코스트 센터 및 그룹 이름 등). 예를 들어, 키필드에서 이름을 선택하고 값 필드에서 **`GRP-*a`**를 입력하면 CodeDeploy가 해당 패턴에 맞는 모든 인스턴스(예: **`GRP-1a`**, **`GRP-2a`**, 및 **`GRP-XYZ-a`**)를 식별합니다.

        ii. 값 필드는 대/소문자를 구분합니다.

        iii. 목록에서 키-값 페어를 제거하려면 태그 제거를 선택합니다.

    CodeDeploy에서는 지정된 각 키-값 페어 또는 Amazon EC2 Auto Scaling 그룹 이름과 일치하는 인스턴스를 찾기 때문에 일치하는 인스턴스 개수가 표시됩니다. 인스턴스에 대한 자세한 내용을 확인하려면 이 숫자를 선택하세요.

    배포된 인스턴스의 기준을 구체화하려면 태그 그룹 추가를 선택하여 태그 그룹을 생성합니다. 키-값 페어가 각각 최대 10개인 태그 그룹을 최대 세 개까지 만들 수 있습니다. 배포 그룹에서 여러 태그 그룹을 사용하는 경우 모든 태그 그룹으로 식별되는 인스턴스만 배포 그룹에

포함됩니다. 즉 인스턴스는 배포 그룹에 포함될 각 그룹의 태그 중 적어도 하나와 일치해야 합니다.

태그 그룹을 사용하여 배포 그룹을 세분화하는 방법에 대한 자세한 내용은 Tagging Instances for Deployments 단원을 참조하세요.

12. 배포 설정에서, 애플리케이션이 인스턴스에 배포되는 속도를 제어하는 배포 구성을 선택합니다(예: 한 번에 하나씩 또는 한 번에 모두). 배포 구성에 대한 자세한 내용은 CodeDeploy에서 배포 구성 작업 단원을 참조하세요.
13. (선택 사항) 로드 밸런서에서 로드 밸런싱 활성화를 선택한 후, 목록에서 CodeDeploy 배포 프로세스 중 인스턴스로 가는 트래픽을 관리할 Classic Load Balancer, Application Load Balancer 대상 그룹 및 Network Load Balancer 대상 그룹을 선택합니다. 최대 10개의 Classic Load Balancer 및 10개의 대상 그룹으로 총 20개의 항목을 선택할 수 있습니다. 배포하려는 Amazon EC2 인스턴스가 선택한 로드 밸런서(Classic Load Balancer) 또는 대상 그룹(Application Load Balancer 및 Network Load Balancer)에 등록되어 있는지 확인합니다.

    배포 중에는 선택한 로드 밸런서 및 대상 그룹에서 원본 인스턴스의 등록이 취소되어 배포 중에 트래픽이 이러한 인스턴스로 라우팅되는 것이 방지됩니다. 배포가 완료되면 각 인스턴스가 선택한 모든 Classic Load Balancer 및 대상 그룹에 다시 등록됩니다.

    CodeDeploy 배포에 사용할 수 있는 로드 밸런서에 대한 자세한 내용은 Integrating CodeDeploy with Elastic Load Balancing 단원을 참조하세요.
14. (선택 사항) 고급을 확장하고 배포에 포함하려는 모든 옵션(예: Amazon SNS 알림 트리거, Amazon CloudWatch 경보 또는 자동 롤백)을 구성합니다.

    자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하십시오.
15. [Create deployment group]을 선택합니다.

다음 단계는 애플리케이션 및 배포 그룹에 배포할 개정을 준비하는 것입니다. 지침은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.

Blue/Green 배포를 위한 애플리케이션 생성(콘솔)

CodeDeploy 콘솔을 사용하여 Blue/Green 배포에 필요한 애플리케이션을 생성하려면

> **Note**
>
> AWS Lambda 컴퓨팅 플랫폼에 대한 배포는 항상 블루/그린 배포입니다. 배포 유형 옵션은 지정하지 않습니다.

> **Warning**
>
> 다음과 같은 경우 아래 단계를 수행하지 마세요.
>
> - Blue/Green 배포 프로세스 중 대체하려는 인스턴스에 CodeDeploy 에이전트가 설치되어 있지 않은 경우. 인스턴스를 설정하려면 CodeDeploy용 인스턴스 작업 섹션의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 사용자 지정 배포 구성을 사용하는 애플리케이션을 만들고 싶지만 아직 배포 구성을 만들지 못한 경우. Create a Deployment Configuration의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 최소한 2단계: CodeDeploy에 대한 서비스 역할 생성에 설명된 신뢰 및 권한으로 CodeDeploy를 신뢰하는 서비스 역할이 없는 경우. 서비스 역할을 만들고 구성하려면 2단계: CodeDeploy에 대한 서비스 역할 생성의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 대체 환경에서 인스턴스를 등록하기 위해 Elastic Load Balancing에서 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer를 생성하지 않았습니다. 자세한 내용은 Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정 단원을 참조하십시오.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 시작하기를 선택합니다.
3. 애플리케이션 이름에 애플리케이션의 이름을 입력합니다.
4. 컴퓨팅 플랫폼에서 EC2/온프레미스를 선택합니다.
5. 애플리케이션 생성을 선택합니다.

6. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다.
7. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.

> **Note**
>
> (배포 그룹 이름, 태그, Amazon EC2 Auto Scaling 그룹 이름 및 배포 구성을 비롯하여) 다른 배포 그룹에서 사용되는 것과 동일한 설정을 사용하려면 이 페이지에서 해당 설정을 선택합니다. 새 배포 그룹과 기존 배포 그룹의 이름이 같더라도 각 배포 그룹은 개별 애플리케이션과 연결되므로 CodeDeploy에서는 이러한 배포 그룹을 개별 배포 그룹으로 취급합니다.

8. 서비스 역할에서 대상 인스턴스에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할을 선택합니다.
9. 배포 유형에서 Blue/Green(블루/그린)을 선택합니다.
10. [Environment configuration]에서 대체 환경에 인스턴스를 제공하는 데 사용할 방법을 선택합니다.
    a. Amazon EC2 Auto Scaling 그룹 자동 복사: CodeDeploy 그룹은 지정된 그룹을 복사하여 Amazon EC2 Auto Scaling 그룹을 생성합니다.
    b. [Manually provision instances]: 배포를 만들 때까지 대체 환경에 필요한 인스턴스를 지정할 수 없습니다. 배포를 시작하려면 먼저 인스턴스를 만들어야 합니다. 대신, 여기서 대체할 인스턴스를 지정합니다.
11. 10단계에서 선택한 항목에 따라 다음 중 하나를 수행하세요.
    - Amazon EC2 Auto Scaling 그룹 자동 복사를 선택한 경우: Amazon EC2 Auto Scaling 그룹에서 대체 환경의 인스턴스에 대한 Amazon EC2 Auto Scaling 그룹의 템플릿으로 사용할 Amazon EC2 Auto Scaling 그룹의 이름을 선택하거나 입력합니다. 선택한 Amazon EC2 Auto Scaling 그룹의 현재 정상 인스턴스 개수가 대체 환경에서 생성됩니다.
    - 인스턴스 수동 프로비저닝을 선택한 경우: Amazon EC2 Auto Scaling 그룹이나 Amazon EC2 인스턴스 또는 둘 다에서 이 배포 그룹에 추가할 인스턴스를 지정할 수 있습니다. 원본 환경에서 인스턴스(즉, 대체할 인스턴스 또는 현재 애플리케이션 개정을 실행 중인 인스턴스)를 식별할 Amazon EC2 태그 값 또는 Amazon EC2 Auto Scaling 그룹 이름을 입력합니다.
12. 로드 밸런서에서 로드 밸런싱 활성화를 선택한 후 목록에서 대체 Amazon EC2 인스턴스를 등록할 Classic Load Balancer, Application Load Balancer 대상 그룹 및 Network Load Balancer 대상 그룹을 선택합니다. 각 대체 인스턴스는 선택한 모든 Classic Load Balancer 및 대상 그룹에 등록됩니다. 최대 10개의 Classic Load Balancer 및 10개의 대상 그룹으로 총 20개의 항목을 선택할 수 있습니다.

선택한 트래픽 재라우팅 및 배포 구성 설정에 따라 트래픽이 원본 인스턴스에서 대체 인스턴스로 다시 라우팅됩니다.

CodeDeploy 배포에 사용할 수 있는 로드 밸런서에 대한 자세한 내용은 Integrating CodeDeploy with Elastic Load Balancing 단원을 참조하세요.

13. [Deployment settings]에서 대체 환경으로 트래픽을 다시 라우팅하기 위한 기본 옵션(배포에 사용할 배포 구성임)과 배포 후 원본 환경의 인스턴스 처리 방법에 대한 기본 옵션을 검토합니다.

    설정을 변경하려면 다음 단계로 진행합니다. 그렇지 않으면 15단계로 건너뛰세요.

14. Blue/Green 배포에 대한 배포 설정을 변경하려면 다음 설정을 변경합니다.

설정	옵션
트래픽 다시 라우팅	• 즉시 트래픽 다시 라우팅: 대체 환경의 인스턴스가 프로비저닝되고 해당 인스턴스에 최신 애플리케이션 개정이 설치되면 바로 지정된 로드 밸런서 및 대상 그룹에 자동으로 등록되어 이러한 인스턴스로 트래픽이 자동으로 다시 라우팅됩니다. 그러면 원본 환경의 인스턴스가 등록 취소됩니다. • 트래픽을 다시 라우팅할지 여부를 선택합니다: 수동으로 트래픽을 다시 라우팅하지 않는 한, 대체 환경의 인스턴스는 지정된 로드 밸런서 및 대상 그룹에 등록되지 않습니다. 트래픽이 다시 라우팅되지 않고 지정된 시간이 경과되면 배포 상태가 중지됨으로 변경됩니다.
배포 구성	대체 환경의 인스턴스가 로드 밸런서 및 대상 그룹에 등록되는 속도를 선택합니다(예: 한 번에 하나씩 또는 한 번에 모두). **Note** 대체 환경으로 트래픽이 성공적으로 라우팅되면, 어떤 배포 구성을 선택했

설정	옵션
	든 간에 원본 환경의 인스턴스가 한번에 전부 등록 취소됩니다. 자세한 내용은 CodeDeploy에서 배포 구성 작업 단원을 참조하십시오.
[Original instances]	• 배포 그룹의 원본 인스턴스 종료: 트래픽이 대체 환경으로 다시 라우팅되면 지정한 대기 시간이 경과한 후 로드 밸런서 및 대상 그룹에서 등록 취소된 인스턴스가 종료됩니다. • 배포 그룹의 원본 인스턴스 계속 실행: 트래픽이 대체 환경으로 다시 라우팅된 후에도 로드 밸런서 및 대상 그룹에서 등록 취소된 인스턴스가 계속 실행됩니다.

15. (선택 사항) 고급에서 배포에 포함하려는 모든 옵션(예: Amazon SNS 알림 트리거, Amazon CloudWatch 경보 또는 자동 롤백)을 구성합니다.

    배포 그룹에서 고급 옵션 지정에 대한 자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하세요.

16. [Create deployment group]을 선택합니다.

다음 단계는 애플리케이션 및 배포 그룹에 배포할 개정을 준비하는 것입니다. 지침은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.

Amazon ECS 서비스 배포를 위한 애플리케이션 생성 (콘솔)

CodeDeploy 콘솔을 사용하여 Amazon ECS 서비스 배포를 위한 애플리케이션을 생성할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 시작하기를 선택합니다.
3. 애플리케이션 생성 페이지에서 codeDeploy 사용을 선택합니다.
4. 애플리케이션 이름에 애플리케이션의 이름을 입력합니다.
5. 컴퓨팅 플랫폼에서 Amazon ECS를 선택합니다.
6. 애플리케이션 생성을 선택합니다.
7. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다. Amazon ECS 배포에 대한 배포 그룹을 만드는 데 필요한 항목에 대한 자세한 정보는 Amazon ECS 배포를 시작하기 전 단원을 참조하세요.
8. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.

> **Note**
>
> (배포 그룹 이름 및 배포 구성을 포함하여) 다른 배포 그룹에서 사용되는 것과 동일한 설정을 사용하려면 이 페이지에서 해당 설정을 선택합니다. 새 그룹과 기존 그룹의 이름이 같더라도 각 배포 그룹은 개별 애플리케이션과 연결되므로 CodeDeploy는 이러한 배포 그룹을 개별 배포 그룹으로 취급합니다.

9. 서비스 역할에서 Amazon ECS에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할을 선택합니다. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.
10. 로드밸런서 이름에서 Amazon ECS 서비스에 트래픽을 공급하는 로드밸런서의 이름을 선택합니다.
11. 프로덕션 리스너 포트에서 해당 Amazon ECS 서비스에 서비스 프로덕션 트래픽을 공급하는 리스너의 프로토콜과 포트를 선택합니다.
12. (선택 사항) 테스트 리스너 포트에서 배포 중 해당 Amazon ECS 서비스의 대체 작업 세트에 트래픽을 공급하는 테스트 리스너의 프로토콜과 포트를 선택합니다. `AfterAllowTestTraffic` 후크 중 실행하는 AppSpec 파일의 Lambda 함수를 한 개 이상 지정할 수 있습니다. 이 함수는 확인 테스트를 실행할 수 있습니다. 확인 테스트가 실패하면 배포 롤백이 트리거됩니다. 확인 테스트가 성공하면 배포 수명 주기의 다음 후크인 `BeforeAllowTraffic`이 트리거됩니다. 테스트 리스너 포트를 지정하지 않으면 `AfterAllowTestTraffic` 후크 중 아무것도 수행되지 않습니다. 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션 단원을 참조하십시오.

13. 대상 그룹 1 이름과 대상 그룹 2 이름에서 배포 중 트래픽을 라우팅하는 데 사용되는 대상 그룹을 선택합니다. CodeDeploy는 한 대상 그룹을 Amazon ECS 서비스의 기존 작업 세트에 바인딩하고 다른 대상 그룹을 대체 작업 세트에 바인딩합니다. 자세한 내용은 Application Load Balancer의 대상 그룹 지정을 참조하세요.
14. 즉시 트래픽 다시 라우팅 또는 트래픽을 언제 다시 라우팅할지 지정을 선택하여 업데이트된 Amazon ECS 서비스에 트래픽을 언제 다시 라우팅할지를 지정합니다.

    Reroute traffic immediately(즉시 트래픽 다시 라우팅)를 선택한 경우 대체 작업 세트가 프로비저닝된 후 배포가 트래픽을 자동으로 다시 라우팅합니다.

    Specify when to reroute traffic(트래픽을 언제 다시 라우팅할지 지정)을 선택한 경우 대체 작업 세트가 성공적으로 프로비저닝된 후 대기할 일, 시간 및 분을 선택합니다. 이 대기 시간 동안 AppSpec 파일에 지정된 Lambda 함수의 유효성 테스트가 실행됩니다. 트래픽이 다시 라우팅되기 전에 대기 시간이 만료되면 배포 상태가 `Stopped`로 변경됩니다.
15. 기존 개정 종료에서 배포 성공 후 얼마 후에 Amazon ECS 서비스의 기존 작업 세트가 종료될지를 일, 시간, 분으로 선택합니다.
16. (선택 사항) 고급에서 배포에 포함하려는 모든 옵션(예: Amazon SNS 알림 트리거, Amazon CloudWatch 경보 또는 자동 롤백)을 구성합니다.

    자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하십시오.

AWS Lambda 함수 배포를 위한 애플리케이션 생성(콘솔)

CodeDeploy 콘솔을 사용하여 AWS Lambda 함수 배포를 위한 애플리케이션을 생성할 수 있습니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 시작하기를 선택합니다.
3. 애플리케이션 생성 페이지에서 codeDeploy 사용을 선택합니다.
4. 애플리케이션 이름에 애플리케이션의 이름을 입력합니다.
5. 컴퓨팅 플랫폼에서 AWS Lambda을(를) 선택합니다.
6. 애플리케이션 생성을 선택합니다.

7. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다.
8. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.

> **Note**
>
> (배포 그룹 이름 및 배포 구성을 포함하여) 다른 배포 그룹에서 사용되는 것과 동일한 설정을 사용하려면 이 페이지에서 해당 설정을 선택합니다. 새 배포 그룹과 기존 배포 그룹의 이름이 같더라도 각 배포 그룹은 개별 애플리케이션과 연결되므로 CodeDeploy는 이러한 배포 그룹을 개별 배포 그룹으로 취급합니다.

9. 서비스 역할에서 CodeDeploy에 액세스 권한을 부여하는 서비스 역할을 선택합니다 AWS Lambda. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.
10. 미리 정의된 배포 구성을 사용하려면 배포 구성을 선택한 다음 12단계로 건너뜁니다. 사용자 지정 구성을 생성하려면 다음 단계를 계속합니다.

    배포 구성에 대한 자세한 내용은 AWS Lambda 컴퓨팅 플랫폼의 배포 구성 단원을 참조하세요.
11. 사용자 지정 구성을 생성하려면배포 구성 생성을 클릭한 후 다음을 수행합니다.

    a. 배포 구성 이름에 구성 이름을 입력합니다.

    b. 유형에서 구성 유형을 선택합니다. Canary를 선택한 경우 트래픽이 2 증분씩 이동합니다. Linear를 선택한 경우 트래픽이 동일한 증분으로 이동하며 각 증분 간에 시간(분)이 동일합니다.

    c. 단계에 이동할 트래픽의 백분율(1~99 사이)을 입력합니다. 구성 유형이 Canary인 경우에 첫 번째 증분이 이동되는 트래픽의 비율입니다. 나머지 트래픽은 선택한 간격 후에 두 번째 증분으로 이동합니다. 구성 유형이 Linear인 경우에 각 간격이 시작될 때 이동되는 트래픽의 비율입니다.

    d. 간격에 시간(분)을 입력합니다. 구성 유형이 Canary인 경우 첫 번째와 두 번째 트래픽 이동 사이의 시간(분)입니다. 구성 유형이 Linear인 경우 각 증분 이동 사이의 시간(분)입니다.

    > **Note**
    >
    > AWS Lambda 배포의 최대 길이는 2일 또는 2,880분입니다. 따라서 canary 구성의 간격에 지정되는 최대 값은 2,800분입니다. linear(선형) 구성의 최대 값은 단계의 값에 따릅니다. 예를 들어 선형 트래픽 이동의 단계 백분율이 25%인 경우 트래픽 이동이 네 개입니다. 최대 간격 값은 2,880을 4분 또는 720분으로 나눈 값입니다.

    e. 배포 구성 만들기를 선택합니다.

12. (선택 사항) 고급에서 배포에 포함하려는 모든 옵션(예: Amazon SNS 알림 트리거, Amazon CloudWatch 경보 또는 자동 롤백)을 구성합니다.

    자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하십시오.

13. [Create deployment group]을 선택합니다.

애플리케이션(CLI) 생성

AWS CLI 를 사용하여 애플리케이션을 생성하려면 create-application 명령을 호출하여 애플리케이션을 고유하게 나타내는 이름을 지정합니다. (계정 AWS 에서 CodeDeploy 애플리케이션 이름은 리전당 한 번만 사용할 수 있습니다. 다른 리전에서 애플리케이션 이름을 재사용할 수 있습니다.)

AWS CLI 를 사용하여 애플리케이션을 생성한 후 다음 단계는 개정을 배포할 인스턴스를 지정하는 배포 그룹을 생성하는 것입니다. 지침은 CodeDeploy에서 배포 그룹 만들기 단원을 참조하세요.

배포 그룹을 만든 후 다음 단계는 애플리케이션 및 배포 그룹에 배포할 개정을 준비하는 것입니다. 지침은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.

CodeDeploy로 애플리케이션 세부 정보 보기

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy APIs를 사용하여 AWS 계정과 연결된 모든 애플리케이션에 대한 세부 정보를 볼 수 있습니다.

주제

- 애플리케이션 세부 정보 보기(콘솔)
- 애플리케이션 세부 정보 보기(CLI)

애플리케이션 세부 정보 보기(콘솔)

CodeDeploy 콘솔을 사용하여 애플리케이션의 세부 정보를 보려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 시작하기를 선택합니다.
3. 추가 애플리케이션 세부 정보를 보려면 목록에서 애플리케이션 이름을 선택합니다.

애플리케이션 세부 정보 보기(CLI)

AWS CLI 를 사용하여 애플리케이션 세부 정보를 보려면 get-application 명령, batch-get-application 명령 또는 list-applications 명령을 호출합니다.

단일 애플리케이션에 대한 세부 정보를 보려면 get-application 명령을 호출하고 애플리케이션 이름을 지정합니다.

여러 애플리케이션에 대한 세부 정보를 보려면 batch-get-applications 명령을 호출하여 여러 애플리케이션의 이름을 지정합니다.

애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.

알림 규칙 생성

배포 성공 및 실패와 같이 배포 애플리케이션이 변경된 경우 알림 규칙을 사용하여 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 CodeDeploy.

알림 규칙을 생성하려면 (콘솔)

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

   자세한 내용은 알림 내용 및 보안 이해를 참조하세요.
6. 알림을 트리거하는 이벤트에서 알림을 보내고자 하는 이벤트를 선택합니다.

카테고리	이벤트
배포	실패
	성공
	시작됨

7. 대상에서 SNS 주제 생성을 선택합니다.

> **Note**
>
> 주제를 생성하면 CodeDeploy가 주제에 이벤트를 게시하도록 허용하는 정책이 적용됩니다. CodeDeploy 알림용으로 특별히 생성된 주제를 사용하면 이 배포 애플리케이션에 대한 알림을 보려는 해당 주제의 가입 목록에만 사용자를 추가할 수 있습니다.

codestar-notifications- 접두사 뒤에 주제 이름을 입력한 다음 제출을 선택합니다.

> **Note**
>
> 새 항목을 만드는 대신 기존 Amazon SNS 항목을 사용하려면 대상에서 해당 ARN을 선택합니다. 주제에 적절한 액세스 정책이 있는지와 구독자 목록에 배포 애플리케이션에 대한 정보를 볼 수 있도록 허용된 사용자만 포함되어 있는지 확인합니다. 자세한 내용은 알림에 대한 기존 Amazon SNS 주제 구성과 알림 내용 및 보안 이해를 참조하세요.

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

알림 규칙을 생성하려면(AWS CLI)

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

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

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

2. 일반 텍스트 편집기에서 JSON 파일을 열고 규칙에 대해 원하는 리소스, 이벤트 유형 및 Amazon SNS 대상을 포함하도록 편집합니다. 다음 예제는 ID가 *123456789012*인 AWS 계정의 *MyDeploymentApplication*이라는 애플리케이션에 **MyNotificationRule** 대한 라는 알림 규칙을 보여줍니다. 배포가 성공적일 때 *codestar-notifications-MyNotificationTopic*이라는 Amazon SNS 주제에 전체 세부 정보 유형과 함께 알림이 전송됩니다.

```
{
    "Name": "MyNotificationRule",
    "EventTypeIds": [
        "codedeploy-application-deployment-succeeded"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyDeploymentApplication",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

파일을 저장합니다.

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

```
aws 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"
}
```

CodeDeploy 애플리케이션 이름 바꾸기

AWS CLI 또는 CodeDeploy APIs를 사용하여 애플리케이션의 이름을 변경할 수 있습니다.

애플리케이션 이름 목록을 보려면 AWS CLI 를 사용하여 list-applications 명령을 호출합니다.

를 사용하여 애플리케이션 이름을 변경하는 AWS CLI 방법에 대한 자세한 내용은 update-application을 참조하세요.

CodeDeploy API를 사용하여 애플리케이션 이름을 변경하는 방법에 대한 자세한 내용은 API_UpdateApplication을 참조하세요.

CodeDCodeDeploy에서 애플리케이션을 삭제합니다.

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy API 작업을 사용하여 애플리케이션을 삭제할 수 있습니다. CodeDeploy API 작업을 사용하는 방법에 대한 자세한 내용은 DeleteApplication을 참조하세요.

> **Warning**
>
> 애플리케이션을 삭제하면 관련된 모든 배포 그룹 정보 및 배포 세부 정보를 포함해 CodeDeploy 시스템에서 애플리케이션에 대한 정보가 제거됩니다. EC2/온프레미스 배포용으로 생성된 애플리케이션을 삭제해도 인스턴스에서 애플리케이션 수정 버전이 제거되지 않으며 Amazon S3 버킷에서 수정 버전이 삭제되지 않습니다. EC2/온프레미스 배포용으로 생성된 애플리케이션을 삭제해도 Amazon EC2 인스턴스가 종료되거나 온프레미스 인스턴스의 등록이 취소되지 않습니다. 이 작업은 실행을 취소할 수 없습니다.

주제

- 애플리케이션 삭제(콘솔)
- 애플리케이션(AWS CLI)을 삭제합니다.

애플리케이션 삭제(콘솔)

CodeDeploy 콘솔을 사용하여 애플리케이션을 삭제하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포(Deploy)를 확장하고 애플리케이션(Applications)을 선택합니다.
3. 애플리케이션 목록에서 삭제할 애플리케이션을 선택합니다.

   애플리케이션에 대한 세부 정보를 포함하는 페이지가 나타납니다.
4. 오른쪽 상단에서 애플리케이션 삭제를 선택합니다.
5. 메시지가 표시되면 **delete**를 입력하여 삭제할 것인지 확인한 후 삭제를 선택합니다.

애플리케이션(AWS CLI)을 삭제합니다.

AWS CLI 를 사용하여 애플리케이션을 삭제하려면 애플리케이션 이름을 지정하여 delete-application 명령을 호출합니다. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.

CodeDeploy에서 배포 그룹 작업

CodeDeploy 애플리케이션에 대해 배포 그룹을 하나 이상 지정할 수 있습니다. 각 애플리케이션 배포는 해당 배포 그룹 중 하나를 사용합니다. 배포 그룹에는 배포 중에 사용되는 설정 및 구성이 포함됩니다. 대부분의 배포 그룹 설정은 애플리케이션에서 사용하는 컴퓨팅 플랫폼에 따라 다릅니다. 모든 컴퓨팅 플랫폼의 배포 그룹에 대해 롤백, 트리거 및 경보와 같은 일부 설정을 구성할 수 있습니다.

Amazon ECS 컴퓨팅 플랫폼 배포의 배포 그룹

Amazon ECS 배포에서 배포 그룹은 Amazon ECS 서비스, 로드 밸런서, 테스트 리스너 옵션 및 두 개의 대상 그룹을 지정합니다. 또한 대체 작업 세트로 트래픽을 재라우팅할 시기와 배포 성공 후 원래 작업 세트 및 Amazon ECS 애플리케이션을 종료할 시기를 지정합니다.

AWS Lambda 컴퓨팅 플랫폼 배포의 배포 그룹

AWS Lambda 배포에서 배포 그룹은 향후 AWS Lambda 함수 배포를 위한 CodeDeploy 구성 세트를 정의합니다. 예를 들어 배포 그룹은 Lambda 함수의 새 버전으로 트래픽을 라우팅하는 방법을 지정합니다. 또한 경보 및 롤백을 지정할 수도 있습니다. AWS Lambda 배포 그룹의 단일 배포는 하나 이상의 그룹 구성을 재정의할 수 있습니다.

EC2/온프레미스 컴퓨팅 플랫폼 배포의 배포 그룹

EC2/온프레미스 배포에서 배포 그룹은 배포를 대상으로 하는 개별 인스턴스 집합입니다. 배포 그룹에는 개별적으로 태그가 지정된 인스턴스, Amazon EC2 Auto Scaling 그룹의 Amazon EC2 인스턴스 또는 둘 다가 포함됩니다.

인 플레이스(in-place) 배포에서 배포 그룹의 인스턴스는 최신 애플리케이션 개정 버전으로 업데이트됩니다.

블루/그린 배포에서는 하나 이상의 로드 밸런서에서 원래 인스턴스의 등록을 취소하고 일반적으로 최신 애플리케이션 개정 버전이 이미 설치된 대체 인스턴스 집합을 등록하여 인스턴스 집합에서 다른 인스턴스 집합으로 트래픽이 다시 라우팅됩니다.

CodeDeploy의 애플리케이션과 둘 이상의 배포 그룹을 연결할 수 있습니다. 따라서 애플리케이션 개정 버전을 서로 다른 시간에 서로 다른 인스턴스 세트에 배포할 수 있습니다. 예를 들어 하나의 배포 그룹을 사용하여 코드 품질을 보장하는 `Test` 태그의 인스턴스 세트에 애플리케이션 개정 버전을 배포

합니다. 그런 다음 추가적인 확인을 위해 `Staging` 태그의 인스턴스가 있는 배포 그룹에 동일한 애플리케이션 개정 버전을 배포합니다. 마지막으로 고객에게 최신 애플리케이션을 릴리스할 준비가 되면 `Production` 태그의 인스턴스가 포함된 배포 그룹에 배포합니다.

여러 태그 그룹을 사용하여 배포 그룹에 포함된 인스턴스의 기준을 더욱 세분화할 수도 있습니다. 자세한 내용은 Tagging Instances for Deployments 단원을 참조하세요.

CodeDeploy 콘솔을 사용하여 애플리케이션을 만드는 경우 동시에 이 애플리케이션의 첫 번째 배포 그룹을 구성합니다. AWS CLI 를 사용하여 애플리케이션을 생성할 때 별도의 단계에서 첫 번째 배포 그룹을 생성합니다.

AWS 계정과 이미 연결된 배포 그룹 목록을 보려면 섹션을 참조하세요CodeDeploy에서 배포 그룹 세부 정보 보기.

Amazon EC2 인스턴스 태그에 대한 자세한 정보는 콘솔을 사용한 태그 작업을 참조하세요. 온프레미스 인스턴스에 대한 자세한 정보는 Working with On-Premises Instances 단원을 참조하세요. Amazon EC2 Auto Scaling에 대한 자세한 내용은 Amazon EC2 Auto Scaling과 CodeDeploy 통합 단원을 참조하세요.

주제

- the section called “배포 그룹 만들기”
- the section called “배포 그룹 세부 정보 보기”
- the section called “배포 그룹 설정 변경”
- the section called “배포 그룹에 대한 고급 옵션 구성”
- the section called “배포 그룹 삭제”

CodeDeploy에서 배포 그룹 만들기

CodeDeploy 콘솔, AWS CLI, CodeDeploy APIs 또는 CloudFormation 템플릿을 사용하여 배포 그룹을 생성할 수 있습니다. CloudFormation 템플릿을 사용하여 배포 그룹을 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요CloudFormation CodeDeploy 참조용 템플릿.

CodeDeploy 콘솔을 사용하여 애플리케이션을 만드는 경우 동시에 이 애플리케이션의 첫 번째 배포 그룹을 구성합니다. AWS CLI 를 사용하여 애플리케이션을 생성할 때 별도의 단계에서 첫 번째 배포 그룹을 생성합니다.

배포 그룹 만들기의 일부로 서비스 역할을 지정해야 합니다. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.

주제

- 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)
- EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)
- Amazon ECS 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)
- Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정
- CodeDeploy Amazon ECS 배포를 위한 로드 밸런서, 대상 그룹 및 리스너 설정
- 배포 그룹 만들기(CLI)

인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)

CodeDeploy 콘솔을 사용하여 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹을 만들려면:

> **Warning**
>
> 다음과 같은 경우 아래 단계를 수행하지 마세요.
>
> - 애플리케이션의 첫 번째 CodeDeploy 배포에 사용할 인스턴스를 아직 준비하지 않은 경우. 인스턴스를 설정하려면 CodeDeploy용 인스턴스 작업 섹션의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 사용자 지정 배포 구성을 사용하는 배포 그룹을 만들고 싶지만 아직 배포 구성을 만들지 못한 경우. Create a Deployment Configuration의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 최소한 2단계: CodeDeploy에 대한 서비스 역할 생성에 설명된 신뢰 및 권한으로 CodeDeploy를 신뢰하는 서비스 역할이 없는 경우. 서비스 역할을 만들고 구성하려면 2단계: CodeDeploy에 대한 서비스 역할 생성의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
> - 인 플레이스(in-place) 배포에 대해 Elastic Load Balancing에서 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer를 선택하려고 하지만 아직 생성하지 않은 경우.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. [Applications] 페이지에서 배포 그룹을 만들려는 애플리케이션의 이름을 선택합니다.
4. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다.
5. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.

> **Note**
>
> (배포 그룹 이름, 태그, Amazon EC2 Auto Scaling 그룹 이름 또는 둘 다 및 배포 구성을 비롯하여) 다른 배포 그룹에서 사용되는 것과 동일한 설정을 사용하려면 이 페이지에서 지정합니다. 새 배포 그룹과 기존 배포 그룹의 이름이 같더라도 각 배포 그룹은 개별 애플리케이션과 연결되므로 CodeDeploy에서는 이러한 배포 그룹을 개별 배포 그룹으로 취급합니다.

6. 서비스 역할에서 대상 인스턴스에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할을 선택합니다.
7. 배포 유형에서 In-place(현재 위치)를 선택합니다.
8. 환경 구성에서 다음을 수행합니다.

   a. 애플리케이션을 Amazon EC2 Auto Scaling 그룹에 배포하려면 Amazon EC2 Auto Scaling 그룹을 선택한 다음 애플리케이션 리비전을 배포할 Amazon EC2 Auto Scaling 그룹의 이름을 선택합니다. 새 Amazon EC2 인스턴스가 Amazon EC2 Auto Scaling 그룹의 일부로 시작되면 CodeDeploy에서는 새 인스턴스로 개정을 자동으로 배포할 수 있습니다. 배포 그룹당 최대 10개의 Amazon EC2 Auto Scaling 그룹을 추가할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling과 CodeDeploy 통합 단원을 참조하십시오.

   b. Amazon EC2 Auto Scaling 그룹을 선택한 경우, 배포 그룹을 생성하거나 업데이트할 때 Auto Scaling 그룹에 종료 후크를 설치하도록 하려면 선택적으로 Auto Scaling 그룹에 종료 후크 추가를 선택합니다. 이 후크가 설치되면 CodeDeploy가 종료 배포를 수행합니다. 자세한 내용은 Auto Scaling 확장 이벤트 중 종료 배포 활성화 단원을 참조하십시오.

   c. 인스턴스에 태그를 지정하려면 Amazon EC2 인스턴스 또는 온프레미스 인스턴스를 선택합니다. 키 및 값 필드에 인스턴스에 태그를 지정하는 데 사용한 키-값 쌍의 값을 입력합니다. 한 태그 그룹에서 최대 10개의 키-값 페어에 태그를 지정할 수 있습니다.

i. 값 필드에서 와일드카드를 사용해 특정 패턴으로 태그가 지정된 모든 인스턴스를 식별할 수 있습니다(예: Amazon EC2 인스턴스, 코스트 센터 및 그룹 이름 등). 예를 들어, 키필드에서 이름을 선택하고 값 필드에서 **GRP-*a**를 입력하면 CodeDeploy가 해당 패턴에 맞는 모든 인스턴스(예: **GRP-1a**, **GRP-2a**, 및 **GRP-XYZ-a**)를 식별합니다.

ii. 값 필드는 대/소문자를 구분합니다.

iii. 목록에서 키-값 페어를 제거하려면 제거 아이콘을 선택합니다.

CodeDeploy에서는 지정된 각 키-값 페어 또는 Amazon EC2 Auto Scaling 그룹 이름과 일치하는 인스턴스를 찾기 때문에 일치하는 인스턴스 개수가 표시됩니다. 인스턴스에 대한 자세한 내용을 확인하려면 이 숫자를 클릭하세요.

배포된 인스턴스의 기준을 구체화하려면 태그 그룹 추가를 선택하여 태그 그룹을 생성합니다. 키-값 페어가 각각 최대 10개인 태그 그룹을 최대 세 개까지 만들 수 있습니다. 배포 그룹에서 여러 태그 그룹을 사용하는 경우 모든 태그 그룹으로 식별되는 인스턴스만 배포 그룹에 포함됩니다. 즉 인스턴스는 배포 그룹에 포함될 각 그룹의 태그 중 적어도 하나와 일치해야 합니다.

태그 그룹을 사용하여 배포 그룹을 세분화하는 방법에 대한 자세한 내용은 Tagging Instances for Deployments 단원을 참조하세요.

9. Systems Manager를 사용한 에이전트 구성(Agent configuration with Systems Manager)에서 배포 그룹의 인스턴스에 CodeDeploy 에이전트를 설치하고 업데이트하는 방법을 지정합니다. CodeDeploy 에이전트에 대한 자세한 내용은 CodeDeploy 에이전트 작업을 참조하세요. Systems Manager에 대한 자세한 내용은 Systems Manager란 무엇입니까?를 참조하세요.

   a. 안 함: Systems Manager를 사용한 CodeDeploy 설치 구성을 건너뜁니다. 인스턴스에 배포에 사용할 에이전트가 설치되어 있어야 하므로 CodeDeploy 에이전트를 다른 방법으로 설치할 경우에만 이 옵션을 선택합니다.

   b. 한 번만: Systems Manager가 배포 그룹의 모든 인스턴스에 CodeDeploy 에이전트를 한 번 설치합니다.

   c. 지금 설치 및 업데이트 예약: Systems Manager가 상태 관리자와 연결을 생성하여, 사용자가 구성한 일정에 따라 CodeDeploy 에이전트를 설치합니다. 상태 관리자 및 연결에 대한 자세한 내용은 상태 관리자 정보를 참조하세요.

10. Deployment configuration(배포 구성)에서 인스턴스가 배포되는 속도를 제어하는 배포 구성을 선택합니다(예: 한 번에 하나씩 또는 한 번에 모두). 배포 구성에 대한 자세한 내용은 CodeDeploy에서 배포 구성 작업 단원을 참조하세요.

11. (선택 사항) 로드 밸런서에서 로드 밸런싱 활성화를 선택한 후, 목록에서 CodeDeploy 배포 프로세스 중 인스턴스로 가는 트래픽을 관리할 Classic Load Balancer, Application Load Balancer 대상 그룹 및 Network Load Balancer 대상 그룹을 선택합니다. 최대 10개의 Classic Load Balancer 및 10개의 대상 그룹으로 총 20개의 항목을 선택할 수 있습니다. 배포하려는 Amazon EC2 인스턴스가 선택한 로드 밸런서(Classic Load Balancer) 또는 대상 그룹(Application Load Balancer 및 Network Load Balancer)에 등록되어 있는지 확인합니다.

    배포 중에는 선택한 로드 밸런서 및 대상 그룹에서 원본 인스턴스의 등록이 취소되어 배포 중에 트래픽이 이러한 인스턴스로 라우팅되는 것이 방지됩니다. 배포가 완료되면 각 인스턴스가 선택한 모든 Classic Load Balancer 및 대상 그룹에 다시 등록됩니다.

    CodeDeploy 배포에 사용할 수 있는 로드 밸런서에 대한 자세한 내용은 Integrating CodeDeploy with Elastic Load Balancing 단원을 참조하세요.

    > **Warning**
    >
    > 이 배포 그룹에서 오토 스케일링과 Elastic Load Balancing 로드 밸런서를 모두 구성하고 로드 밸런서를 오토 스케일링에 연결하려는 경우 이 배포 그룹에서 CodeDeploy 배포를 생성하기 전에 이 연결을 완료하는 것이 좋습니다. 배포를 생성한 후 연결을 완료하려고 하면 예기치 않게 로드 밸런서에서 모든 인스턴스의 등록이 취소될 수 있습니다.

12. (선택 사항) 고급을 확장하고 배포에 포함하려는 모든 옵션(예: Amazon SNS 알림 트리거, Amazon CloudWatch 경보, Auto Scaling 옵션 또는 자동 롤백)을 구성합니다.

    자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하십시오.

13. [Create deployment group]을 선택합니다.

EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)

CodeDeploy 콘솔을 사용하여 블루/그린 배포에 사용할 수 있는 배포 그룹을 만들려면:

> **Warning**
>
> 다음과 같은 경우 아래 단계를 수행하지 마세요.
>
> - Blue/Green 배포 프로세스 중 대체하려는 인스턴스에 CodeDeploy 에이전트가 설치되어 있지 않은 경우. 인스턴스를 설정하려면 CodeDeploy용 인스턴스 작업 섹션의 지침을 수행한 다음 이 주제의 단계를 수행하세요.

- 사용자 지정 배포 구성을 사용하는 애플리케이션을 만들고 싶지만 아직 배포 구성을 만들지 못한 경우. Create a Deployment Configuration의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
- 최소한 2단계: CodeDeploy에 대한 서비스 역할 생성에 설명된 신뢰 및 권한으로 CodeDeploy를 신뢰하는 서비스 역할이 없는 경우. 서비스 역할을 만들고 구성하려면 2단계: CodeDeploy에 대한 서비스 역할 생성의 지침을 수행한 다음 이 주제의 단계를 수행하세요.
- 대체 환경에서 인스턴스를 등록하기 위해 Elastic Load Balancing에서 Classic Load Balancer 또는 Application Load Balancer를 생성하지 않은 경우. 자세한 내용은 Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정 단원을 참조하십시오.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. [Applications] 페이지에서 배포 그룹을 만들려는 애플리케이션의 이름을 선택합니다.
4. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다.
5. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.

   > **Note**
   >
   > (배포 그룹 이름, 태그, Amazon EC2 Auto Scaling 그룹 이름 및 배포 구성을 비롯하여) 다른 배포 그룹에서 사용되는 것과 동일한 설정을 사용하려면 이 페이지에서 해당 설정을 선택합니다. 새 배포 그룹과 기존 배포 그룹의 이름이 같더라도 각 배포 그룹은 개별 애플리케이션과 연결되므로 CodeDeploy에서는 이러한 배포 그룹을 개별 배포 그룹으로 취급합니다.

6. 서비스 역할에서 대상 인스턴스에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할을 선택합니다.
7. 배포 유형에서 Blue/Green(블루/그린)을 선택합니다.
8. 환경 구성에서 다음을 수행합니다.

- 교체 환경에 인스턴스를 제공하는 데 사용할 방법을 선택합니다. 다음과 같은 옵션이 있습니다.
  - Amazon EC2 Auto Scaling 그룹 자동 복사: CodeDeploy 그룹은 지정된 그룹을 복사하여 Amazon EC2 Auto Scaling 그룹을 생성합니다.
  - [Manually provision instances]: 배포를 만들 때까지 대체 환경에 필요한 인스턴스를 지정할 수 없습니다. 배포를 시작하려면 먼저 인스턴스를 만들어야 합니다. 대신, 여기서 대체할 인스턴스를 지정합니다.
- Amazon EC2 Auto Scaling 그룹 자동 복사를 선택한 경우, 선택적으로 Auto Scaling 그룹에 종료 후크 추가를 선택하여 배포 그룹을 생성하거나 업데이트할 때 CodeDeploy가 Auto Scaling 그룹에 종료 후크를 설치하도록 합니다. 이 후크가 설치되면 CodeDeploy가 종료 배포를 수행합니다. 자세한 내용은 Auto Scaling 확장 이벤트 중 종료 배포 활성화 단원을 참조하십시오.

9. Systems Manager를 사용한 에이전트 구성(Agent configuration with Systems Manager)에서 배포 그룹의 인스턴스에 CodeDeploy 에이전트를 설치하고 업데이트하는 방법을 지정합니다. CodeDeploy 에이전트에 대한 자세한 내용은 CodeDeploy 에이전트 작업을 참조하세요. Systems Manager에 대한 자세한 내용은 Systems Manager란 무엇입니까?를 참조하세요.

   a. 안 함: Systems Manager를 사용한 CodeDeploy 설치 구성을 건너뜁니다. 인스턴스에 배포에 사용할 에이전트가 설치되어 있어야 하므로 CodeDeploy 에이전트를 다른 방법으로 설치할 경우에만 이 옵션을 선택합니다.

   b. 한 번만: Systems Manager가 배포 그룹의 모든 인스턴스에 CodeDeploy 에이전트를 한 번 설치합니다.

   c. 지금 설치 및 업데이트 예약: Systems Manager가 상태 관리자와 연결을 생성하여, 사용자가 구성한 일정에 따라 CodeDeploy 에이전트를 설치합니다. 상태 관리자 및 연결에 대한 자세한 내용은 상태 관리자 정보를 참조하세요.

10. 8단계에서 선택한 항목에 따라 다음 중 하나를 수행하세요.

    - Amazon EC2 Auto Scaling 그룹 자동 복사를 선택한 경우: Amazon EC2 Auto Scaling 그룹에서 대체 환경의 인스턴스에 대해 생성한 Amazon EC2 Auto Scaling 그룹의 템플릿으로 사용할 Amazon EC2 Auto Scaling 그룹의 이름을 선택하거나 입력합니다. 선택한 Amazon EC2 Auto Scaling 그룹의 현재 정상 인스턴스 개수가 대체 환경에서 생성됩니다.
    - 인스턴스 수동 프로비저닝을 선택한 경우: Amazon EC2 Auto Scaling 그룹, Amazon EC2 Auto Scaling 인스턴스 또는 둘 다를 선택하여이 배포 그룹에 추가할 인스턴스를 지정합니다. 원본 환경의 인스턴스(즉, 대체할 인스턴스 또는 현재 애플리케이션 개정을 실행 중인 인스턴스)를 식별할 Amazon EC2 Auto Scaling 태그 값 또는 Amazon EC2 Auto Scaling 그룹 이름을 입력합니다.

11. 로드 밸런서에서 로드 밸런싱 활성화를 선택한 후 목록에서 대체 Amazon EC2 인스턴스를 등록할 Classic Load Balancer, Application Load Balancer 대상 그룹 및 Network Load Balancer 대상 그룹을 선택합니다. 각 대체 인스턴스는 선택한 모든 Classic Load Balancer 및 대상 그룹에 등록됩니다. 최대 10개의 Classic Load Balancer 및 10개의 대상 그룹으로 총 20개의 항목을 선택할 수 있습니다.

    선택한 트래픽 재라우팅 및 배포 구성 설정에 따라 트래픽이 원본 인스턴스에서 대체 인스턴스로 다시 라우팅됩니다.

    CodeDeploy 배포에 사용할 수 있는 로드 밸런서에 대한 자세한 내용은 Integrating CodeDeploy with Elastic Load Balancing 단원을 참조하세요.

    > ⚠ Warning
    >
    > 이 배포 그룹에서 오토 스케일링과 Elastic Load Balancing 로드 밸런서를 모두 구성하고 로드 밸런서를 오토 스케일링에 연결하려는 경우 이 배포 그룹에서 CodeDeploy 배포를 생성하기 전에 이 연결을 완료하는 것이 좋습니다. 배포를 생성한 후 연결을 완료하려고 하면 예기치 않게 로드 밸런서에서 모든 인스턴스의 등록이 취소될 수 있습니다.

12. [Deployment settings]에서 대체 환경으로 트래픽을 다시 라우팅하기 위한 기본 옵션(배포에 사용할 배포 구성임)과 배포 후 원본 환경의 인스턴스 처리 방법에 대한 기본 옵션을 검토합니다.

    설정을 변경하려면 다음 단계로 진행합니다. 그렇지 않으면 14단계로 건너뛰세요.

13. 블루/그린 배포에 대한 배포 설정을 변경하려면 다음 설정을 선택합니다.

설정	옵션
트래픽 다시 라우팅	• 즉시 트래픽 다시 라우팅: 대체 환경의 인스턴스가 프로비저닝되고 해당 인스턴스에 최신 애플리케이션 개정이 설치되면 바로 지정된 로드 밸런서 및 대상 그룹에 자동으로 등록되어 이러한 인스턴스로 트래픽이 자동으로 다시 라우팅됩니다. 그러면 원본 환경의 인스턴스가 등록 취소됩니다. • 트래픽을 다시 라우팅할지 여부를 선택합니다: 수동으로 트래픽을 다시 라우팅하지 않는 한, 대체 환경의 인스턴스는 지정된 로드

설정	옵션
	밸런서 및 대상 그룹에 등록되지 않습니다. 트래픽이 다시 라우팅되지 않고 지정된 시간이 경과되면 배포 상태가 중지됨으로 변경됩니다.
배포 구성	대체 환경의 인스턴스가 로드 밸런서 및 대상 그룹에 등록되는 속도를 선택합니다(예: 한 번에 하나씩 또는 한 번에 모두). **Note** 대체 환경으로 트래픽이 성공적으로 라우팅되면, 어떤 배포 구성을 선택했든 간에 원본 환경의 인스턴스가 한번에 전부 등록 취소됩니다. 자세한 내용은 CodeDeploy에서 배포 구성 작업 단원을 참조하십시오.

설정	옵션
[Original instances]	• 배포 그룹의 원본 인스턴스 종료: 트래픽이 대체 환경으로 다시 라우팅되면 지정한 대기 시간이 경과한 후 로드 밸런서 및 대상 그룹에서 등록 취소된 인스턴스가 종료됩니다. 다음 사항에 유의하세요. • Auto Scaling 그룹에 Deletion Protection이 활성화된 경우(`prevent-force-deletion` 또는 `prevent-all-deletion`) CodeDeploy 블루/그린 배포는 배포 정리 중에 삭제하기 전에 원래(블루) Auto Scaling 그룹에서이 보호를 비활성화합니다. 대체(녹색) Auto Scaling 그룹은 원래 그룹과 동일한 삭제 방지 설정을 유지합니다. • 배포 그룹의 원본 인스턴스 계속 실행: 트래픽이 대체 환경으로 다시 라우팅된 후에도 로드 밸런서 및 대상 그룹에서 등록 취소된 인스턴스가 계속 실행됩니다.

14. (선택 사항) 고급에서 배포에 포함하려는 모든 옵션(예: Amazon SNS 알림 트리거, Auto Scaling 옵션, Amazon CloudWatch 경보 또는 자동 롤백)을 구성합니다.

    배포 그룹에서 고급 옵션 지정에 대한 자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하세요.

15. [Create deployment group]을 선택합니다.

Amazon ECS 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. Applications table(애플리케이션 테이블)에서 편집하려는 배포 그룹과 연결된 애플리케이션의 이름을 선택합니다.
4. 애플리케이션 페이지의 배포 그룹에서 편집하려는 배포 그룹의 이름을 선택합니다.
5. 애플리케이션 페이지의 배포 그룹 탭에서 배포 그룹 생성을 선택합니다. Amazon ECS 배포에 대한 배포 그룹을 만드는 데 필요한 항목에 대한 자세한 정보는 Amazon ECS 배포를 시작하기 전 단원을 참조하세요.
6. Deployment group name(배포 그룹 이름)에 배포 그룹을 설명하는 이름을 입력합니다.

> **Note**
>
> (배포 그룹 이름 및 배포 구성을 포함하여) 다른 배포 그룹에서 사용되는 것과 동일한 설정을 사용하려면 이 페이지에서 해당 설정을 선택합니다. 새 그룹과 기존 그룹의 이름이 같더라도 각 배포 그룹은 개별 애플리케이션과 연결되므로 CodeDeploy는 이러한 배포 그룹을 개별 배포 그룹으로 취급합니다.

7. 서비스 역할에서 Amazon ECS에 액세스할 수 있는 권한을 CodeDeploy에 부여하는 서비스 역할을 선택합니다. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하십시오.
8. 로드밸런서 이름에서 Amazon ECS 서비스에 트래픽을 공급하는 로드밸런서의 이름을 선택합니다.
9. 프로덕션 리스너 포트에서 해당 Amazon ECS 서비스에 서비스 프로덕션 트래픽을 공급하는 리스너의 프로토콜과 포트를 선택합니다.
10. (선택 사항) 테스트 리스너 포트에서 배포 중 해당 Amazon ECS 서비스의 대체 작업 세트에 트래픽을 공급하는 테스트 리스너의 프로토콜과 포트를 선택합니다. `AfterAllowTestTraffic` 후크 중 실행하는 AppSpec 파일의 Lambda 함수를 한 개 이상 지정할 수 있습니다. 이 함수는 확인 테스트를 실행할 수 있습니다. 확인 테스트가 실패하면 배포 롤백이 트리거됩니다. 확인 테스트가 성공하면 배포 수명 주기의 다음 후크인 `BeforeAllowTraffic`이 트리거됩니다. 테스트 리스너 포트를 지정하지 않으면 `AfterAllowTestTraffic` 후크 중 아무것도 수행되지 않습니다. 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션 단원을 참조하십시오.
11. 대상 그룹 1 이름과 대상 그룹 2 이름에서 배포 중 트래픽을 라우팅하는 데 사용되는 대상 그룹을 선택합니다. CodeDeploy는 한 대상 그룹을 Amazon ECS 서비스의 기존 작업 세트에 바인딩하고 다른 대상 그룹을 대체 작업 세트에 바인딩합니다. 자세한 내용은 Application Load Balancer의 대상 그룹 지정을 참조하세요.
12. 즉시 트래픽 다시 라우팅 또는 트래픽을 언제 다시 라우팅할지 지정을 선택하여 업데이트된 Amazon ECS 서비스에 트래픽을 언제 다시 라우팅할지를 지정합니다.

Reroute traffic immediately(즉시 트래픽 다시 라우팅)를 선택한 경우 대체 작업 세트가 프로비저닝된 후 배포가 트래픽을 자동으로 다시 라우팅합니다.

Specify when to reroute traffic(트래픽을 언제 다시 라우팅할지 지정)을 선택한 경우 대체 작업 세트가 성공적으로 프로비저닝된 후 대기할 일, 시간 및 분을 선택합니다. 이 대기 시간 동안 AppSpec 파일에 지정된 Lambda 함수의 유효성 테스트가 실행됩니다. 트래픽이 다시 라우팅되기 전에 대기 시간이 만료되면 배포 상태가 `Stopped`로 변경됩니다.

13. 기존 개정 종료에서 배포 성공 후 얼마 후에 Amazon ECS 서비스의 기존 작업 세트가 종료될지를 일, 시간, 분으로 선택합니다.
14. (선택 사항) 고급에서 배포에 포함하려는 모든 옵션(예: Amazon SNS 알림 트리거, Amazon CloudWatch 경보 또는 자동 롤백)을 구성합니다.

    자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하십시오.

Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정

배포 그룹에서 선택적 로드 밸런서를 지정하려는 블루/그린 배포 또는 인 플레이스 배포를 실행하려면 먼저 Elastic Load Balancing에 최소 하나의 Classic Load Balancer, Application Load Balancer, 또는 Network Load Balancer를 만들어야 합니다. 블루/그린 배포의 경우 해당 로드 밸런서를 사용하여 대체 환경을 구성하는 인스턴스를 등록합니다. 원본 환경의 인스턴스는 동일한 로드 밸런서에 선택적으로 등록할 수 있습니다. 인플레이스 배포의 경우 CodeDeploy에서 작업 중인 인스턴스를 등록 취소하고 작업이 완료되면 다시 등록하는 데 로드 밸런서가 사용됩니다.

CodeDeploy는 여러 로드 밸런서를 사용하여 Amazon EC2 인스턴스에 대한 블루/그린 및 인플레이스 배포를 지원합니다. 예를 들어, 200개의 Amazon EC2 인스턴스가 있고 그 중 100개는 2개 Classic Load Balancer에 등록되어 있고, 나머지 100개는 2개 Application Load Balancer의 4개 대상 그룹에 등록되어 있다고 가정해 보겠습니다. 이 시나리오에서 CodeDeploy를 사용하면 인스턴스가 2개 Classic Load Balancer, 2개 Application Load Balancer 및 4개 대상 그룹에 분산되어 있더라도 200개 인스턴스 모두에 블루/그린 및 인플레이스 배포를 수행할 수 있습니다.

CodeDeploy는 최대 10개의 Classic Load Balancer 및 10개의 대상 그룹으로 총 20개의 항목을 지원합니다.

하나 이상의 Classic Load Balancer를 구성하려면 Classic Load Balancer 사용 설명서의 자습서: Classic Load Balancer의 지침을 따릅니다. 다음 사항에 유의하세요.

- 2단계: 로드 밸런서 정의의 내부에서 LB 만들기에서 인스턴스를 생성할 때 선택한 바로 그 VPC를 선택합니다.
- 5단계: 로드 밸런서에 EC2 인스턴스 등록에서 배포 그룹의 현재 인스턴스(인 플레이스(in-place) 배포) 또는 원본 환경에서 지정한 인스턴스(블루/그린 배포)를 선택합니다.
- 7단계: 로드 밸런서 생성 및 확인에서 로드 밸런서의 DNS 주소를 기록해 둡니다.

  예를 들어 로드 밸런서의 이름을 `my-load-balancer`로 지정하면 DNS 주소는 `my-load-balancer-1234567890.us-east-2.elb.amazonaws.com`과 같은 형식으로 나타납니다.

하나 이상의 Application Load Balancer를 구성하려면 다음 주제 중 하나의 지침을 따릅니다.

- Application Load Balancer 생성
- 자습서:를 사용하여 Application Load Balancer 생성 AWS CLI

하나 이상의 Network Load Balancer를 구성하려면 다음 주제 중 하나의 지침을 따릅니다.

- Network Load Balancer 생성
- 자습서:를 사용하여 Network Load Balancer 생성 AWS CLI

CodeDeploy Amazon ECS 배포를 위한 로드 밸런서, 대상 그룹 및 리스너 설정

Amazon ECS 컴퓨팅 플랫폼을 사용하여 배포를 실행하려면 먼저 Application Load Balancer 또는 Network Load Balancer, 대상 그룹 두 개, 리스너 한 개 또는 두 개를 만들어야 합니다. 이 주제에서는 Application Load Balancer를 생성하는 방법을 보여줍니다. 자세한 내용은 Amazon ECS 배포를 시작하기 전 단원을 참조하십시오.

대상 그룹 중 하나는 트래픽을 Amazon ECS 애플리케이션의 원래 작업 세트로 보냅니다. 다른 대상 그룹은 트래픽을 대체 작업 세트로 보냅니다. 배포 중에 CodeDeploy는 대체 작업 세트를 만들고 원래 작업 세트에서 새 작업 세트로 트래픽을 다시 라우팅합니다. CodeDeploy는 각 작업 세트에 사용되는 대상 그룹을 결정합니다.

리스너는 로드 밸런서가 대상 그룹으로 트래픽을 보내기 위해 사용합니다. 프로덕션 리스너 한 개는 필수입니다. 확인 테스트를 실행하는 동안 대체 작업 세트로 트래픽을 보내는 선택적 테스트 리스너를 지정할 수 있습니다.

로드밸런서는 다른 가용 영역에 두 개의 퍼블릭 서브넷이 있는 VPC를 사용해야 합니다. 다음 단계에서는 기본 VPC를 확인하고, Amazon EC2 Application Load Balancer를 생성한 다음, 로드밸런서를 위한 대상 그룹을 두 개 생성합니다. 자세한 내용은 네트워크 로드밸런서의 대상 그룹 지정을 참조하세요.

기본 VPC, 퍼블릭 서브넷 및 보안 그룹 확인

이 주제에서는 Amazon EC2 Application Load Balancer, 대상 그룹 2개, Amazon ECS 배포 중에 사용할 수 있는 포트 2개를 생성하는 방법을 보여줍니다. 포트 중 하나는 선택 사항이며 배포 중 검증 테스트를 위해 테스트 포트로 트래픽을 보내는 경우에만 필요합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/vpc/ Amazon VPC 콘솔을 엽니다.
2. 사용할 기본 VPC를 확인합니다. 탐색 창에서 Your VPCs를 선택합니다. 기본 VPC 열에서 예로 표시된 VPC를 확인합니다. 이 VPC가 기본 VPC입니다. 이 VPC에는 사용하는 기본 서브넷이 포함됩니다.
3. 서브넷을 선택합니다. 기본 서브넷 열에 예로 표시되는 서브넷 두 개의 서브넷 ID를 기록해 둡니다. 로드 밸런서를 생성할 때 이러한 ID를 사용합니다
4. 각 서브넷을 선택한 다음 설명 탭을 선택합니다. 사용할 서브넷이 다른 가용 영역에 있는지 확인합니다.
5. 서브넷을 선택한 후 라우팅 테이블 탭을 선택합니다. 사용할 각 서브넷이 퍼블릭 서브넷인지 확인하려면, 인터넷 게이트웨이에 대한 링크가 있는 행이 라우팅 테이블에 포함되어 있는지 확인합니다.
6. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
7. 탐색 창에서 보안 그룹을 선택합니다.
8. 사용할 보안 그룹이 사용 가능한지 확인하고 그룹 ID(예: sg-abcd1234)를 기록해 둡니다. 로드 밸런서를 생성할 때 이 정보를 사용합니다

Amazon EC2 Application Load Balancer, 두 개의 대상 그룹 및 리스너 생성(콘솔)

Amazon EC2 콘솔을 사용하여 Amazon EC2 Application Load Balancer를 생성하려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/ec2/ Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 로드 밸런서를 클릭합니다.

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. (선택 사항) 로드 밸런스를 위한 두 번째 리스너 포트를 구성합니다. 이 포트에 서비스되는 테스트 트래픽을 사용하여 배포 검증 테스트를 실행할 수 있습니다.

   a. 로드 밸런서 프로토콜에서 리스너 추가를 선택합니다.
   b. 두 번째 리스너의 로드밸런서 프로토콜에서 HTTP를 선택합니다.
   c. 로드 밸런서 포트에 **8080**을 입력합니다.
9. 가용 영역의 VPC에서 기본 VPC를 선택한 후 사용할 기본 서브넷 두 개를 선택합니다.
10. 다음: 보안 설정 구성을 선택합니다.
11. 다음: 보안 그룹 구성(Next: Configure Security Groups)을 선택합니다.
12. 기존 보안 그룹 선택을 선택하고, 기본 보안 그룹을 선택한 다음, 해당 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 **80**을 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 **80**을 입력합니다.

6. VPC에서 기본 VPC를 선택합니다.
7. 생성(Create)을 선택합니다.

> **Note**
>
> Amazon ECS 배포를 실행하려면 로드밸런서에 대해 두 개의 대상 그룹을 생성해야 합니다. Amazon ECS 서비스를 생성할 때 대상 그룹 중 하나의 ARN을 사용합니다. 자세한 내용은 Amazon ECS 사용 설명서의 4단계: Amazon ECS 서비스 생성을 참조하세요.

Amazon EC2 Application Load Balancer, 두 개의 대상 그룹 및 리스너 생성(CLI)

AWS CLI을(를) 이용하여 Application Load Balancer 생성

1. create-load-balancer 명령을 사용하여 Application Load Balancer를 생성합니다. 동일한 가용 영역에 속하지 않은 서브넷 2개와 보안 그룹 하나를 지정합니다.

```
aws elbv2 create-load-balancer --name bluegreen-alb \
--subnets subnet-abcd1234 subnet-abcd5678 --security-groups sg-abcd1234 --
region us-east-1
```

출력에는 로드 밸런서의 Amazon 리소스 이름(ARN)이 다음 형식으로 포함됩니다.

```
arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642
```

2. create-target-group 명령을 사용하여 첫 번째 대상 그룹을 만듭니다. CodeDeploy는 이 대상 그룹의 트래픽을 서비스에 있는 원래 또는 대체 작업 세트로 라우팅합니다.

```
aws elbv2 create-target-group --name bluegreentarget1 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

출력에는 첫 번째 대상 그룹의 ARN이 다음 형식으로 포함됩니다.

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget1/209a844cd01825a4
```

3. create-target-group 명령을 사용하여 두 번째 대상 그룹을 만듭니다. CodeDeploy는 대상 그룹의 트래픽을 첫 번째 대상 그룹에서 서비스되지 않는 작업 세트로 라우팅합니다. 예를 들어, 첫 번째

대상 그룹이 원래 작업 세트로 트래픽을 라우팅하는 경우 이 대상 그룹은 대체 작업 세트로 트래픽을 라우팅합니다.

```
aws elbv2 create-target-group --name bluegreentarget2 --protocol HTTP --port 80 \
--target-type ip --vpc-id vpc-abcd1234 --region us-east-1
```

출력에는 두 번째 대상 그룹의 ARN이 다음 형식으로 포함됩니다.

```
arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup/
bluegreentarget2/209a844cd01825a4
```

4. create-listener 명령을 사용하여 프로덕션 트래픽을 포트 80에 전달하는 기본 규칙이 있는 리스너를 생성합니다.

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 80 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget1/209a844cd01825a4 --region us-east-1
```

출력에는 리스너의 ARN이 다음 형식으로 포함됩니다.

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

5. (선택 사항) create-listener 명령을 사용하여 테스트 트래픽을 포트 8080에 전달하는 기본 규칙이 있는 두 번째 리스너를 생성합니다. 이 포트에 서비스되는 테스트 트래픽을 사용하여 배포 검증 테스트를 실행할 수 있습니다.

```
aws elbv2 create-listener --load-balancer-arn
 arn:aws:elasticloadbalancing:region:aws_account_id:loadbalancer/app/bluegreen-alb/
e5ba62739c16e642 \
--protocol HTTP --port 8080 \
--default-actions
 Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:region:aws_account_id:targetgroup
bluegreentarget2/209a844cd01825a4 --region us-east-1
```

출력에는 리스너의 ARN이 다음 형식으로 포함됩니다.

```
arn:aws:elasticloadbalancing:region:aws_account_id:listener/app/bluegreen-alb/
e5ba62739c16e642/665750bec1b03bd4
```

배포 그룹 만들기(CLI)

AWS CLI 를 사용하여 배포 그룹을 생성하려면 create-deployment-group 명령을 호출하여 다음을 지정합니다.

- 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
- 배포 그룹의 이름입니다. 지정된 애플리케이션에 대해 이 이름의 배포 그룹이 생성됩니다. 배포 그룹은 애플리케이션 하나와만 연결할 수 있습니다.
- 배포 그룹에 포함되는 인스턴스를 식별하는 태그, 태그 그룹 또는 Amazon EC2 Auto Scaling 그룹 이름에 대한 정보.
- 다른 AWS 서비스와 상호 작용할 때 CodeDeploy가 AWS 계정을 대신하여 작업하도록 허용하는 서비스 역할의 Amazon 리소스 이름(ARN) 식별자입니다. 서비스 역할 ARN을 확인하려면 서비스 역할 ARN 확인(CLI) 단원을 참조하세요. 서비스 역할에 대한 자세한 내용은 IAM 사용 설명서의 역할 용어 및 개념을 참조하세요.
- 배포 그룹과 연결할 배포 유형(인 플레이스(in-place) 또는 블루/그린)에 대한 정보.
- (선택 사항) 기존 배포 구성의 이름. 배포 구성의 목록을 보려면 View Deployment Configuration Details 단원을 참조하세요. 지정하지 않은 경우 CodeDeploy에서는 기본 배포 구성을 사용합니다.
- (선택 사항) 배포 및 인스턴스 이벤트에 대한 알림을 Amazon Simple Notification Service 주제 구독자에게 푸시하는 트리거를 만드는 명령. 자세한 내용은 Monitoring Deployments with Amazon SNS Event Notifications 단원을 참조하십시오.
- (선택 사항) 경보에 지정된 지표가 정의된 임계값을 초과하거나 해당 임계값 미만인 경우 활성화되는 기존 CloudWatch 경보를 배포 그룹에 추가하는 명령.
- (선택 사항) 배포에 실패하거나 CloudWatch 경보가 활성화된 경우 알려진 최신 정상 개정으로 롤백하는 배포에 대한 명령.
- (선택 사항) Auto Scaling 스케일 인 이벤트 중에 수명 주기 이벤트 후크를 생성하기 위한 배포 명령입니다. 자세한 내용은 Amazon EC2 Auto Scaling에서 CodeDeploy를 사용하는 방식 단원을 참조하십시오.
- 인 플레이스(in-place) 배포의 경우:
  - (선택 사항) 배포 프로세스 중 인스턴스로 가는 트래픽을 관리하는 Elastic Load Balancing의 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer의 이름.

- 블루/그린 배포의 경우:
  - 블루/그린 배포 프로세스의 구성:
    - 대체 환경의 새 인스턴스를 프로비저닝하는 방법
    - 트래픽을 대체 환경으로 즉시 다시 라우팅할지, 아니면 지정된 시간 동안 대기한 후 트래픽을 수동으로 다시 라우팅할지 여부
    - 원본 환경의 인스턴스를 종료해야 할지 여부
  - 대체 환경에서 인스턴스를 등록하는 데 사용하는 Elastic Load Balancing의 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer의 이름.

> **Warning**
>
> 배포 그룹에서 오토 스케일링과 Elastic Load Balancing 로드 밸런서를 모두 구성하고 로드 밸런서를 오토 스케일링에 연결하려는 경우 이 배포 그룹에서 CodeDeploy 배포를 생성하기 전에 이 연결을 완료하는 것이 좋습니다. 배포를 생성한 후 연결을 완료하려고 하면 예기치 않게 로드 밸런서에서 모든 인스턴스의 등록이 취소될 수 있습니다.

CodeDeploy에서 배포 그룹 세부 정보 보기

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy APIs를 사용하여 애플리케이션과 연결된 모든 배포 그룹에 대한 세부 정보를 볼 수 있습니다.

주제

- 배포 그룹 세부 정보 보기(콘솔)
- 배포 그룹 세부 정보 보기(CLI)

배포 그룹 세부 정보 보기(콘솔)

CodeDeploy 콘솔을 사용하여 배포 그룹 세부 정보를 보려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. 애플리케이션 페이지에서 배포 그룹과 연결된 애플리케이션 이름을 선택합니다.

> **Note**
>
> 항목이 표시되지 않으면 올바른 리전이 선택되어 있는지 확인합니다. 탐색 모음의 리전 선택기에서 AWS 일반 참조의 리전 및 엔드포인트에 나열된 리전 중 하나를 선택합니다. CodeDeploy는 이러한 리전에서만 지원됩니다.

4. 개별 배포 그룹에 대한 세부 정보를 보려면 배포 그룹 탭에서 배포 그룹 이름을 선택합니다.

배포 그룹 세부 정보 보기(CLI)

AWS CLI 를 사용하여 배포 그룹 세부 정보를 보려면 `get-deployment-group` 명령 또는 `list-deployment-groups` 명령을 호출합니다.

단일 배포 그룹에 대한 세부 정보를 보려면 get-deployment-group 명령을 호출하여, 다음을 지정합니다.

- 배포 그룹과 연결된 애플리케이션 이름. 애플리케이션 이름을 가져오려면 list-applications 명령을 호출합니다.
- 배포 그룹 이름. 배포 그룹 이름을 가져오려면 list-deployment-groups 명령을 호출합니다.

배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출하여 배포 그룹과 연결된 애플리케이션 이름을 지정합니다. 애플리케이션 이름을 가져오려면 list-applications 명령을 호출합니다.

CodeDeploy에서 배포 그룹 설정 변경

CodeDeploy 콘솔 AWS CLI, 또는 CodeDeploy APIs를 사용하여 배포 그룹의 설정을 변경할 수 있습니다.

> **Warning**
>
> 배포 그룹이 아직 생성되지 않은 사용자 지정 배포 그룹을 사용하도록 하려면 다음 단계를 사용하지 마세요. 그 대신 Create a Deployment Configuration의 지침을 따르고 이 주제로 돌아옵니다. 배포 그룹이 아직 생성되지 않은 다른 서비스 역할을 사용하도록 하려면 이 단계를 사용하지 마세요. 최소한 2단계: CodeDeploy에 대한 서비스 역할 생성에서 설명된 권한을

CodeDeploy에 부여해야 합니다. 올바른 권한이 있는 서비스 역할을 만들고 구성하려면 2단계: CodeDeploy에 대한 서비스 역할 생성의 지침을 수행한 다음 이 주제로 돌아갑니다.

주제

- 배포 그룹 설정 변경(콘솔)
- 배포 그룹 설정 변경(CLI)

배포 그룹 설정 변경(콘솔)

CodeDeploy 콘솔을 사용하여 배포 그룹 설정을 변경하려면:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. 애플리케이션 목록에서 변경하려는 배포 그룹과 연결된 애플리케이션의 이름을 선택합니다.

   > **Note**
   >
   > 항목이 표시되지 않으면 올바른 리전이 선택되어 있는지 확인합니다. 탐색 모음의 리전 선택기에서 AWS 일반 참조의 리전 및 엔드포인트에 나열된 리전 중 하나를 선택합니다. CodeDeploy는 이러한 리전에서만 지원됩니다.

4. 배포 그룹 탭을 선택한 다음 변경하려는 배포 그룹의 이름을 선택합니다.
5. 배포 그룹 페이지에서 편집을 선택합니다.
6. 필요에 따라 배포 그룹 옵션을 편집합니다.

   배포 그룹 구성 요소에 대한 자세한 내용은 CodeDeploy에서 배포 그룹 만들기 단원을 참조하세요.

7. 변경 사항 저장을 선택합니다.

배포 그룹 설정 변경(CLI)

AWS CLI 를 사용하여 배포 그룹 설정을 변경하려면 update-deployment-group 명령을 호출하여 다음을 지정합니다.

- EC2/온프레미스 및 AWS Lambda 배포의 경우:
  - 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
  - 현재 배포 그룹 이름. 배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출합니다.
  - (선택 사항) 다른 배포 그룹 이름.
  - (선택 사항) 다른 서비스와 상호 작용할 때 CodeDeploy가 AWS 계정을 대신하여 작업하도록 허용하는 AWS 서비스 역할에 해당하는 다른 Amazon 리소스 이름(ARN)입니다. 서비스 역할 ARN을 확인하려면 서비스 역할 ARN 확인(CLI) 단원을 참조하세요. 서비스 역할에 대한 자세한 내용은 IAM 사용 설명서의 역할 용어 및 개념을 참조하세요.
  - (선택 사항) 배포 구성의 이름. 배포 구성의 목록을 보려면 View Deployment Configuration Details 단원을 참조하세요. (지정하지 않은 경우 CodeDeploy에서는 기본 배포 구성을 사용합니다.)
  - (선택 사항) 경보에 지정된 지표가 정의된 임계값을 초과하거나 해당 임계값 미만인 경우 활성화되는 하나 이상의 기존 CloudWatch 경보를 배포 그룹에 추가하는 명령.
  - (선택 사항) 배포에 실패하거나 CloudWatch 경보가 활성화된 경우 알려진 최신 정상 개정으로 롤백하는 배포에 대한 명령.
  - (선택 사항) Auto Scaling 스케일 인 이벤트 중에 수명 주기 이벤트 후크를 생성하기 위한 배포 명령입니다. 자세한 내용은 Amazon EC2 Auto Scaling에서 CodeDeploy를 사용하는 방식 단원을 참조하십시오.
  - (선택 사항) Amazon Simple Notification Service의 주제에 게시하는 트리거를 생성하거나 업데이트하여 해당 주제에 대한 구독자가 이 배포 그룹의 배포 및 인스턴스 이벤트에 대한 알림을 받을 수 있도록 하는 명령. 자세한 내용은 Monitoring Deployments with Amazon SNS Event Notifications 단원을 참조하세요.
- EC2/온프레미스 배포의 경우:
  - (선택 사항) 배포 그룹에 포함되는 인스턴스를 고유하게 식별하는 대체 태그 또는 태그 그룹.
  - (선택 사항) 배포 그룹에 추가할 대체 Amazon EC2 Auto Scaling 그룹의 이름.
- Amazon ECS 배포의 경우:
  - 배포할 Amazon ECS 서비스.
  - Application Load Balancer 또는 Network Load Balancer를 비롯한 로드 밸런서 정보, Amazon ECS 배포에 필요한 대상 그룹, 프로덕션 및 선택적 테스트 리스너 정보.

배포 그룹에 대한 고급 옵션 구성

배포 그룹을 만들거나 업데이트할 때 여러 옵션을 구성하여 해당 배포 그룹의 배포를 보다 효과적으로 제어하고 감독할 수 있습니다.

이 페이지의 정보를 사용하여 다음과 같은 주제에서 배포 그룹을 작업할 때 고급 옵션을 구성할 수 있습니다.

- CodeDeploy를 사용하여 애플리케이션 생성
- CodeDeploy에서 배포 그룹 만들기
- CodeDeploy에서 배포 그룹 설정 변경

Amazon SNS 알림 트리거: CodeDeploy 배포 그룹에 트리거를 추가하여 배포 또는 배포 그룹과 관련된 이벤트에 대한 알림을 수신할 수 있습니다. 이러한 알림은 트리거 작업의 일부로 만든 Amazon SNS 주제를 구독하는 수신자에게 전송됩니다.

이 트리거가 가리키는 Amazon SNS 주제를 이미 설정해야 하며, CodeDeploy 는 이 배포 그룹에서 주제에 게시할 권한이 있어야 합니다. 이러한 설치 단계를 아직 완료하지 않은 경우 나중에 배포 그룹에 트리거를 추가할 수 있습니다.

지금 트리거를 생성하여 이 애플리케이션에 대한 배포 그룹의 배포 이벤트에 대한 알림을 받으려면 트리거 생성을 선택합니다.

Amazon EC2 인스턴스에 배포하는 경우 인스턴스에 대한 알림을 생성하고 인스턴스에 대한 알림을 받을 수 있습니다.

자세한 내용은 Monitoring Deployments with Amazon SNS Event Notifications 단원을 참조하세요.

Amazon CloudWatch 경보 – 지정하는 기간 동안 단일 지표를 관찰하는 CloudWatch 경보를 생성하고 특정 기간 동안 지정된 임계값을 기준으로 지표의 값에 따라 하나 이상의 작업을 수행할 수 있습니다. Amazon EC2 배포의 경우 CodeDeploy 작업에 사용 중인 인스턴스 또는 Amazon EC2 Auto Scaling 그룹에 대한 경보를 생성할 수 있습니다. AWS Lambda 및 Amazon ECS 배포의 경우 Lambda 함수의 오류에 대한 경보를 생성할 수 있습니다.

Amazon CloudWatch 경보가 지표가 정의된 임계값 아래로 떨어지거나 초과되었음을 감지할 때 배포를 중지하도록 구성할 수 있습니다.

CloudWatch에서 경보를 이미 생성했어야 배포 그룹에 추가할 수 있습니다.

1. 배포 그룹에 경보 모니터링을 추가하려면 경보에서 경보 추가를 선택합니다.

2. 이 배포를 모니터링하도록 이미 설정한 CloudWatch 경보의 이름을 입력합니다.

   CloudWatch 경보는 CloudWatch에서 생성된 것과 정확히 동일하게 입력해야 합니다. 경보 목록을 보려면 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 열고 경보를 선택합니다.

추가 옵션:

- 추가한 경보를 고려하지 않고 배포를 계속하려면 경보 구성 무시를 선택합니다.

  이 옵션은 나중에 동일한 경보를 다시 추가하지 않고도 배포 그룹에 대한 경보 모니터링을 일시적으로 비활성화하려는 경우에 유용합니다.
- (선택 사항) CodeDeploy가 Amazon CloudWatch에서 경보 상태를 검색할 수 없는 경우 배포를 계속하려면 경보 상태를 사용할 수 없는 경우에도 배포 계속을 선택합니다.

> **Note**
>
> 이 옵션은 CodeDeploy API에 있는 AlarmConfiguration 객체의 ignorePollAlarmFailure에 해당합니다.

자세한 내용은 CodeDeploy에서 CloudWatch 경보를 사용하여 배포 모니터링 단원을 참조하세요.

자동 롤백: 배포에 실패한 경우 또는 지정한 모니터링 임계값에 도달한 경우 자동으로 롤백하도록 배포 그룹 또는 배포를 구성할 수 있습니다. 이 경우 마지막으로 알려진 정상 버전의 애플리케이션 개정이 배포됩니다. 콘솔을 사용하여 애플리케이션을 만들거나, 배포 그룹을 만들거나, 배포 그룹을 업데이트할 때 배포 그룹에 대한 설정 옵션을 구성할 수 있습니다. 새 배포를 만들 때 배포 그룹에 지정된 자동 롤백 구성을 재정의하도록 선택할 수도 있습니다.

- 문제가 발생하면 다음 중 하나 또는 둘 다를 선택하여 배포가 가장 최근에 알려진 정상 개정 버전으로 롤백되도록 할 수 있습니다.
  - 배포 실패 시 롤백. CodeDeploy는 마지막으로 알려진 정상 개정 버전을 새 배포로 재배포합니다.
  - 경보 임계값에 도달하면 롤백. 이전 단계에서 이 애플리케이션에 경보를 추가한 경우 CodeDeploy는 지정된 경보 중 하나 이상이 활성화될 때 마지막으로 알려진 정상 개정 버전을 재배포합니다.

> **Note**
>
> 롤백 구성을 일시적으로 무시하려면 롤백 사용 안 함을 선택합니다. 이 옵션은 나중에 동일한 구성을 다시 설정하지 않고도 자동 롤백을 일시적으로 비활성화하려는 경우에 유용합니다.

자세한 내용은 CodeDeploy를 사용하여 재배포 및 배포 롤백 단원을 참조하세요.

오래된 인스턴스에 대한 자동 업데이트: 특정 상황에서 CodeDeploy는 오래된 개정 버전의 애플리케이션을 Amazon EC2 인스턴스에 배포할 수 있습니다. 예를 들어 CodeDeploy 배포가 진행되는 동안 EC2 인스턴스가 ASG(Auto Scaling 그룹)로 시작되는 경우 해당 인스턴스는 최신 애플리케이션 대신 이전 개정 버전의 애플리케이션을 받게 됩니다. 이러한 인스턴스를 최신 상태로 유지하기 위해 CodeDeploy는 자동으로 후속 배포를 시작하여 오래된 인스턴스를 업데이트합니다. 오래된 EC2 인스턴스가 이전 개정에 남아 있도록이 기본 동작을 변경하려면 CodeDeploy API 또는 AWS Command Line Interface (CLI)를 통해 변경할 수 있습니다.

API를 통해 오래된 인스턴스의 자동 업데이트를 구성하려면 `UpdateDeploymentGroup` 또는 `CreateDeploymentGroup` 작업에 `outdatedInstancesStrategy` 요청 파라미터를 포함합니다. 자세한 내용은 AWS CodeDeploy API 참조를 참조하세요.

를 통해 자동 업데이트를 구성하려면 다음 명령 중 하나를 AWS CLI사용합니다.

```
aws deploy update-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

또는...

```
aws deploy create-deployment-group arguments --outdated-instances-strategy
UPDATE|IGNORE
```

... 여기서, *arguments*는 배포에 필요한 인수로 대체되고 *UPDATE|IGNORE*는 자동 업데이트를 활성화하는 `UPDATE` 또는 비활성화하는 `IGNORE`로 대체됩니다.

예제:

```
aws deploy update-deployment-group --application-name "MyApp" --current-
deployment-group-name "MyDG" --region us-east-1 --outdated-instances-
strategy IGNORE
```

이러한 AWS CLI 명령에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하세요.

CodeDeploy에서 배포 그룹 삭제

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy APIs를 사용하여 AWS 계정과 연결된 배포 그룹을 삭제할 수 있습니다.

> **Warning**
>
> 배포 그룹을 삭제하면 해당 배포 그룹과 연결된 모든 세부 정보도 CodeDeploy에서 삭제됩니다 배포 그룹에 사용된 인스턴스는 변경되지 않은 상태로 유지됩니다. 이 작업은 실행을 취소할 수 없습니다.

주제

- 배포 그룹 삭제(콘솔)
- 배포 그룹 삭제(CLI)

배포 그룹 삭제(콘솔)

CodeDeploy 콘솔을 사용하여 배포 그룹을 삭제하려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. 애플리케이션 목록에서 배포 그룹과 연결된 애플리케이션의 이름을 선택합니다.
4. 애플리케이션 세부 정보 페이지의 배포 그룹 탭에서 삭제할 배포 그룹의 이름을 선택합니다.
5. 배포 세부 정보 페이지에서 삭제를 선택합니다.

6. 메시지가 표시되면 삭제를 확인할 배포 그룹의 이름을 입력한 후 삭제를 선택합니다.

배포 그룹 삭제(CLI)

AWS CLI 를 사용하여 배포 그룹을 삭제하려면 delete-deployment-group 명령을 호출하여 다음을 지정합니다.

- 배포 그룹과 연결된 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
- 애플리케이션과 연결된 배포 그룹 이름. 배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출합니다.

CodeDeploy의 애플리케이션 개정 작업

CodeDeploy에서 개정에는 CodeDeploy가 인스턴스에 배포할 소스 파일 또는 CodeDeploy가 인스턴스에서 실행할 스크립트 버전이 포함됩니다.

개정을 계획하고 개정에 AppSpec 파일을 추가한 다음 개정을 Amazon S3 또는 GitHub에 푸시합니다. 개정을 푸시하고 나면 이제 버전을 배포할 수 있습니다.

주제

- CodeDeploy의 개정 계획
- CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가
- CodeDeploy 리포지토리 유형 선택
- Amazon S3에 CodeDeploy의 개정 푸시(EC2 온프레미스 배포 전용)
- CodeDeploy를 사용하여 애플리케이션 개정 세부 정보 보기
- CodeDeploy를 사용하여 Amazon S3에 애플리케이션 개정을 등록합니다.

CodeDeploy의 개정 계획

계획을 세우면 개정판을 훨씬 쉽게 배포할 수 있습니다.

AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼에 배포하는 경우 개정은 AppSpec 파일과 동일합니다. 다음 정보는 적용되지 않습니다. 자세한 내용은 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가을(를) 참조하세요.

EC2/온프레미스 컴퓨팅 플랫폼에 배포하려면 먼저 개발 시스템에 빈 루트 디렉토리(폴더)를 생성합니다. 여기서 인스턴스에 배포될 소스 파일(예: 텍스트 및 이진 파일, 실행 파일, 패키지 등) 또는 인스턴스에서 실행될 스크립트를 저장합니다.

예: Linux, macOS, Unix의 경우 `/tmp/` 루트 폴더 또는 Windows의 경우 `c:\temp` 루트 폴더

```
/tmp/ or c:\temp (root folder)
  |--content (subfolder)
  |    |--myTextFile.txt
  |    |--mySourceFile.rb
  |    |--myExecutableFile.exe
  |    |--myInstallerFile.msi
  |    |--myPackage.rpm
  |    |--myImageFile.png
```

```
  |--scripts (subfolder)
  |    |--myShellScript.sh
  |    |--myBatchScript.bat
  |    |--myPowerShellScript.ps1
  |--appspec.yml
```

루트 폴더에는 다음과 같이 애플리케이션 사양 파일(AppSpec 파일)도 포함되어야 합니다. 자세한 내용은 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 섹션을 참조하세요.

CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가

이 주제에서는 AppSpec 파일을 배포에 추가하는 방법을 살펴봅니다. 또한 AWS Lambda 및 EC2/온프레미스 배포를 위한 AppSpec 파일을 생성하는 템플릿도 포함되어 있습니다.

주제

- Amazon ECS 배포용 AppSpec 파일 추가
- AWS Lambda 배포를 위한 AppSpec 파일 추가
- EC2 온프레미스 배포용 AppSpec 파일 추가

Amazon ECS 배포용 AppSpec 파일 추가

Amazon ECS 컴퓨팅 플랫폼에 배포하는 경우:

- AppSpec 파일은 배포에 사용되는 Amazon ECS 작업 정의, 트래픽을 라우팅하는 데 사용되는 포트 매핑과 컨테이너 이름, 배포 수명 주기 이벤트 후 실행되는 Lambda 함수 옵션을 지정합니다.
- 개정은 AppSpec 파일과 동일합니다.
- APSpec 파일은 JSON 또는 YAML을 사용하여 작성할 수 있습니다.
- AppSpec 파일은 텍스트 파일로 저장하거나 배포를 만들 때 콘솔에 직접 입력할 수 있습니다. 자세한 내용은 Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(콘솔) 단원을 참조하세요.

AppSpec 파일을 만들려면

1. JSON 또는 YAML 템플릿을 텍스트 편집기에 복사하거나 콘솔의 AppSpec 편집기에 복사합니다.
2. 필요에 따라 템플릿을 수정합니다.
3. JSON 또는 YAML 유효성 검사기를 사용하여 AppSpec 파일을 검증합니다. AppSpec 편집기를 사용하는 경우 배포 만들기를 선택할 때 파일이 검증됩니다.

4. 텍스트 편집기를 사용하는 경우 파일을 저장합니다. 를 사용하여 배포 AWS CLI 를 생성하는 경우 AppSpec 파일이 하드 드라이브 또는 Amazon S3 버킷에 있는 경우 해당 파일을 참조하세요. 콘솔을 사용하는 경우 AppSpec 파일을 Amazon S3로 푸시해야 합니다.

지침이 포함된 Amazon ECS 배포용 YAML AppSpec 파일 템플릿

다음은 사용 가능한 모든 옵션이 포함된 Amazon ECS 배포용 AppSpec 파일의 YAML 템플릿입니다. hooks 섹션에서 사용할 수명 주기 이벤트에 대한 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션을(를) 참조하세요.

```
# This is an appspec.yml template file for use with an Amazon ECS deployment in
 CodeDeploy.
# The lines in this template that start with the hashtag are
#   comments that can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section, you must specify the following: the Amazon ECS service,
 task definition name,
# and the name and port of the load balancer to route traffic,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "" # Specify the ARN of your task definition
 (arn:aws:ecs:region:account-id:task-definition/task-definition-family-name:task-
definition-revision-number)
        LoadBalancerInfo:
          ContainerName: "" # Specify the name of your Amazon ECS application's
 container
          ContainerPort: "" # Specify the port for your container where traffic
 reroutes
# Optional properties
        PlatformVersion: "" # Specify the version of your Amazon ECS Service
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["",""] # Specify one or more comma-separated subnets in your
 Amazon ECS service
```

```
            SecurityGroups: ["",""] # Specify one or more comma-separated security
 groups in your Amazon ECS service
            AssignPublicIp: "" # Specify "ENABLED" or "DISABLED"
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event.
Hooks:
# Hooks for Amazon ECS deployments are:
    - BeforeInstall: "" # Specify a Lambda function name or ARN
    - AfterInstall: "" # Specify a Lambda function name or ARN
    - AfterAllowTestTraffic: "" # Specify a Lambda function name or ARN
    - BeforeAllowTraffic: "" # Specify a Lambda function name or ARN
    - AfterAllowTraffic: "" # Specify a Lambda function name or ARN
```

Amazon ECS 배포용 JSON AppSpec 파일 템플릿

다음은 사용 가능한 모든 옵션이 포함된 Amazon ECS 배포용 AppSpec 파일의 JSON 템플릿입니다. 템플릿 지침은 이전 섹션의 YAML 버전에 있는 설명을 참조하세요. hooks 섹션에서 사용할 수명 주기 이벤트에 대한 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션을(를) 참조하세요.

```
{
 "version": 0.0,
 "Resources": [
  {
   "TargetService": {
    "Type": "AWS::ECS::Service",
    "Properties": {
       "TaskDefinition": "",
       "LoadBalancerInfo": {
        "ContainerName": "",
        "ContainerPort":
       },
       "PlatformVersion": "",
       "NetworkConfiguration": {
        "AwsvpcConfiguration": {
         "Subnets": [
          "",
          ""
         ],
         "SecurityGroups": [
          "",
          ""
         ],
         "AssignPublicIp": ""
```

```
        }
       }
      }
    }
   }
  ],
  "Hooks": [
   {
    "BeforeInstall": ""
   },
   {
    "AfterInstall": ""
   },
   {
    "AfterAllowTestTraffic": ""
   },
   {
    "BeforeAllowTraffic": ""
   },
   {
    "AfterAllowTraffic": ""
   }
  ]
}
```

AWS Lambda 배포를 위한 AppSpec 파일 추가

AWS Lambda 컴퓨팅 플랫폼에 배포하는 경우:

- AppSpec 파일에는 배포되고 배포 유효성 검사에 사용할 Lambda 함수에 대한 지침이 포함되어 있습니다.
- 개정은 AppSpec 파일과 동일합니다.
- APSpec 파일은 JSON 또는 YAML을 사용하여 작성할 수 있습니다.
- AppSpec 파일을 텍스트 파일로 저장할 수도 있고 배포 생성 시 콘솔 AppSpec 편집기에 직접 입력할 수도 있습니다. 자세한 내용은 AWS Lambda 컴퓨팅 플랫폼 배포 생성(콘솔) 단원을 참조하세요.

AppSpec 파일을 만들려면:

1. JSON 또는 YAML 템플릿을 텍스트 편집기에 복사하거나 콘솔의 AppSpec 편집기에 복사합니다.
2. 필요에 따라 템플릿을 수정합니다.

3. JSON 또는 YAML 유효성 검사기를 사용하여 AppSpec 파일을 검증합니다. AppSpec 편집기를 사용하는 경우 배포 만들기를 선택할 때 파일이 검증됩니다.
4. 텍스트 편집기를 사용하는 경우 파일을 저장합니다. 를 사용하여 배포 AWS CLI 를 생성하는 경우 AppSpec 파일이 하드 드라이브 또는 Amazon S3 버킷에 있는 경우 해당 파일을 참조하세요. 콘솔을 사용하는 경우 AppSpec 파일을 Amazon S3로 푸시해야 합니다.

지침이 포함된 AWS Lambda 배포용 YAML AppSpec 파일 템플릿

후크 섹션에서 사용할 수명 주기 이벤트에 대한 자세한 내용은 AWS Lambda 배포를 위한 AppSpec 'hooks' 섹션을(를) 참조하세요.

```
# This is an appspec.yml template file for use with an AWS Lambda deployment in
 CodeDeploy.
# The lines in this template starting with the hashtag symbol are
#   instructional comments and can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# In the Resources section specify the name, alias,
# target version, and (optional) the current version of your AWS Lambda function.
Resources:
  - MyFunction: # Replace "MyFunction" with the name of your Lambda function
      Type: AWS::Lambda::Function
      Properties:
        Name: "" # Specify the name of your Lambda function
        Alias: "" # Specify the alias for your Lambda function
        CurrentVersion: "" # Specify the current version of your Lambda function
        TargetVersion: "" # Specify the version of your Lambda function to deploy
# (Optional) In the Hooks section, specify a validation Lambda function to run during
# a lifecycle event. Replace "LifeCycleEvent" with BeforeAllowTraffic
# or AfterAllowTraffic.
Hooks:
    - LifeCycleEvent: "" # Specify a Lambda validation function between double-quotes.
```

AWS Lambda 배포 템플릿용 JSON AppSpec 파일

다음 템플릿에서 “MyFunction”을 AWS Lambda 함수 이름으로 바꿉니다. 후크 옵션 섹션에서 수명 주기 이벤트를 BeforeAllowTraffic 또는 AfterAllowTraffic으로 바꿉니다.

후크 섹션에서 사용할 수명 주기 이벤트에 대한 자세한 내용은 AWS Lambda 배포를 위한 AppSpec 'hooks' 섹션을(를) 참조하세요.

```
{
  "version": 0.0,
  "Resources": [{
   "MyFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "",
     "Alias": "",
     "CurrentVersion": "",
     "TargetVersion": ""
    }
   }
  }],
  "Hooks": [{
    "LifeCycleEvent": ""
   }
  ]
 }
```

EC2 온프레미스 배포용 AppSpec 파일 추가

AppSpec 파일이 없으면 CodeDeploy는 애플리케이션 개정의 소스 파일을 대상에 매핑하거나 배포용 스크립트를 EC2/온프레미스 컴퓨팅 플랫폼에 실행할 수 없습니다.

각 개정에는 AppSpec 파일을 하나만 포함해야 합니다.

개정에 AppSpec 파일을 추가하려면:

1. 템플릿을 텍스트 편집기에 복사합니다.
2. 필요에 따라 템플릿을 수정합니다.
3. YAML 유효성 검사기를 사용하여 AppSpec 파일을 검증합니다.
4. 파일을 개정의 루트 디렉터리에 `appspec.yml`로 저장합니다.
5. AppSpec 파일을 루트 디렉터리에 저장했는지 확인하려면 다음 명령 중 하나를 실행합니다.

   - Linux, macOS 또는 Unix의 경우:

     ```
     find /path/to/root/directory -name appspec.yml
     ```

AppSpec 파일을 찾을 수 없으면 출력이 없습니다.

- Windows의 경우:

```
dir path\to\root\directory\appspec.yml
```

AppSpec 파일이 저장되지 않은 경우 파일을 찾을 수 없음 오류가 표시됩니다.

6. 개정을 Amazon S3 또는 GitHub에 푸시합니다.

지침은 Amazon S3에 CodeDeploy의 개정 푸시(EC2 온프레미스 배포 전용) 단원을 참조하세요.

지침이 포함된 EC2 온프레미스 배포용 AppSpec 파일 템플릿

> **Note**
>
> Windows Server 인스턴스에 배포하는 기능은 `runas` 요소를 지원하지 않습니다. Windows Server 인스턴스를 배포하고 있다면 이 요소를 AppSpec 파일에 포함하지 마세요.

```
# This is an appspec.yml template file for use with an EC2/On-Premises deployment in
 CodeDeploy.
# The lines in this template starting with the hashtag symbol are
#   instructional comments and can be safely left in the file or
#   ignored.
# For help completing this file, see the "AppSpec File Reference" in the
#   "CodeDeploy User Guide" at
#   https://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html
version: 0.0
# Specify "os: linux" if this revision targets Amazon Linux,
#   Red Hat Enterprise Linux (RHEL), or Ubuntu Server
#   instances.
# Specify "os: windows" if this revision targets Windows Server instances.
# (You cannot specify both "os: linux" and "os: windows".)
os: linux
# os: windows
# During the Install deployment lifecycle event (which occurs between the
#   BeforeInstall and AfterInstall events), copy the specified files
#   in "source" starting from the root of the revision's file bundle
#   to "destination" on the Amazon EC2 instance.
# Specify multiple "source" and "destination" pairs if you want to copy
```

```
#   from multiple sources or to multiple destinations.
# If you are not copying any files to the Amazon EC2 instance, then remove the
#   "files" section altogether. A blank or incomplete "files" section
#   may cause associated deployments to fail.
files:
  - source:
    destination:
  - source:
    destination:
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
#   you can specify a "permissions"
#   section here that describes special permissions to apply to the files
#   in the "files" section as they are being copied over to
#   the Amazon EC2 instance.
#   For more information, see the documentation.
# If you are deploying to Windows Server instances,
#   then remove the
#   "permissions" section altogether. A blank or incomplete "permissions"
#   section may cause associated deployments to fail.
permissions:
  - object:
    pattern:
    except:
    owner:
    group:
    mode:
    acls:
      -
    context:
      user:
      type:
      range:
    type:
      -
# If you are not running any commands on the Amazon EC2 instance, then remove
#   the "hooks" section altogether. A blank or incomplete "hooks" section
#   may cause associated deployments to fail.
hooks:
# For each deployment lifecycle event, specify multiple "location" entries
#   if you want to run multiple scripts during that event.
# You can specify "timeout" as the number of seconds to wait until failing the
 deployment
#   if the specified scripts do not run within the specified time limit for the
#   specified event. For example, 900 seconds is 15 minutes. If not specified,
```

```
#   the default is 1800 seconds (30 minutes).
#   Note that the maximum amount of time that all scripts must finish executing
#   for each individual deployment lifecycle event is 3600 seconds (1 hour).
#   Otherwise, the deployment will stop and CodeDeploy will consider the deployment
#   to have failed to the Amazon EC2 instance. Make sure that the total number of
 seconds
#   that are specified in "timeout" for all scripts in each individual deployment
#   lifecycle event does not exceed a combined 3600 seconds (1 hour).
# For deployments to Amazon Linux, Ubuntu Server, or RHEL instances,
#   you can specify "runas" in an event to
#   run as the specified user. For more information, see the documentation.
#   If you are deploying to Windows Server instances,
#   remove "runas" altogether.
# If you do not want to run any commands during a particular deployment
#   lifecycle event, remove that event declaration altogether. Blank or
#   incomplete event declarations may cause associated deployments to fail.
# During the ApplicationStop deployment lifecycle event, run the commands
#   in the script specified in "location" starting from the root of the
#   revision's file bundle.
  ApplicationStop:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the BeforeInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  BeforeInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the AfterInstall deployment lifecycle event, run the commands
#   in the script specified in "location".
  AfterInstall:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
```

```
# During the ApplicationStart deployment lifecycle event, run the commands
#   in the script specified in "location".
  ApplicationStart:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
# During the ValidateService deployment lifecycle event, run the commands
#   in the script specified in "location".
  ValidateService:
    - location:
      timeout:
      runas:
    - location:
      timeout:
      runas:
```

CodeDeploy 리포지토리 유형 선택

CodeDeploy에서 필요한 파일의 저장소를 리포지토리라고 합니다. 리포지토리 사용은 해당 배포에서 어떤 컴퓨팅 플랫폼을 사용하는지에 따라 다릅니다.

- EC2/온프레미스: 하나 이상의 인스턴스에 애플리케이션 코드를 배포하려면 코드를 아카이브 파일로 번들링하여 배포 프로세스 중에 액세스할 수 있는 리포지토리에 배치해야 합니다. 배포형 콘텐츠와 AppSpec 파일을 아카이브 파일로 번들링한 다음 CodeDeploy에서 지원하는 리포지토리 유형 중 하나로 업로드합니다.
- AWS Lambda 및 Amazon ECS: 배포에는 다음 방법 중 하나로 배포 중에 액세스할 수 있는 AppSpec 파일이 필요합니다.
  - Amazon S3 버킷에서 액세스.
  - 콘솔의 AppSpec 편집기에 직접 입력된 텍스트에서 액세스. 자세한 내용은 AWS Lambda 컴퓨팅 플랫폼 배포 생성(콘솔) 및 Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(콘솔) 단원을 참조하세요.
  - 를 사용하는 경우 하드 드라이브 또는 네트워크 드라이브에 있는 AppSpec 파일을 참조할 AWS CLI수 있습니다. 자세한 내용은 AWS Lambda 컴퓨팅 플랫폼 배포 생성(CLI) 및 Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(CLI) 단원을 참조하세요.

CodeDeploy는 현재 다음 리포지토리 유형을 지원합니다.

리포지토리 유형	리포지토리 세부 정보	지원되는 컴퓨팅 플랫폼
Amazon S3	Amazon Simple Storage Service(Amazon S3)는 안전하고 확장 가능한 객체 스토리지를 위한 AWS 솔루션입니다. Amazon S3는 데이터를 버킷 내에 객체로 저장합니다. 객체는 파일과 해당 파일을 설명하는 메타데이터(선택 사항)로 구성됩니다. Amazon S3에 객체를 저장하려면 파일을 버킷에 업로드합니다. 파일을 업로드하면 객체에 대해 권한 및 메타데이터를 설정할 수 있습니다. 자세히 알아보기: • Amazon S3에서 버킷 생성 • Amazon S3에 CodeDeploy의 개정 푸시(EC2 온프레미스 배포 전용) • CodeDeploy를 사용하여 Amazon S3에서 자동으로 배포	다음 컴퓨팅 플랫폼을 사용하는 배포는 Amazon S3 버킷에 개정을 저장할 수 있습니다. • EC2/온프레미스 • AWS Lambda • Amazon ECS
GitHub	GitHub 리포지토리에 애플리케이션 개정을 저장할 수 있습니다. GitHub 리포지토리의 소스 코드가 변경될 때마다 GitHub 리포지토리에서 배포를 트리거할 수 있습니다. 자세히 알아보기:	EC2/온프레미스 배포만 GitHub 리포지토리에 개정을 저장할 수 있습니다.

	• GitHub와 CodeDeploy 통합 • 튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포	
Bitbucket	Bitbucket 파이프라인에서 CodeDeploy 파이프를 사용하여 EC2 인스턴스의 배포 그룹에 코드를 배포할 수 있습니다. Bitbucket 파이프라인은 Bitbucket 배포를 포함한 지속적인 통합 및 지속적인 배포(CI/CD) 기능을 제공합니다. CodeDeploy 파이프는 먼저 사용자가 지정한 S3 버킷에 Artifact를 푸시한 다음 버킷에서 코드 Artifact를 배포합니다. 자세히 알아보기: • Bitbucket용 CodeDeploy 파이프 참조	EC2/온프레미스 배포만 BitBucket 리포지토리에 개정을 저장할 수 있습니다.

> **Note**
>
> AWS Lambda 배포는 Amazon S3 리포지토리에서만 작동합니다.

Amazon S3에 CodeDeploy의 개정 푸시(EC2 온프레미스 배포 전용)

CodeDeploy의 개정 계획에서 설명한 대로 개정을 계획하고 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가에 설명한 대로 개정에 AppSpec 파일을 추가하면 구성 요소 파일을 번들로 묶고 개정을 Amazon S3로 푸시할 준비가 되었습니다. Amazon EC2 인스턴스에 배포하는 경우 개정을 푸시한 후 CodeDeploy를 사용하여 Amazon S3 에서 인스턴스로 개정을 배포할 수 있습니다.

> **Note**
>
> CodeDeploy를 사용하여 GitHub에 푸시된 개정을 배포할 수도 있습니다. 자세한 내용은 GitHub 문서를 참조하세요.

AWS CLI을(를) 설정하기 위해 CodeDeploy 시작하기의 지침을 이미 따른 것으로 가정합니다. 이는 특히 추후에 설명되는 push 명령을 호출하는 데 중요합니다.

Amazon S3 버킷이 있는지 확인합니다. 버킷 생성의 지침을 따릅니다.

Amazon EC2 인스턴스에 배포하는 경우 대상 Amazon S3 버킷을 생성하거나 대상 인스턴스와 동일한 리전에 존재해야 합니다. 예를 들어 미국 동부(버지니아 북부) 리전에 있는 일부 인스턴스와 미국 서부(오레곤) 리전에 있는 다른 인스턴스에 개정을 배포하려면 미국 동부(버지니아 북부) 리전에 하나의 버킷과 개정 사본이 있어야 하고 미국 서부(오레곤) 리전에 또 다른 버킷과 동일 개정 사본이 있어야 합니다. 이 시나리오에서는 두 개의 개별 배포를 생성해야 합니다. 하나는 미국 동부(버지니아 북부) 리전에 있고 다른 하나는 미국 서부(오레곤) 리전에 있습니다. 개정이 리전과 버킷 모두에서 동일하더라도 마찬가지입니다.

또한 Amazon S3 버킷으로 업로드할 수 있는 권한이 있어야 합니다. Amazon S3 버킷 정책을 통해 이러한 권한을 지정할 수 있습니다. 예를 들어 다음 Amazon S3 버킷 정책에서 와일드카드 문자(*)를 사용하면 AWS 계정이 라는 Amazon S3 버킷의 모든 디렉터리에 파일을 111122223333 업로드할 수 있습니다amzn-s3-demo-bucket.

```
{
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "111122223333"
                ]
            }
        }
    ]
}
```

AWS 계정 ID를 보려면 AWS 계정 ID 찾기를 참조하세요.

Amazon S3 버킷 정책을 생성하고 연결하는 방법은 버킷 정책 예제를 참조하세요.

push 명령을 호출하는 사용자는 최소한 각 대상 Amazon S3 버킷에 개정을 업로드할 수 있는 권한이 있어야 합니다. 예를 들어 다음 정책은 사용자가 `amzn-s3-demo-bucket`이라는 Amazon S3 버킷의 어디에나 개정을 업로드할 수 있도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

IAM 정책을 생성하고 연결하는 방법을 알아보려면 정책 작업 단원을 참조하세요.

를 사용하여 개정 푸시 AWS CLI

> **Note**
>
> push 명령은 애플리케이션 Artifact와 AppSpec 파일을 개정으로 번들링합니다. 이 개정의 파일 형식은 압축된 ZIP 파일입니다. 명령은 각각 JSON 형식 또는 YAML 형식 AppSpec 파일인 개정을 예상하기 때문에 AWS Lambda 또는 Amazon ECS 배포에 사용할 수 없습니다.

push 명령을 호출하여 배포에 대한 개정을 번들링하고 푸시합니다. 해당 파라미터는 다음과 같습니다.

- --application-name: (문자열) 필수. 애플리케이션 개정과 연관된 CodeDeploy 애플리케이션의 이름.

- --s3-location: (문자열) 필수. Amazon S3에 업로드할 애플리케이션 개정의 위치에 대한 정보. Amazon S3 버킷과 키를 지정해야 합니다. 키는 개정의 이름입니다. CodeDeploy는 업로드되기 전에 콘텐츠를 압축합니다. `s3://amzn-s3-demo-bucket/your-key.zip` 형식을 사용합니다.
- --ignore-hidden-files 또는 --no-ignore-hidden-files: (부울) 선택 사항. `--no-ignore-hidden-files` 플래그(기본값)를 사용하여 숨겨진 파일을 번들링하고 Amazon S3로 업로드합니다. `--ignore-hidden-files` 플래그를 사용하여 숨겨진 파일을 번들링하지 않고 Amazon S3로 업로드합니다.
- --source (문자열) 선택 사항. 배포할 콘텐츠와, 압축하여 Amazon S3로 업로드될 개발 시스템 상의 AppSpec 파일의 위치. 이 위치는 현재 디렉터리를 기준으로 한 상대 경로로 지정됩니다. 상대 경로가 지정되지 않았거나 경로에 단일 마침표('.')를 사용하는 경우 현재 디렉터리가 사용됩니다.
- --description (문자열) 선택 사항. 애플리케이션 개정을 요약하는 설명. 지정하지 않으면 기본 문자열 'Uploaded by AWS CLI 'time' UTC'가 사용됩니다. 여기서 'time'은 협정 세계시(UTC)의 현재 시스템 시간입니다.

를 사용하여 Amazon EC2 배포에 대한 개정을 푸시 AWS CLI 할 수 있습니다. 예를 들어 푸시 명령은 다음과 같습니다.

Linux, macOS 또는 Unix:

```
aws deploy push \
  --application-name WordPress_App \
  --description "This is a revision for the application WordPress_App" \
  --ignore-hidden-files \
  --s3-location s3://amzn-s3-demo-bucket/WordPressApp.zip \
  --source .
```

Windows:

```
aws deploy push --application-name WordPress_App --description "This is a revision for
 the application WordPress_App" --ignore-hidden-files --s3-location s3://amzn-s3-demo-
bucket/WordPressApp.zip --source .
```

이 명령은 다음 작업을 수행합니다.

- 번들링된 파일을 `WordPress_App`이라는 애플리케이션과 연결합니다.
- 개정에 설명을 첨부합니다.
- 숨겨진 파일을 무시합니다.

- 개정을 `WordPressApp.zip`으로 지정하고 `amzn-s3-demo-bucket`이라는 버킷으로 푸시합니다.
- 루트 디렉토리의 모든 파일을 개정으로 번들링합니다.

푸시가 성공하면 AWS CLI 또는 CodeDeploy 콘솔을 사용하여 Amazon S3에서 개정을 배포할 수 있습니다. 를 사용하여이 개정을 배포하려면 AWS CLI:

Linux, macOS 또는 Unix:

```
aws deploy create-deployment \
  --application-name WordPress_App \
  --deployment-config-name your-deployment-config-name \
  --deployment-group-name your-deployment-group-name \
  --s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip
```

Windows:

```
aws deploy create-deployment --application-name WordPress_App --deployment-config-
name your-deployment-config-name --deployment-group-name your-deployment-group-name --
s3-location bucket=amzn-s3-demo-bucket,key=WordPressApp.zip,bundleType=zip
```

자세한 내용은 CodeDeploy에서 배포 만들기 섹션을 참조하세요

CodeDeploy를 사용하여 애플리케이션 개정 세부 정보 보기

CodeDeploy 콘솔 AWS CLI, 또는 CodeDeploy APIs를 사용하여 지정된 애플리케이션의 AWS 계정에 등록된 모든 애플리케이션 개정에 대한 세부 정보를 볼 수 있습니다.

개정 등록에 대한 자세한 내용은 CodeDeploy를 사용하여 Amazon S3에 애플리케이션 개정을 등록합니다. 단원을 참조하세요.

주제

- 애플리케이션 개정 세부 정보 보기(콘솔)
- 애플리케이션 개정 세부 정보 보기(CLI)

애플리케이션 개정 세부 정보 보기(콘솔)

애플리케이션 개정 세부 정보를 보려면:

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.

> **Note**
>
> 항목이 표시되지 않으면 올바른 리전이 선택되어 있는지 확인합니다. 탐색 모음의 리전 선택기에서 AWS 일반 참조의 리전 및 엔드포인트에 나열된 리전 중 하나를 선택합니다. CodeDeploy는 이러한 지역에서만 지원됩니다.

3. 보려는 개정이 있는 애플리케이션의 이름을 선택합니다.
4. 애플리케이션 세부 정보 페이지에서 개정 탭을 클릭하고 애플리케이션에 대해 등록된 개정 목록을 검토합니다. 개정을 선택한 다음 세부 정보 보기를 선택합니다.

애플리케이션 개정 세부 정보 보기(CLI)

AWS CLI 를 사용하여 애플리케이션 개정을 보려면 get-application-revision 명령 또는 list-application-revisions 명령을 호출합니다.

> **Note**
>
> GitHub에 대한 참조는 EC2/온프레미스 배포에 대한 배포에만 적용됩니다. AWS Lambda 배포에 대한 개정은 GitHub에서 작동하지 않습니다.

단일 애플리케이션 개정에 대한 세부 정보를 보려면 get-application-revision 명령을 호출하고 다음을 지정합니다.

- 애플리케이션 이름. 애플리케이션 이름을 가져오려면 list-applications 명령을 호출합니다.
- GitHub에 저장된 개정의 경우, 리포지토리에 푸시된 애플리케이션 개정을 참조하는 커밋의 ID와 GitHub 리포지토리 이름.

- Amazon S3에 저장된 개정의 경우, 개정이 포함된 Amazon S3 버킷 이름, 업로드된 아카이브 파일의 이름 및 파일 유형, (선택적으로) 아카이브 파일의 Amazon S3 버전 식별자 및 ETag. register-application-revision 호출 중에 버전 식별자, ETag 또는 둘 다 지정된 경우 여기에 지정해야 합니다.

단일 애플리케이션 개정에 대한 세부 정보를 보려면 list-application-revisions 명령을 호출하고 다음을 지정합니다.

- 애플리케이션 이름. 애플리케이션 이름을 가져오려면 list-applications 명령을 호출합니다.
- (선택 사항) Amazon S3 애플리케이션 개정에 대한 세부 정보만 보려면 개정이 포함된 Amazon S3 버킷 이름.
- (선택 사항) Amazon S3 애플리케이션 개정에 대한 세부 정보만 보려면 Amazon S3 애플리케이션 개정으로 검색을 제한하는 접두사 문자열. (지정하지 않으면 CodeDeploy에서 일치하는 모든 Amazon S3 애플리케이션 개정을 나열합니다.)
- (선택 사항) 각 개정이 배포 그룹의 대상 개정인지 여부에 따라 개정 세부 정보를 나열할지 여부를 지정합니다. (지정하지 않으면 CodeDeploy에서 일치하는 모든 개정을 나열합니다.)
- (선택 사항) 개정 세부 정보 목록을 정렬할 열 이름과 순서. (지정하지 않으면 CodeDeploy에서 결과를 임의의 순서로 나열합니다.)

모든 개정 또는 Amazon S3에 저장된 개정만 나열할 수 있습니다. GitHub에 저장된 개정만 나열할 수는 없습니다.

CodeDeploy를 사용하여 Amazon S3에 애플리케이션 개정을 등록합니다.

이미 push 명령을 호출하여 애플리케이션 개정을 Amazon S3으로 푸시하는 경우 개정을 등록할 필요가 없습니다. 그러나 다른 방법을 통해 Amazon S3에 개정을 업로드하고 해당 개정을 CodeDeploy 콘솔에 또는 AWS CLI을(를) 통해 표시하려면 먼저 이러한 단계에 따라 개정을 등록하세요.

애플리케이션 개정을 GitHub 리포지토리로 푸시하고 개정을 CodeDeploy 콘솔 또는를 통해 표시하려면 다음 단계도 따라야 AWS CLI합니다.

AWS CLI 또는 CodeDeploy APIs만 사용하여 Amazon S3 또는 GitHub에 애플리케이션 개정을 등록할 수 있습니다.

주제

- CodeDeploy(CLI)를 사용하여 Amazon S3에 개정을 등록합니다.

- CodeDeploy(CLI)를 사용하여 GitHub에 개정 등록

CodeDeploy(CLI)를 사용하여 Amazon S3에 개정을 등록합니다.

1. Amazon S3에 개정을 업로드합니다.
2. register-application-revision 명령을 호출해 다음을 지정합니다.
   - 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
   - 등록할 개정에 대한 정보:
     - 개정이 포함된 Amazon S3 버킷의 이름.
     - 업로드된 개정의 이름 및 파일 유형. AWS Lambda 배포의 경우 개정은 JSON 또는 YAML로 작성된 AppSpec 파일입니다. EC2/온프레미스 배포의 경우 개정에는 CodeDeploy가 인스턴스에 배포할 소스 파일 버전 또는 인스턴스에서 실행할 스크립트가 포함되어 있습니다.

       > **Note**
       >
       > tar 및 압축된 tar 아카이브 파일 형식(.tar 및 .tar.gz)은 Windows Server 인스턴스에서 지원되지 않습니다.

     - (선택 사항) 개정의 Amazon S3 버전 식별자. (버전 식별자를 지정하지 않으면 CodeDeploy에서는 가장 최신 버전을 사용합니다.)
     - (선택 사항) 개정의 ETag. (ETag를 지정하지 않으면 CodeDeploy가 객체 검증을 건너뜁니다.)
   - (선택 사항) 개정과 연관시킬 설명.

다음 구문을 register-application-revision 호출의 일부로 사용하여 Amazon S3에 있는 개정에 대한 정보를 명령줄에서 지정할 수 있습니다(`version` 및 `eTag`는 선택 사항).

EC2/온프레미스 배포에 대한 개정 파일의 경우:

```
--s3-location bucket=string,key=string,bundleType=tar|tgz|
zip,version=string,eTag=string
```

AWS Lambda 배포를 위한 개정 파일의 경우:

```
--s3-location bucket=string,key=string,bundleType=JSON|YAML,version=string,eTag=string
```

CodeDeploy(CLI)를 사용하여 GitHub에 개정 등록

> **Note**
>
> AWS Lambda 배포는 GitHub에서 작동하지 않습니다.

1. GitHub 리포지토리에 개정을 업로드합니다.
2. register-application-revision 명령을 호출해 다음을 지정합니다.
   - 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
   - 등록할 개정에 대한 정보:
     - 개정이 포함되어 있는 리포지토리의 GitHub 사용자 또는 그룹 이름, 슬래시(/), 리포지토리 이름(해당 순으로 입력)
     - 리포지토리에서 개정을 참조하는 커밋 ID.
   - (선택 사항) 개정과 연관시킬 설명.

다음 구문을 register-application-revision 호출의 일부로 사용하여 GitHub에 있는 개정에 대한 정보를 명령줄에서 지정할 수 있습니다.

```
--github-location repository=string,commitId=string
```

CodeDeploy에서 배포 작업

CodeDeploy에서 배포는 하나 이상의 인스턴스에 콘텐츠를 설치하는 프로세스 및 프로세스와 관련된 구성 요소입니다. 이 컨텐츠는 코드, 웹 및 구성 파일, 실행 파일, 패키지, 스크립트 등으로 구성될 수 있습니다. CodeDeploy는 지정한 구성 규칙에 따라 소스 리포지토리에 저장된 콘텐츠를 배포합니다.

EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 경우 동일한 인스턴스 집합에 대한 두 개의 배포가 동시에 실행될 수 있습니다.

CodeDeploy는 두 가지 배포 유형 옵션, 즉 인 플레이스(in-place) 배포와 블루/그린 배포를 제공합니다.

- 인 플레이스(in-place) 배포: 배포 그룹의 각 인스턴스에 있는 애플리케이션이 중지되고 최신 애플리케이션 개정 버전이 설치되며 애플리케이션의 새 버전이 시작되고 유효성이 검사됩니다. 로드 밸런서를 사용하면 배포가 진행될 때 각 인스턴스를 등록 취소한 후 배포가 완료된 후 서비스로 복원할 수 있습니다. EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 배포만 인 플레이스 배포를 사용할 수 있습니다. 현재 위치 배포에 대한 자세한 내용은 인 플레이스 배포 개요 단원을 참조하세요.
- Blue/Green 배포: 배포 동작은 사용하는 컴퓨팅 플랫폼에 따라 다릅니다.
  - EC2/온프레미스 컴퓨팅 플랫폼에서의 블루/그린 배포: 배포 그룹(원래 환경)의 인스턴스가 다음 단계를 거쳐 인스턴스의 다른 집합(대체 환경)으로 대체됩니다.
    - 인스턴스는 대체 환경을 위해 프로비저닝됩니다.
    - 최신 애플리케이션 수정은 대체 인스턴스에 설치됩니다.
    - 애플리케이션 테스트 및 시스템 검증과 같은 활동에 선택적 대기 시간이 발생합니다.
    - 대체 환경의 인스턴스가 하나 이상의 Elastic Load Balancing 로드 밸런서에 등록되고 트래픽이 이러한 인스턴스로 라우팅됩니다. 원래 환경의 인스턴스는 등록이 취소되고 종료되거나 다른 용도로 계속 실행될 수 있습니다.

    > **Note**
    >
    > EC2/온프레미스 컴퓨팅 플랫폼을 사용할 경우 블루/그린 배포는 Amazon EC2 인스턴스에서만 작동합니다.

  - AWS Lambda 또는 Amazon ECS 컴퓨팅 플랫폼의 블루/그린: 트래픽은 canary, 선형 또는 all-at-once 배포 구성에 따라 증분으로 이동합니다.
  - 블루/그린 배포: CloudFormation CloudFormation 스택 업데이트의 일부로 트래픽이 현재 리소스에서 업데이트된 리소스로 이동합니다. 현재는 ECS 블루/그린 배포만 지원됩니다.

블루/그린 배포에 대한 자세한 내용은 블루/그린 배포 개요 단원을 참조하세요.

Amazon S3 에서 자동 배포에 대한 자세한 내용은 CodeDeploy를 사용하여 Amazon S3에서 자동 배포를 참조하세요.

주제

- CodeDeploy에서 배포 만들기
- CodeDeploy 배포 세부 정보 보기
- CodeDeploy EC2/온프레미스 배포에 대한 로그 데이터 보기
- CodeDeploy를 사용하여 배포 중지
- CodeDeploy를 사용하여 재배포 및 배포 롤백
- 다른 AWS 계정에 애플리케이션 배포
- CodeDeploy 에이전트를 사용하여 로컬 시스템에서 배포 패키지의 유효성을 검사합니다.

CodeDeploy에서 배포 만들기

CodeDeploy 콘솔 AWS CLI, 또는 CodeDeploy APIs를 사용하여 Amazon S3에 이미 푸시한 애플리케이션 개정을 설치하는 배포를 생성하거나, 배포가 EC2/온프레미스 컴퓨팅 플랫폼인 경우 배포 그룹의 인스턴스에서 GitHub에 배포할 수 있습니다.

배포 생성 절차는 배포에서 사용하는 컴퓨팅 플랫폼에 따라 다릅니다.

주제

- 배포 사전 조건
- Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(콘솔)
- AWS Lambda 컴퓨팅 플랫폼 배포 생성(콘솔)
- EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔)
- Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(CLI)
- AWS Lambda 컴퓨팅 플랫폼 배포 생성(CLI)
- EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(CLI)
- 를 통해 Amazon ECS 블루/그린 배포 생성 CloudFormation

배포 사전 조건

배포를 시작하기 전에 다음 단계를 완료해야 합니다.

AWS Lambda 컴퓨팅 플랫폼의 배포 사전 조건

- 배포 그룹이 하나 이상 있는 애플리케이션을 만듭니다. 자세한 내용은 CodeDeploy를 사용하여 애플리케이션 생성 및 CodeDeploy에서 배포 그룹 만들기 단원을 참조하세요.
- AppSpec 파일이라고 하는 애플리케이션 개정 버전을 준비합니다. 이 파일은 배포하려는 Lambda 함수 버전을 지정합니다. 또한 AppSpec 파일은 배포를 확인할 Lambda 함수를 지정할 수도 있습니다. 자세한 내용은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.
- 배포에 사용자 지정 배포 구성을 사용하려면 배포 프로세스를 시작하기 전에 만드세요. 자세한 내용은 Create a Deployment Configuration 단원을 참조하세요.

EC2/온프레미스 컴퓨팅 플랫폼에 대한 배포 사전 조건

- 인 플레이스(In-Place) 배포의 경우 배포 대상 인스턴스를 생성하거나 구성합니다. 자세한 내용은 CodeDeploy용 인스턴스 작업 단원을 참조하세요. 블루/그린 배포인 경우 대체 환경에서 템플릿으로 사용할 기존 Amazon EC2 Auto Scaling 그룹이 있거나 원래 환경으로 지정하는 인스턴스 또는 Amazon EC2 Auto Scaling 그룹이 하나 이상 있습니다. 자세한 내용은 튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포 및 Amazon EC2 Auto Scaling과 CodeDeploy 통합 단원을 참조하세요.
- 배포 그룹이 하나 이상 있는 애플리케이션을 만듭니다. 자세한 내용은 CodeDeploy를 사용하여 애플리케이션 생성 및 CodeDeploy에서 배포 그룹 만들기 단원을 참조하세요.
- 배포 그룹의 인스턴스에 배포하려는 애플리케이션 개정을 준비합니다. 자세한 내용은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.
- 배포에 사용자 지정 배포 구성을 사용하려면 배포 프로세스를 시작하기 전에 만드세요. 자세한 내용은 Create a Deployment Configuration 단원을 참조하세요.
- Amazon S3 버킷에서 애플리케이션 개정을 배포하는 경우 버킷은 배포 그룹의 인스턴스와 동일한 AWS 리전에 있습니다.
- Amazon S3 버킷에서 애플리케이션 개정을 배포하는 경우 Amazon S3 버킷 정책이 버킷에 적용되었습니다. 이 정책은 애플리케이션 개정을 다운로드하는 데 필요한 권한을 인스턴스에 부여합니다.

  예를 들어, 다음 Amazon S3 버킷 정책은 ARN `arn:aws:iam::444455556666:role/CodeDeployDemo`이(가) 포함된 IAM 인스턴스 프로필이 연결되어 있는 Amazon EC2 인스턴스가

이름이 amzn-s3-demo-bucket인 Amazon S3 버킷에서 위치에 관계없이 다운로드할 수 있도록 허용합니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

다음 Amazon S3 버킷 정책은 ARN arn:aws:iam::444455556666:user/CodeDeployUser이(가) 포함된 IAM 사용자가 연결되어 있는 온프레미스 인스턴스가 이름이 amzn-s3-demo-bucket인 Amazon S3 버킷에서 위치에 관계없이 다운로드할 수 있도록 허용합니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
        }
    ]
```

```
}
```

Amazon S3 버킷 정책 생성 및 연결에 대한 자세한 내용은 버킷 정책 예제를 참조하세요.

- 블루/그린 배포를 생성하거나 인 플레이스(in-place) 배포의 배포 그룹에 선택 사항으로 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer를 지정한 경우 서브넷 두 개 이상을 포함하고 있는 Amazon VPC를 사용해 VPC를 생성했습니다. (CodeDeploy는 로드 밸런서 그룹의 모든 인스턴스가 단일 VPC에 있어야 하는 Elastic Load Balancing을 사용합니다.)

  아직 VPC를 생성하지 않은 경우 Amazon VPC 시작 가이드를 참조하세요.
- 블루/그린 배포를 생성하는 경우 Elastic Load Balancing에서 하나 이상의 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer를 구성하고 이를 사용하여 원래 환경을 구성하는 인스턴스를 등록했습니다.

> **Note**
>
> 대체 환경의 인스턴스가 나중에 로드 밸런서를 통해 등록됩니다.

로드 밸런서 구성에 대한 자세한 내용은 Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정 및 CodeDeploy Amazon ECS 배포를 위한 로드 밸런서, 대상 그룹 및 리스너 설정 섹션을 참조하세요.

를 통한 블루/그린 배포를 위한 배포 사전 조건 CloudFormation

- 템플릿은 CodeDeploy 애플리케이션 또는 배포 그룹에 대한 리소스를 모델링할 필요가 없습니다.
- 템플릿에는 두 개 이상의 서브넷이 포함된 Amazon VPC를 사용하는 VPC에 대한 리소스가 포함되어야 합니다.
- 템플릿에는 트래픽을 대상 그룹으로 보내는 데 사용되는 Elastic Load Balancing의 하나 이상 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer에 대한 리소스가 포함되어야 합니다.

Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(콘솔)

이 주제에서는 콘솔을 사용하여 Amazon ECS 서비스를 배포하는 방법을 보여 줍니다. 자세한 내용은 튜토리얼: Amazon ECS에 애플리케이션 배포 및 튜토리얼: Amazon ECS 서비스 배포 및 확인 테스트 섹션을 참조하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 다음 중 하나를 수행하세요.

   - 애플리케이션을 배포하려는 경우 탐색 창에서 배포를 확장한 다음 애플리케이션을 선택합니다. 배포할 애플리케이션의 이름을 선택합니다. 애플리케이션에 대한 컴퓨팅 플랫폼 열이 Amazon ECS인지 확인합니다.
   - 배포를 다시 배포하려는 경우 탐색 창에서 배포를 확장한 다음 배포를 선택합니다. 다시 배포하려는 배포를 선택하고 애플리케이션 열에서 해당 애플리케이션의 이름을 선택합니다. 배포에 대한 컴퓨팅 플랫폼 열이 Amazon ECS인지 확인합니다.

3. 배포 탭에서 배포 만들기를 선택합니다.

   > **Note**
   >
   > 애플리케이션을 배포하려면 애플리케이션에 배포 그룹이 있어야 합니다. 애플리케이션에 배포 그룹이 없으면 배포 그룹 탭에서 배포 그룹 생성을 선택합니다. 자세한 내용은 CodeDeploy에서 배포 그룹 만들기 단원을 참조하십시오.

4. 배포 그룹에서 이 배포에 사용할 배포 그룹을 선택합니다.

5. 개정 위치 옆에서 개정이 있는 위치를 선택합니다.

   - 내 애플리케이션은 Amazon S3에 저장됨 — 자세한 내용은 Amazon S3 버킷에 저장된 개정에 대한 정보 지정 단원을 참조하세요. 그런 다음 6단계로 돌아갑니다.
   - AppSpec 편집기 사용 — JSON 또는 YAML을 선택한 다음 제공된 편집기에 AppSpec 파일을 입력합니다. 텍스트 파일로 저장을 선택하여 AppSpec 파일을 저장할 수 있습니다. 이러한 단계가 끝날 때 배포를 선택하면 해당 JSON 또는 YAML이 유효하지 않은 경우 오류가 발생합니다. AppSpec 파일 만들기에 대한 자세한 내용은 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 단원을 참조하세요.

6. (선택 사항) 배포 설명 상자에 이 배포에 대한 설명을 입력합니다.

7. (선택 사항) [Rollback configuration overrides]에서 이 배포에 배포 그룹에 지정된 것이 아닌 다른 자동 롤백 옵션을 지정할 수 있습니다(있는 경우).

CodeDeploy의 롤백에 대한 자세한 내용은 다시 배포 및 배포 롤백 및 CodeDeploy를 사용하여 재배포 및 배포 롤백를 참조하세요.

다음 중에서 선택합니다.

- 배포 실패 시 롤백 — CodeDeploy는 마지막으로 알려진 정상 개정을 새 배포로 다시 배포합니다.
- 경보 임계값이 충족되는 경우 롤백 — 배포 그룹에 경보를 추가한 경우 CodeDeploy는 지정된 경보 중 하나 이상이 활성화될 때 마지막으로 알려진 정상 개정을 다시 배포합니다.
- 롤백 비활성 — 이 배포에 대해 롤백을 수행하지 않습니다.

8. 배포 만들기를 선택합니다.

배포 상태를 추적하려면 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

AWS Lambda 컴퓨팅 플랫폼 배포 생성(콘솔)

이 주제에서는 콘솔을 사용하여 Lambda 함수를 배포하는 방법을 보여 줍니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 다음 중 하나를 수행하세요.

- 애플리케이션을 배포하려는 경우 탐색 창에서 배포를 확장한 다음 애플리케이션을 선택합니다. 배포할 애플리케이션의 이름을 선택합니다. 애플리케이션에 대한 컴퓨팅 플랫폼 열이 AWS Lambda인지 확인합니다.
- 배포를 다시 배포하려는 경우 탐색 창에서 배포를 확장한 다음 배포를 선택합니다. 다시 배포하려는 배포를 선택하고 애플리케이션 열에서 해당 애플리케이션의 이름을 선택합니다. 배포에 대한 컴퓨팅 플랫폼 열이 AWS Lambda인지 확인합니다.

3. 배포 탭에서 배포 만들기를 선택합니다.

> **Note**
>
> 애플리케이션을 배포하려면 애플리케이션에 배포 그룹이 있어야 합니다. 애플리케이션에 배포 그룹이 없으면 배포 그룹 탭에서 배포 그룹 생성을 선택합니다. 자세한 내용은 CodeDeploy에서 배포 그룹 만들기 단원을 참조하십시오.

4. 배포 그룹에서 이 배포에 사용할 배포 그룹을 선택합니다.
5. 개정 위치 옆에서 개정이 있는 위치를 선택합니다.

   - 내 애플리케이션은 Amazon S3에 저장됨 — 자세한 내용은 Amazon S3 버킷에 저장된 개정에 대한 정보 지정 단원을 참조하세요. 그런 다음 6단계로 돌아갑니다.
   - AppSpec 편집기 사용 — JSON 또는 YAML을 선택한 다음 제공된 편집기에 AppSpec 파일을 입력합니다. 텍스트 파일로 저장을 선택하여 AppSpec 파일을 저장할 수 있습니다. 이러한 단계가 끝날 때 배포를 선택하면 해당 JSON 또는 YAML이 유효하지 않은 경우 오류가 발생합니다. AppSpec 파일 만들기에 대한 자세한 내용은 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 단원을 참조하세요.
6. (선택 사항) 배포 설명 상자에 이 배포에 대한 설명을 입력합니다.
7. (선택 사항) 배포 그룹 재정의를 확장하여 배포 그룹에서 지정된 버전과 다른 Lambda 함수 버전으로 트래픽이 전환되는 방법을 제어하는 배포 구성을 선택합니다.

   자세한 내용은 AWS Lambda 컴퓨팅 플랫폼의 배포 구성 단원을 참조하십시오.
8. (선택 사항) [Rollback configuration overrides]에서 이 배포에 배포 그룹에 지정된 것이 아닌 다른 자동 롤백 옵션을 지정할 수 있습니다(있는 경우).

   CodeDeploy의 롤백에 대한 자세한 내용은 다시 배포 및 배포 롤백 및 CodeDeploy를 사용하여 재배포 및 배포 롤백를 참조하세요.

   다음 중에서 선택합니다.

   - 배포 실패 시 롤백 — CodeDeploy는 마지막으로 알려진 정상 개정을 새 배포로 다시 배포합니다.
   - 경보 임계값이 충족되는 경우 롤백 — 배포 그룹에 경보를 추가한 경우 CodeDeploy는 지정된 경보 중 하나 이상이 활성화될 때 마지막으로 알려진 정상 개정을 다시 배포합니다.
   - 롤백 비활성 — 이 배포에 대해 롤백을 수행하지 않습니다.
9. 배포 만들기를 선택합니다.

배포 상태를 추적하려면 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔)

이 주제에서는 콘솔을 사용하여 애플리케이션을 Amazon EC2 또는 온프레미스 서버에 배포하는 방법을 보여 줍니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 다음 중 하나를 수행하세요.

   - 애플리케이션을 배포하려는 경우 탐색 창에서 배포를 확장한 다음 애플리케이션을 선택합니다. 배포할 애플리케이션의 이름을 선택합니다. 애플리케이션에 대한 컴퓨팅 플랫폼 열이 EC2/온프레미스인지 확인합니다.
   - 배포를 다시 배포하려는 경우 탐색 창에서 배포를 확장한 다음 배포를 선택합니다. 다시 배포할 배포를 찾은 후 애플리케이션 열에서 해당 애플리케이션의 이름을 선택합니다. 배포에 대한 컴퓨팅 플랫폼 열이 EC2/온프레미스인지 확인합니다.

3. 배포 탭에서 배포 만들기를 선택합니다.

   > **Note**
   >
   > 애플리케이션을 배포하려면 애플리케이션에 배포 그룹이 있어야 합니다. 애플리케이션에 배포 그룹이 없으면 배포 그룹 탭에서 배포 그룹 생성을 선택합니다. 자세한 내용은 CodeDeploy에서 배포 그룹 만들기 단원을 참조하십시오.

4. 배포 그룹에서 이 배포에 사용할 배포 그룹을 선택합니다.
5. 리포지토리 유형 옆에서 개정이 저장된 리포지토리 유형을 선택합니다.

   - 내 애플리케이션은 Amazon S3에 저장됨 — 자세한 내용은 Amazon S3 버킷에 저장된 개정에 대한 정보 지정 단원을 참조하세요. 그런 다음 6단계로 돌아갑니다.
   - 내 애플리케이션은 GitHub에 저장됨 — 자세한 내용은 GitHub 리포지토리에 저장된 개정 버전에 대한 정보 지정 단원을 참조하세요. 그런 다음 6단계로 돌아갑니다.

6. (선택 사항) 배포 설명 상자에 이 배포에 대한 설명을 입력합니다.
7. (선택 사항) 배포 구성 재정의를 확장하여 배포 그룹에서 지정된 것과 다른 Amazon EC2 또는 온프레미스 서버로 트래픽이 전환되는 방법을 제어하는 배포 구성을 선택합니다.

   자세한 내용은 CodeDeploy에서 배포 구성 작업 단원을 참조하십시오.
8. a. `ApplicationStop` 수명 주기 이벤트가 실패할 경우에도 인스턴스로의 배포가 성공되도록 하려면 ApplicationStop 수명 주기 이벤트가 실패할 경우에도 배포에 실패하지 않습니다를 선택합니다.

   b. 추가 배포 동작 설정을 확장하여 CodeDeploy가 이전에 성공한 배포의 일부가 아닌 배포 대상 위치에서 파일을 처리하는 방식을 지정합니다.

      다음 중에서 선택합니다.

      - 배포 실패 — 오류가 보고되고 배포 상태가 `Failed`로 변경됩니다.
      - 콘텐츠 덮어쓰기 — 대상 위치에 동일한 이름의 파일이 있는 경우, 애플리케이션 개정의 버전이 이를 대체합니다.
      - 콘텐츠 유지 — 대상 위치에 동일한 이름의 파일이 있는 경우, 이 파일이 유지되고 애플리케이션 개정의 버전이 인스턴스에 복사되지 않습니다.

      자세한 내용은 기존 컨텐츠의 롤백 동작 단원을 참조하십시오.
9. (선택 사항) [Rollback configuration overrides]에서 이 배포에 배포 그룹에 지정된 것이 아닌 다른 자동 롤백 옵션을 지정할 수 있습니다(있는 경우).

   CodeDeploy의 롤백에 대한 자세한 내용은 다시 배포 및 배포 롤백 및 CodeDeploy를 사용하여 재배포 및 배포 롤백를 참조하세요.

   다음 중에서 선택합니다.

   - 배포 실패 시 롤백 — CodeDeploy는 마지막으로 알려진 정상 개정을 새 배포로 다시 배포합니다.
   - 경보 임계값이 충족되는 경우 롤백 — 배포 그룹에 경보를 추가한 경우 CodeDeploy는 지정된 경보 중 하나 이상이 활성화될 때 마지막으로 알려진 정상 개정을 다시 배포합니다.
   - 롤백 비활성 — 이 배포에 대해 롤백을 수행하지 않습니다.
10. Start deployment(배포 시작)를 선택합니다.

    배포 상태를 추적하려면 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

주제

- Amazon S3 버킷에 저장된 개정에 대한 정보 지정
- GitHub 리포지토리에 저장된 개정 버전에 대한 정보 지정

Amazon S3 버킷에 저장된 개정에 대한 정보 지정

EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔)의 단계를 따르고자 하는 경우 다음 단계에 따라 Amazon S3 버킷에 저장된 애플리케이션 개정에 대한 세부 정보를 추가합니다.

1. 개정의 Amazon S3 링크를 개정 위치에 복사합니다. 링크 값을 찾으려면:

   a. 별도의 브라우저 탭에서:

      에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/s3/ Amazon S3 콘솔을 엽니다.

      개정을 찾아 선택합니다.

   b. [Properties] 창이 보이지 않으면, [Properties] 버튼을 선택합니다.

   c. 속성패널에서 링크 필드의 값을 CodeDeploy 콘솔의 개정 버전 위치 상자에 복사합니다.

   개정 위치의 일부로 ETag(파일 체크섬)를 지정하려면:

   - 링크 필드 값이 **?versionId=*versionId***이면 **&etag=**과 ETag를 링크 필드 값 끝에 추가합니다.
   - 링크 필드 값이 버전 ID를 지정하지 않으면 **?etag=**와 ETag를 링크 필드 값 끝에 추가합니다.

   > **Note**
   >
   > 이는 [Link] 필드의 값을 복사하는 것만큼 쉽지는 않지만, 개정 위치를 다음 형식 중 하나로 입력할 수도 있습니다.
   >
   > **s3://*bucket-name*/*folders*/*objectName***
   > **s3://*bucket-name*/*folders*/*objectName*?versionId=*versionId***
   > **s3://*bucket-name*/*folders*/*objectName*?etag=*etag***
   > **s3://*bucket-name*/*folders*/*objectName*?versionId=*versionId*&etag=*etag***
   > ***bucket-name*.s3.amazonaws.com/*folders*/*objectName***

2. [File type] 목록에 파일 형식을 찾을 수 없다는 메시지가 표시되는 경우, 개정의 파일 유형을 선택합니다. 그렇지 않은 경우 검색된 파일 형식을 수락합니다.

GitHub 리포지토리에 저장된 개정 버전에 대한 정보 지정

EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔)의 단계를 따르는 경우 다음 단계에 따라 GitHub 리포지토리에 저장된 애플리케이션 개정에 대한 세부 정보를 추가합니다.

1. [Connect to GitHub]에서 다음 중 하나를 수행합니다.

   - GitHub 계정에 대한 CodeDeploy 애플리케이션의 연결을 만들려면 다른 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. GitHub 계정에서 이 연결을 식별하는 이름을 입력한 후 GitHub에 연결을 선택합니다. 웹 페이지에 애플리케이션에 대해 GitHub와 상호 작용할 권한을 CodeDeploy에 부여하라는 메시지가 표시됩니다. 계속해서 2단계를 진행합니다.
   - 이미 만든 연결을 사용하려면 [GitHub account]에서 이름을 선택한 후 [Connect to GitHub]를 선택합니다. 계속해서 4단계를 진행합니다.
   - 다른 GitHub 계정에 대한 연결을 생성하려면 다른 웹 브라우저 탭에서 GitHub에서 로그아웃합니다. [Connect to a different GitHub account]를 선택한 후 [Connect to GitHub]를 선택합니다. 계속해서 2단계를 진행합니다.

2. GitHub에 로그인하라는 메시지가 표시되면 [Sign in] 페이지의 지침을 따릅니다. GitHub 사용자 이름 또는 이메일 및 암호로 로그인합니다.
3. [Authorize application] 페이지가 나타나면 [Authorize application]을 선택합니다.
4. Create deployment(배포 생성) 페이지의 Repository name(리포지토리 이름) 상자에 개정이 포함된 GitHub 사용자 또는 조직 이름, 그 다음에 슬래시(/), 그 다음에 개정이 포함된 리포지토리의 이름을 입력합니다. 입력할 값을 잘 모를 경우:

   a. 다른 웹 브라우저 탭에서 GitHub 대시보드로 이동합니다.

   b. [Your repositories]에서 마우스 포인터를 대상 리포지토리 이름 위에 올려놓습니다. 도구 설명에 GitHub 사용자 또는 조직 이름과 그 뒤에 슬래시(/) 및 리포지토리 이름이 표시됩니다. 표시된 이 이름을 Repository name(리포지토리 이름) 상자에 입력합니다.

   > **Note**
   >
   > 대상 리포지토리 이름이 [Your repositories]에 표시되지 않는 경우, [Search GitHub] 상자를 사용하여 대상 리포지토리 이름과 GitHub 사용자 또는 조직 이름을 찾습니다.

5. Commit ID(커밋 ID)에 리포지토리의 개정을 참조하는 커밋 ID를 입력합니다. 입력할 값을 잘 모를 경우:

   a. 다른 웹 브라우저 탭에서 GitHub 대시보드로 이동합니다.

   b. [Your repositories]에서 대상 커밋이 포함된 리포지토리 이름을 선택합니다.

   c. 커밋 목록에서 리포지토리의 개정을 참조하는 커밋 ID를 찾아 복사합니다. 이 ID는 일반적으로 40자이고 문자와 숫자로 구성됩니다. (일반적으로 더 긴 커밋 ID 버전의 첫 10자인 더 짧은 커밋 ID 버전을 사용하지 마세요.)

   d. 커밋 ID를 [Commit ID] 상자에 붙여 넣습니다.

Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(CLI)

애플리케이션과 개정 버전(Amazon ECS 배포의 경우 AppSpec 파일)을 만든 후:

create-deployment 명령을 호출하여 다음을 지정합니다.

- 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
- 배포 그룹 이름. 배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출합니다.
- 배포할 개정에 대한 정보:

  Amazon S3에 저장된 개정 버전의 경우:

  - 개정 버전이 포함된 Amazon S3 버킷 이름.
  - 업로드된 개정의 이름.
  - (선택 사항) 개정 버전의 Amazon S3 버전 식별자. (버전 식별자를 지정하지 않으면 CodeDeploy에서는 가장 최근 버전을 사용합니다.)
  - (선택 사항) 개정의 ETag. (ETag를 지정하지 않으면 CodeDeploy가 객체 검증을 건너뜁니다.)

  개정이 Amazon S3에 없는 파일에 저장되어 있는 경우 파일 이름과 경로가 필요합니다. 개정 파일은 JSON 또는 YAML을 사용하여 작성되며 따라서 대부분 확장명이 .json 또는 .yaml입니다.
- (선택 사항) 배포의 설명

개정 파일은 Amazon S3 버킷에 업로드된 파일 또는 문자열로 지정할 수 있습니다. create-deployment 명령의 일부로 사용되는 경우 구문은 다음과 같습니다.

- Amazon S3 버킷:

version 및 eTag는 선택 사항입니다.

```
--s3-location bucket=string,key=string,bundleType=JSON|
YAML,version=string,eTag=string
```

- 문자열:

```
--revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'
```

> **Note**
>
> create-deployment 명령은 파일에서 개정을 로드할 수 있습니다. 자세한 내용은 파일에서 파라미터 로드를 참조하세요.

AWS Lambda 배포 개정 템플릿은 섹션을 참조하세요AWS Lambda 배포를 위한 AppSpec 파일 추가. 개정의 예는 AWS Lambda 배포에 대한 AppSpec 파일 예제 파일을 참조하세요.

배포 상태를 추적하려면 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

AWS Lambda 컴퓨팅 플랫폼 배포 생성(CLI)

애플리케이션 및 개정을 생성한 후(AWS Lambda 배포에서 AppSpec 파일):

create-deployment 명령을 호출하여 다음을 지정합니다.

- 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
- 배포 그룹 이름. 배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출합니다.
- 배포할 개정에 대한 정보:

  Amazon S3에 저장된 개정 버전의 경우:
  - 개정 버전이 포함된 Amazon S3 버킷 이름.
  - 업로드된 개정의 이름.
  - (선택 사항) 개정 버전의 Amazon S3 버전 식별자. (버전 식별자를 지정하지 않으면 CodeDeploy에서는 가장 최근 버전을 사용합니다.)
  - (선택 사항) 개정의 ETag. (ETag를 지정하지 않으면 CodeDeploy가 객체 검증을 건너뜁니다.)

개정이 Amazon S3에 없는 파일에 저장되어 있는 경우 파일 이름과 경로가 필요합니다. 개정 파일은 JSON 또는 YAML을 사용하여 작성되며 따라서 대부분 확장명이 .json 또는 .yaml입니다.

- (선택 사항) 사용할 배포 구성의 이름. 배포 구성 목록을 보려면 list-deployment-configs 명령을 호출합니다. (지정하지 않은 경우 CodeDeploy에서는 특정 기본 배포 구성을 사용합니다.)
- (선택 사항) 배포의 설명

개정 파일은 Amazon S3 버킷에 업로드된 파일 또는 문자열로 지정할 수 있습니다. create-deployment 명령의 일부로 사용되는 경우 구문은 다음과 같습니다.

- Amazon S3 버킷:

  `version` 및 `eTag`는 선택 사항입니다.

```
--s3-location bucket=string,key=string,bundleType=JSON|
YAML,version=string,eTag=string
```

- 문자열:

```
--revision '{"revisionType": "String", "string": {"content":"revision-as-string"}}'
```

> **Note**
>
> create-deployment 명령은 파일에서 개정을 로드할 수 있습니다. 자세한 내용은 파일에서 파라미터 로드를 참조하세요.

AWS Lambda 배포 개정 템플릿은 섹션을 참조하세요AWS Lambda 배포를 위한 AppSpec 파일 추가. 개정의 예는 AWS Lambda 배포에 대한 AppSpec 파일 예제 파일을 참조하세요.

배포 상태를 추적하려면 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(CLI)

AWS CLI 를 사용하여 EC2/온프레미스 컴퓨팅 플랫폼에 개정 버전을 배포하려면:

1. 인스턴스를 준비하고, 애플리케이션을 생성하고, 개정을 푸시한 후 다음 중 하나를 수행합니다.
   - Amazon S3 버킷에 있는 개정을 배포하려면 지금 2단계로 진행합니다.

- GitHub 리포지토리에 있는 개정을 배포하려면 먼저 CodeDeploy 애플리케이션을 GitHub 리포지토리에 연결의 단계를 완료한 후 2단계로 진행합니다.

2. create-deployment 명령을 호출하여 다음을 지정합니다.

- `--application-name`: 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
- `--deployment-group-name`: Amazon EC2 배포 그룹 이름. 배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출합니다.
- `--revision`: 배포할 개정에 대한 정보:

  Amazon S3에 저장된 개정 버전의 경우:

  - `s3Location`: 개정이 포함되어 있는 Amazon S3 버킷 이름.
  - `s3Location --> key`: 업로드된 개정의 이름.
  - `s3Location --> bundleType`: 업로드된 개정의 파일 유형.

    > **Note**
    >
    > tar 및 압축된 tar 아카이브 파일 형식(.tar 및 .tar.gz)은 Windows Server 인스턴스에서 지원되지 않습니다.

  - `s3Location --> version`: (선택 사항) 개정 버전의 Amazon S3 버전 식별자. (버전 식별자를 지정하지 않으면 CodeDeploy에서는 가장 최근 버전을 사용합니다.)
  - `s3Location --> eTag`: (선택 사항) 개정의 ETag. (ETag를 지정하지 않으면 CodeDeploy가 객체 검증을 건너뜁니다.)

  GitHub에 저장된 개정의 경우:

  - `gitHubLocation --> repository`: 개정이 포함되어 있는 리포지토리의 GitHub 사용자 또는 그룹 이름, 슬래시(/), 리포지토리 이름(해당 순으로 입력)
  - `gitHubLocation --> commitId`: 개정의 커밋 ID

- `--deployment-config-name`: (선택 사항) 사용할 배포 구성의 이름. 배포 구성 목록을 보려면 list-deployment-configs 명령을 호출합니다. (지정하지 않은 경우 CodeDeploy에서는 특정 기본 배포 구성을 사용합니다.)
- `--ignore-application-stop-failures | --no-ignore-application-stop-failures`: (선택 사항) `ApplicationStop` 배포 수명 주기 이벤트가 실패할 경우 인스턴스에 대한 배포가 `BeforeInstall` 배포 수명 주기 이벤트로 계속되도록 할지 여부.
- `--description`: (선택 사항) 배포의 설명

- `--file-exists-behavior`: (선택 사항) 배포 프로세스의 일부로 CodeDeploy 에이전트는 가장 최근 배포에 의해 설치된 모든 파일을 각 인스턴스에서 제거합니다. 이전 배포의 일부가 아닌 파일이 대상 배포 위치에 나타나면 어떻게 되는지 선택합니다.
- `--target-instances`: 블루/그린 배포의 경우, 블루/그린 배포에서 대체 환경에 속하는 인스턴스에 대한 정보(하나 이상의 Amazon EC2 Auto Scaling 그룹 이름 또는 Amazon EC2 인스턴스를 식별하는 데 사용되는 태그 필터 키, 유형, 값 등)

> **Note**
>
> 다음 구문을 create-deployment 호출의 일환으로 사용해 Amazon S3에 있는 개정에 대한 정보를 명령줄에 직접 지정하세요. (`version` 및 `eTag`는 선택 사항입니다.)
>
> ```
> --s3-location bucket=string,key=string,bundleType=tar|tgz|
> zip,version=string,eTag=string
> ```
>
> 다음 구문을 create-deployment 호출의 일환으로 사용해 GitHub에 있는 개정에 대한 정보를 명령줄에 직접 지정하세요.
>
> ```
> --github-location repository=string,commitId=string
> ```
>
> 이미 푸시된 개정에 대한 정보를 확인하려면 list-application-revisions 명령을 호출하세요.

배포 상태를 추적하려면 CodeDeploy 배포 세부 정보 보기 단원을 참조하세요.

create-deployment 명령 참조

명령에 사용할 수 있는 `create-deployment` 명령 구조 및 옵션은 다음과 같습니다. 자세한 내용은 AWS CLI 명령 참조의 create-deployment 참조를 참조합니다.

```
create-deployment
--application-name <value>
[--deployment-group-name <value>]
[--revision <value>]
[--deployment-config-name <value>]
[--description <value>]
[--ignore-application-stop-failures | --no-ignore-application-stop-failures]
[--target-instances <value>]
```

```
[--auto-rollback-configuration <value>]
[--update-outdated-instances-only | --no-update-outdated-instances-only]
[--file-exists-behavior <value>]
[--s3-location <value>]
[--github-location <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
```

CodeDeploy 애플리케이션을 GitHub 리포지토리에 연결

를 사용하여 GitHub 리포지토리에서 애플리케이션을 처음 배포하려면 먼저 CodeDeploy에 GitHub 계정을 대신하여 GitHub와 상호 작용할 수 있는 권한을 부여 AWS CLI해야 합니다. 이 단계는 CodeDeploy 콘솔을 사용하는 애플리케이션 각각에 대해 한 번씩 완료해야 합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. [Applications]를 선택합니다.
3. [Application]에서 GitHub 사용자 계정에 연결할 애플리케이션을 선택하고 [Deploy application]을 선택합니다.

   > **Note**
   >
   > 배포를 만들고 있지 않습니다. 이는 현재 GitHub 사용자 계정을 대신하여 CodeDeploy에 GitHub와 상호 작용할 권한을 부여하는 유일한 방법입니다.

4. [Repository type] 옆의 [My application revision is stored in GitHub]를 선택합니다.
5. GitHub에 연결을 선택합니다.

   > **Note**
   >
   > 다른 GitHub 계정에 연결 링크가 보이는 경우:
   > 애플리케이션에 대해 다른 GitHub 계정을 대신하여 GitHub와 상호 작용할 권한을 CodeDeploy에 이미 부여했을 수 있습니다.

CodeDeploy에 연결된 모든 애플리케이션에 대해 로그인한 GitHub 계정을 대신하여 GitHub와 상호 작용할 CodeDeploy의 권한을 취소했을 수 있습니다.
자세한 내용은 CodeDeploy에서 애플리케이션으로 GitHub 인증 단원을 참조하십시오.

6. GitHub에 아직 로그인하지 않은 경우, [Sign in] 페이지의 지침을 따릅니다.
7. [Authorize application] 페이지에서 [Authorize application]을 선택합니다.
8. 이제 CodeDeploy에 권한이 있으므로 취소를 선택한 후 EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(CLI)의 단계를 계속 진행합니다.

를 통해 Amazon ECS 블루/그린 배포 생성 CloudFormation

AWS CloudFormation 를 사용하여 CodeDeploy를 통해 Amazon ECS 블루/그린 배포를 관리할 수 있습니다. 그린 및 블루 리소스를 정의하고 CloudFormation에서 사용할 트래픽 라우팅 및 안정화 설정을 지정하여 배포를 생성합니다. 이 주제에서는 CodeDeploy에서 관리하는 Amazon ECS 블루/그린 배포와 CloudFormation에서 관리하는 배포 간 차이점에 대해 설명합니다.

를 사용하여 Amazon ECS 블루/그린 배포 CloudFormation 를 관리하는 방법에 대한 step-by-step 지침은 AWS CloudFormation 사용 설명서의 를 사용하여 CodeDeploy를 통해 ECS 블루/그린 배포 자동화 AWS CloudFormation를 참조하세요.

> **Note**
> 아시아 태평양(오사카) 리전에서는를 사용하여 Amazon ECS 블루/그린 배포를 관리할 수 CloudFormation 없습니다.

CodeDeploy 및를 통한 Amazon ECS 블루/그린 배포의 차이점 CloudFormation

CloudFormation 스택 템플릿은 Amazon ECS 작업 관련 리소스 및 인프라와 배포를 위한 구성 옵션을 모델링합니다. 따라서 표준 Amazon ECS 블루/그린 배포와를 통해 생성되는 블루/그린 배포 간에는 차이가 있습니다 CloudFormation.

표준 Amazon ECS 블루/그린 배포와 달리 다음을 모델링하거나 수동으로 만들지 않습니다.

- 배포하려는 항목을 고유하게 나타내는 이름을 지정하여 AWS CodeDeploy 애플리케이션을 생성하지 않습니다.
- AWS CodeDeploy 배포 그룹은 생성하지 않습니다.

- 애플리케이션 사양 파일(AppSpec 파일)을 지정하지 않습니다. 일반적으로 AppSpec 파일로 관리되는 정보(예: 가중치 기반 구성 옵션 또는 수명 주기 이벤트)는 `AWS::CodeDeploy::BlueGreen` 후크에 의해 관리됩니다.

이 표에는 배포 유형 간 상위 수준 워크플로의 차이점이 요약되어 있습니다.

함수	표준 블루/그린 배포	를 통한 블루/그린 배포 CloudFormation
Amazon ECS 클러스터, Amazon ECS 서비스, Application Load Balancer 또는 Network Load Balancer, 프로덕션 리스너, 테스트 리스너, 대상 그룹 2개를 지정합니다.	이러한 리소스를 지정하는 CodeDeploy 배포 그룹을 만듭니다.	CloudFormation 템플릿을 생성하여 이러한 리소스를 모델링합니다.
배포할 변경 사항을 지정합니다.	CodeDeploy 애플리케이션을 만듭니다.	컨테이너 이미지를 지정하는 CloudFormation 템플릿을 생성합니다.
Amazon ECS 작업 정의, 컨테이너 이름 및 컨테이너 포트를 지정합니다.	이러한 리소스를 지정하는 AppSpec 파일을 만듭니다.	CloudFormation 템플릿을 생성하여 이러한 리소스를 모델링합니다.
배포 트래픽 이동 옵션과 수명 주기 이벤트 후크를 지정합니다.	이러한 옵션을 지정하는 AppSpec 파일을 만듭니다.	`AWS::CodeDeploy::BlueGreen` 후크 파라미터를 사용하여 이러한 옵션을 지정하는 CloudFormation 템플릿을 생성합니다.
CloudWatch 경보.	롤백을 트리거하는 CloudWatch 경보를 만듭니다.	롤백을 트리거하는 CloudFormation 스택 수준에서 CloudWatch 경보를 구성합니다.
롤백/재배포.	롤백 및 재배포 옵션을 지정합니다.	스택 업데이트를 취소합니다 CloudFormation.

를 통한 Amazon ECS 블루/그린 배포 모니터링 CloudFormation

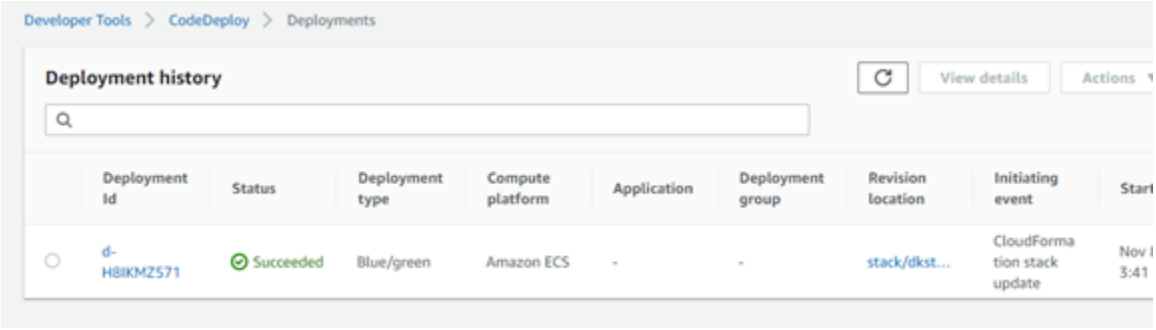
CloudFormation 및 CodeDeploy를 통해 블루/그린 배포를 모니터링할 수 있습니다. 를 통한 모니터링에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 에서 블루/그린 이벤트 모니터링을 CloudFormation CloudFormation참조하세요.

CodeDeploy에서 블루/그린 배포의 배포 상태를 보려면

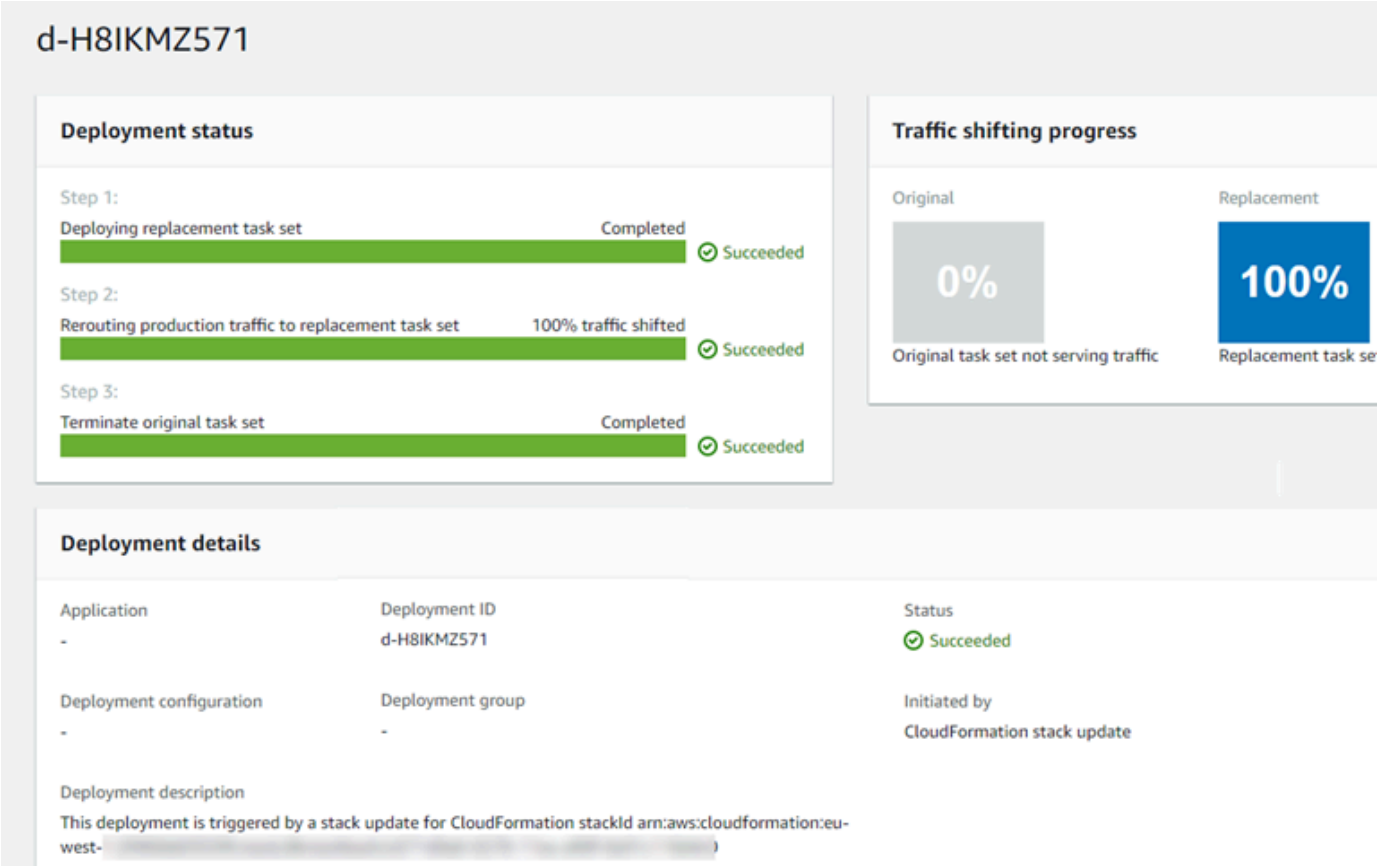
1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 배포에서 CloudFormation 스택 업데이트에 의해 트리거된 배포가 나타납니다. 배포를 선택하여 Deployment history(배포 이력)을 봅니다.



3. 배포를 선택하여 트래픽 이동 상태를 봅니다. 애플리케이션 및 배포 그룹은 생성되지 않습니다.



4. 다음은 배포 롤백 또는 중지에 적용됩니다.

   - CodeDeploy에 성공적인 배포가 나타나고 배포가 CloudFormation에서 시작되었음을 보여 줍니다.
   - 배포를 중지하고 롤백하려면 스택 업데이트를 취소해야 합니다 CloudFormation.

CodeDeploy 배포 세부 정보 보기

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy APIs를 사용하여 AWS 계정과 연결된 배포에 대한 세부 정보를 볼 수 있습니다.

> **Note**
>
> 다음 위치에서 인스턴스에 대한 EC2/온프레미스 배포 로그를 볼 수 있습니다.
>
> - Amazon Linux, RHEL 및 Ubuntu Server: `/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`

- Windows Server: C:\ProgramData\Amazon\CodeDeploy<DEPLOYMENT-GROUP-ID><DEPLOYMENT-ID>\logs\scripts.log

자세한 내용은 인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석 단원을 참조하십시오.

주제

- 배포 세부 정보 보기(콘솔)
- 배포 세부 정보 보기(CLI)

배포 세부 정보 보기(콘솔)

CodeDeploy 콘솔을 사용하여 배포 세부 정보를 보려면 다음과 같이 하세요.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 배포를 선택합니다.

   > **Note**
   >
   > 항목이 표시되지 않으면 올바른 리전이 선택되어 있는지 확인합니다. 탐색 모음의 리전 선택기에서 AWS 일반 참조의 리전 및 엔드포인트에 나열된 리전 중 하나를 선택합니다. CodeDeploy는 이러한 리전에서만 지원됩니다.

3. 단일 배포에 대한 세부 정보를 보려면배포 이력에서 배포 ID를 선택하거나 배포 ID 옆에 있는 단추를 선택한 후 보기를 선택합니다.

배포 세부 정보 보기(CLI)

AWS CLI 를 사용하여 배포 세부 정보를 보려면 `get-deployment` 명령 또는 `batch-get-deployments` 명령을 호출합니다. `list-deployments` 명령을 호출하여 `get-deployment` 명령 및 `batch-get-deployments` 명령에 입력으로 사용할 고유한 배포 ID 목록을 가져올 수 있습니다.

단일 배포 구성에 대한 세부 정보를 보려면 get-deployment 명령을 호출하여고유한 배포 식별자를 지정합니다. 배포 ID를 가져오려면 list-deployments 명령을 호출합니다.

여러 배포에 대한 세부 정보를 보려면 batch-get-deployments 명령을 호출하여 여러 고유한 배포 식별자를 지정합니다. 배포 ID를 가져오려면 list-deployments 명령을 호출합니다.

배포 ID 목록을 보려면 list-deployments 명령을 호출하여 다음을 지정합니다.

- 배포와 연결된 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
- 애플리케이션과 연결된 배포 그룹 이름. 배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출합니다.
- 선택적으로 배포 상태에 따라 배포에 대한 세부 정보를 포함할지 여부를 지정합니다. (지정하지 않으면 배포 상태에 관계없이 일치하는 모든 배포가 나열됩니다.)
- 선택적으로 배포 생성 시작 시간 또는 종료 시간 또는 둘 다에 따라 배포에 대한 세부 정보를 포함할지 여부를 지정합니다. (지정하지 않으면 생성 시간에 관계없이 일치하는 모든 배포가 나열됩니다.)

CodeDeploy EC2/온프레미스 배포에 대한 로그 데이터 보기

CloudWatch 콘솔에서 집계 데이터를 보도록 Amazon CloudWatch 에이전트를 설정하거나 로그 파일을 검토하기 위해 개별 인스턴스에 로그인하여 CodeDeploy 배포에서 생성된 로그 데이터를 볼 수 있습니다.

> **Note**
>
> AWS Lambda 또는 Amazon ECS 배포에는 로그가 지원되지 않습니다. EC2/온프레미스 배포에만 사용할 수 있습니다.

주제

- Amazon CloudWatch 콘솔에서 로그 파일 데이터 보기

- 인스턴스의 로그 파일 보기

Amazon CloudWatch 콘솔에서 로그 파일 데이터 보기

Amazon CloudWatch 에이전트가 인스턴스에 설치되면 해당 인스턴스에 대한 모든 배포의 배포 데이터를 CloudWatch 콘솔에서 볼 수 있게 됩니다. 간소화를 위해 인스턴스별로 로그 파일을 보는 대신 CloudWatch를 사용하여 중앙에서 로그 파일을 모니터링하는 것이 좋습니다. 자세한 내용은 CloudWatch로 CodeDeploy 에이전트 로그 전송 단원을 참조하십시오.

인스턴스의 로그 파일 보기

개별 인스턴스에 대한 배포 로그 데이터를 보려면 인스턴스에 로그인하여 오류 또는 기타 배포 이벤트에 대한 정보를 찾아볼 수 있습니다.

주제

- Amazon Linux, RHEL 및 Ubuntu Server 인스턴스에서 배포 로그 파일을 보려면 다음을 수행하세요.
- Windows Server 인스턴스에서 배포 로그 파일을 보려면

Amazon Linux, RHEL 및 Ubuntu Server 인스턴스에서 배포 로그 파일을 보려면 다음을 수행하세요.

Amazon Linux, RHEL 및 Ubuntu Server 인스턴스에서 배포 로그는 다음 위치에 저장됩니다.

`/opt/codedeploy-agent/deployment-root/deployment-logs/codedeploy-agent-deployments.log`

Amazon Linux, RHEL 및 Ubuntu Server 인스턴스에서 배포 로그를 보거나 분석하려면 인스턴스에 로그인한 후 다음 명령을 입력하여 CodeDeploy 에이전트 로그 파일을 엽니다.

```
less /var/log/aws/codedeploy-agent/codedeploy-agent.log
```

버전 2.0.x 이상의 경우 에이전트 로그는 매일 또는 파일이 64MB(구성 가능)에 도달하면 교체됩니다. 교체된 아카이브의 이름은 `codedeploy-agent.YYYYMMDD.log` (당일 교체의 경우 숫자 접미사 포함)이며 7일 후에 정리됩니다. 스크립트 로그(`scripts.log`)는 최대 8개의 아카이브된 파일(구성 가능)과 함께 64MB로 크기가 회전됩니다.

다음 명령을 입력하여 로그 파일에서 오류 메시지를 찾습니다.

명령	결과
& ERROR	로그 파일에 오류 메시지만 표시. **ERROR** 단어 앞뒤에 단일 공백을 사용합니다.
/ ERROR	다음 오류 메시지를 검색합니다.[1]
? ERROR	이전 오류 메시지를 검색합니다.[2] **ERROR** 단어 앞뒤에 공백을 하나 사용합니다.
G	로그 파일의 끝 부분으로 이동합니다.
g	로그 파일의 시작 부분으로 이동.
q	로그 파일 종료.
h	추가 명령에 대해 알아봅니다.

[1] **/ ERROR** 입력 후 다음 오류 메시지에 **n**을(를) 입력합니다. 이전 오류 메시지에 **N**을(를) 입력합니다.

[2] **? ERROR** 입력 후 다음 오류 메시지에 **n**을(를) 입력하거나 이전 오류 메시지에 **N**을(를) 입력합니다.

다음 명령을 입력하여 CodeDeploy 스크립트 로그 파일을 열 수도 있습니다.

```
less /opt/codedeploy-agent/deployment-root/deployment-group-ID/deployment-ID/logs/
scripts.log
```

다음 명령을 입력하여 로그 파일에서 오류 메시지를 찾습니다.

명령	결과
&stderr	로그 파일에 오류 메시지만 표시.
/stderr	다음 오류 메시지 검색.[1]
?stderr	이전 오류 메시지 검색.[2]

명령	결과
G	로그 파일의 끝 부분으로 이동.
g	로그 파일의 시작 부분으로 이동.
q	로그 파일 종료.
h	추가 명령에 대해 알아봅니다.

[1] **/stderr** 입력 후 다음 오류 메시지 메시지 앞으로 **n**을(를) 입력합니다. 이전 오류 메시지 뒤로 **N**을(를) 입력합니다.

[2] **?stderr** 입력 후 다음 오류 메시지 뒤로 **n**을(를) 입력합니다. 이전 오류 메시지 앞으로 **N**을(를) 입력합니다.

Windows Server 인스턴스에서 배포 로그 파일을 보려면

CodeDeploy 에이전트 로그 파일: Windows Server 인스턴스에서 CodeDeploy 에이전트 로그 파일은 다음 위치에 저장됩니다.

버전 2.0.x 이상의 경우:

`C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent.log`

버전 1.8.x 이하의 경우:

`C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt`

Windows Server 인스턴스에서 CodeDeploy 에이전트 로그 파일을 보거나 분석하려면 인스턴스에 로그인한 후 다음 명령을 입력하여 파일을 엽니다.

버전 2.0.x 이상의 경우:

```
notepad C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent.log
```

버전 1.8.x 이하의 경우:

```
notepad C:\ProgramData\Amazon\CodeDeploy\log\codedeploy-agent-log.txt
```

로그 파일에서 오류 메시지를 찾아보려면 Ctrl+F를 누르고 **ERROR**을(를) 입력한 다음 Enter 키를 눌러 첫 번째 오류를 찾습니다.

CodeDeploy 스크립트 로그 파일: Windows Server 인스턴스에서 배포 로그는 다음 위치에 저장됩니다.

`C:\ProgramData\Amazon\CodeDeploy\`*`deployment-group-id`*`\`*`deployment-id`*`\logs\scripts.log`

위치:

- *`deployment-group-id`*는 `examplebf3a9c7a-7c19-4657-8684-b0c68d0cd3c4`와(과) 같은 문자열입니다.
- *`deployment-id`*는 `d-12EXAMPLE`와(과) 같은 식별자입니다.

다음 명령을 입력하여 CodeDeploy 스크립트 로그 파일을 엽니다.

```
notepad C:\ProgramData\Amazon\CodeDeploy\deployment-group-ID\deployment-ID\logs
\scripts.log
```

버전 2.0.x 이상의 경우 에이전트 로그는 매일 또는 파일이 64MB(구성 가능)에 도달하면 교체됩니다. 스크립트 로그는 최대 8개의 아카이브된 파일(구성 가능)과 함께 64MB로 크기가 회전됩니다.

로그 파일에서 오류 메시지를 찾아보려면 Ctrl+F를 누르고 **stderr**을(를) 입력한 다음 Enter 키를 눌러 첫 번째 오류를 찾습니다.

CodeDeploy를 사용하여 배포 중지

CodeDeploy 콘솔, AWS CLI또는 CodeDeploy APIs를 사용하여 AWS 계정과 연결된 배포를 중지할 수 있습니다.

> **Warning**
>
> EC2/온프레미스 배포를 중지하면 배포 그룹의 일부 또는 모든 인스턴스가 확정되지 않은 배포 상태로 유지될 수 있습니다. 자세한 내용은 중지 및 실패한 배포 단원을 참조하십시오.

배포를 중지하거나 배포를 중지 및 롤백할 수 있습니다.

- 배포 중지(콘솔)

- 배포 중지(CLI)

> **Note**
>
> 배포가를 통한 블루/그린 배포인 경우 CodeDeploy 콘솔에서이 작업을 수행할 CloudFormation 수 없습니다. CloudFormation 콘솔로 이동하여이 작업을 수행합니다.

배포 중지(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 배포를 선택합니다.

   > **Note**
   >
   > 항목이 표시되지 않으면 올바른 리전이 선택되어 있는지 확인합니다. 탐색 모음의 리전 선택기에서 AWS 일반 참조의 리전 및 엔드포인트에 나열된 리전 중 하나를 선택합니다. CodeDeploy는 이러한 리전에서만 지원됩니다.

3. 중지할 배포를 선택하려면 다음 중 하나를 수행합니다.

   1. 배포 중지를 선택하여 롤백 없이 배포를 중지합니다.
   2. 배포 중지 및 롤백을 선택하여 배포를 중지하고 롤백합니다.

   자세한 내용은 CodeDeploy를 사용하여 재배포 및 배포 롤백 단원을 참조하십시오.

   > **Note**
   >
   > 배포 중지 및 배포 중지 및 롤백을 사용할 수 없다면 배포가 중지할 수 없는 지점으로 진행한 것입니다.

배포 중지(CLI)

stop-deployment 명령을 호출하여 배포 ID를 지정합니다. 배포 ID 목록을 보려면 list-deployments 명령을 호출합니다.

CodeDeploy를 사용하여 재배포 및 배포 롤백

CodeDeploy는 이전에 배포된 애플리케이션의 개정 버전을 새 배포로 다시 배포하여 롤백합니다. 이러한 롤백 배포는 기술적으로 이전 배포의 복원된 버전이 아닌 새 배포 ID를 사용하는 새로운 배포입니다.

배포는 자동 또는 수동으로 롤백할 수 있습니다.

주제

- 자동 롤백
- 수동 롤백
- 롤백 및 재배포 워크플로
- 기존 컨텐츠의 롤백 동작

자동 롤백

배포에 실패하거나 지정한 모니터링 임계값이 충족될 때 자동으로 롤백하도록 배포 그룹 또는 배포를 구성할 수 있습니다. 이 경우 마지막으로 알려진 정상 버전의 애플리케이션 개정 버전이 배포됩니다. 애플리케이션을 만들거나 배포 그룹을 만들거나 업데이트할 때 자동 롤백을 구성합니다.

새 배포를 만들 때 배포 그룹에 지정된 자동 롤백 구성을 재정의하도록 선택할 수도 있습니다.

> **Note**
>
> 배포가 자동으로 롤백될 때마다 Amazon Simple Notification Service를 사용하여 알림을 받을 수 있습니다. 자세한 내용은 Monitoring Deployments with Amazon SNS Event Notifications 단원을 참조하세요.

자동 롤백을 구성하는 방법에 대한 자세한 내용은 배포 그룹에 대한 고급 옵션 구성 단원을 참조하세요.

수동 롤백

자동 롤백을 설정하지 않은 경우 이전에 배포된 애플리케이션 개정 버전을 사용하는 새 배포를 만들고 개정 버전을 다시 배포하는 단계에 따라 배포를 수동으로 롤백할 수 있습니다. 이 작업은 애플리케이션이 알 수 없는 상태가 된 경우에 수행할 수 있습니다. 문제 해결에 많은 시간을 소비하는 대신 애플리케이션을 알려진 작업 상태로 다시 배포할 수 있습니다. 자세한 내용은 CodeDeploy에서 배포 만들기 단원을 참조하십시오.

> **Note**
>
> 배포 그룹에서 인스턴스를 제거해도 CodeDeploy는 해당 인스턴스에 이미 설치되어 있을 수 있는 항목을 제거하지 않습니다.

롤백 및 재배포 워크플로

자동 롤백이 시작되거나 재배포 또는 수동 롤백을 수동으로 시작할 때 CodeDeploy는 먼저 각 참여 인스턴스에서 마지막으로 성공적으로 설치된 모든 파일을 제거하려고 시도합니다. CodeDeploy는 정리 파일을 확인하여 이 작업을 수행합니다.

`/opt/codedeploy-agent/deployment-root/deployment-instructions/`*`deployment-group-ID`*`-cleanup` 파일(Amazon Linux, Ubuntu Server 및 RHEL 인스턴스의 경우)

`C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`*`deployment-group-ID`*`-cleanup` 파일(Windows Server 인스턴스의 경우)

존재하는 경우 CodeDeploy 는 새 배포를 시작하기 전에 정리 파일을 사용하여 나열된 모든 파일을 인스턴스에서 제거합니다.

예를 들어, 처음 두 개의 텍스트 파일과 두 개의 스크립트 파일이 이미 Windows Server를 실행하는 Amazon EC2 인스턴스에 배포되었으며, 이 스크립트는 배포 수명 주기 이벤트 동안 두 개의 텍스트 파일을 더 만들었습니다.

```
c:\temp\a.txt (previously deployed by CodeDeploy)
c:\temp\b.txt (previously deployed by CodeDeploy)
c:\temp\c.bat (previously deployed by CodeDeploy)
c:\temp\d.bat (previously deployed by CodeDeploy)
c:\temp\e.txt (previously created by c.bat)
c:\temp\f.txt (previously created by d.bat)
```

정리 파일에는 처음 두 개의 텍스트 파일과 두 개의 스크립트 파일만 나열됩니다.

```
c:\temp\a.txt
c:\temp\b.txt
c:\temp\c.bat
c:\temp\d.bat
```

새 배포 전에 CodeDeploy는 처음 두 개의 텍스트 파일과 두 개의 스크립트 파일만 제거하고 마지막 두 개의 텍스트 파일은 그대로 둡니다.

```
c:\temp\a.txt will be removed
c:\temp\b.txt will be removed
c:\temp\c.bat will be removed
c:\temp\d.bat will be removed
c:\temp\e.txt will remain
c:\temp\f.txt will remain
```

이 프로세스의 일부로 CodeDeploy는 수동 또는 자동 롤백 여부에 관계없이 후속 재배포 중에 이전 배포의 스크립트에서 수행한 작업을 되돌리거나 조정하지 않습니다. 예를 들어, `c.bat` 및 `d.bat` 파일에 `e.txt` 및 `f.txt` 파일을 재생성하지 않는 로직이 포함되어 있는 경우 이들이 이미 존재한다면 CodeDeploy가 후속 배포에서 `c.bat` 및 `d.bat`을(를) 실행할 때마다 `e.txt` 및 `f.txt`의 이전 버전이 그대로 남아 있게 됩니다. 새 버전을 생성하기 전에 항상 `e.txt` 및 `f.txt`의 이전 버전을 확인하여 삭제하는 로직을 `c.bat` 및 `d.bat`에 추가할 수 있습니다.

기존 컨텐츠의 롤백 동작

배포 프로세스의 일환으로 CodeDeploy 에이전트는 최신 배포가 설치한 모든 파일을 각 인스턴스에서 제거합니다. 이전 배포의 일부가 아닌 파일이 대상 배포 위치에 표시되는 경우 다음 배포 중에 CodeDeploy가 해당 파일을 사용하여 수행할 작업을 선택할 수 있습니다.

- 배포 실패 — 오류가 보고되고 배포 상태가 실패로 변경됩니다.
- 컨텐츠 덮어쓰기 — 애플리케이션 개정 버전의 파일 버전이 인스턴스에 이미 있는 버전을 대체합니다.
- 콘텐츠 유지 — 대상 위치의 파일이 유지되고 애플리케이션 개정 버전의 버전이 인스턴스에 복사되지 않습니다.

배포를 생성할 때 이 동작을 선택할 수 있습니다. 콘솔에서 배포를 생성하는 경우 EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔) 단원을 참조하세요. 를 사용하여 배포를 생성하는 경우 섹션을 AWS CLI참조하세요EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(CLI).

애플리케이션 개정 패키지에 추가하지 않고 다음 배포의 일부로 포함하려는 파일을 보관하도록 선택할 수 있습니다. 예를 들어 배포에 필요하지만 애플리케이션 개정 번들에 추가되지 않은 파일을 인스턴스에 직접 업로드할 수 있습니다. 또는 애플리케이션이 이미 프로덕션 환경에 있지만 CodeDeploy를 사용하여 처음으로 배포하려는 경우 인스턴스에 파일을 업로드할 수 있습니다.

배포 실패로 인해 가장 최근에 성공적으로 배포된 애플리케이션 개정 버전이 다시 배포되는 롤백의 경우 마지막으로 성공한 배포에 대한 콘텐츠 처리 옵션이 롤백 배포에 적용됩니다.

그러나 실패한 배포가 파일을 보존하는 대신 덮어쓰도록 구성된 경우 롤백 중에 예기치 않은 결과가 발생할 수 있습니다. 특히, 보존해야 할 파일이 실패한 배포에 의해 제거될 수 있습니다. 롤백 배포가 실행될 때 파일이 인스턴스에 없습니다.

다음 예제에는 3가지 배포가 있습니다. 실패한 두 번째 배포 중에 덮어쓰여진(삭제된) 모든 파일은 배포 3 중에 애플리케이션 개정 버전 1이 다시 배포될 때 더 이상 사용할 수 없습니다(보존할 수 없음).

배포	애플리케이션 개정 버전	컨텐츠 덮어쓰기 옵션	배포 상태	동작 및 결과
배포 1	애플리케이션 개정 버전 1	RETAIN	성공	CodeDeploy는 이전 배포에서 배포하지 않은 대상 위치에서 파일을 검색합니다. 이러한 파일은 의도적으로 현재 배치의 일부가 될 수 있습니다. 현재 배포 패키지의 일부로 유지 및 기록됩니다.
배포 2	애플리케이션 개정 버전 2	OVERWRITE	실패	배포 프로세스 중에 CodeDeploy는 이전에 성공한 배포의 일부인 모든 파일을 삭제합니다. 여기에는 배치 1 중에 보존

배포	애플리케이션 개정 버전	컨텐츠 덮어쓰기 옵션	배포 상태	동작 및 결과
				된 파일이 포함됩니다. 그러나 무관한 이유로 배포가 실패합니다.
배포 3	애플리케이션 개정 버전 1	RETAIN		배포 또는 배포 그룹에 대해 자동 롤백이 사용되므로 CodeDeploy는 마지막으로 알려진 정상 애플리케이션 개정 버전인 애플리케이션 개정 버전 1을 배포합니다. 그러나 배포 1에 보관하려는 파일은 배포 2가 실패하기 전에 삭제되었으며 검색할 수 없습니다 AWS CodeDeploy. 애플리케이션 개정 버전 1에 필요한 경우 인스턴스에 직접 추가하거나 새 애플리케이션 개정 버전을 생성할 수 있습니다.

다른 AWS 계정에 애플리케이션 배포

조직에는 일반적으로 다양한 목적으로 사용하는 여러 계정이 있습니다(예: 시스템 관리 작업용 계정과 개발, 테스트 및 프로덕션 작업용 계정 또는 개발 및 테스트 환경과 연결된 AWS 계정과 프로덕션 환경과 연결된 계정).

서로 다른 계정에서 관련 작업을 수행할 수도 있지만, CodeDeploy 배포 그룹과 배포하는 것에 대한 Amazon EC2 인스턴스는 해당 인스턴스가 생성된 계정과 엄밀하게 연결됩니다. 예를 들어 한 계정에서 시작한 인스턴스를 다른 계정의 배포 그룹에 추가할 수 없습니다.

개발 AWS 계정과 프로덕션 계정이라는 두 개의 계정이 있다고 가정합니다. 주로 개발 계정에서 작업하지만, 전체 자격 증명 세트 없이 또는 개발 계정에서 로그아웃하여 프로덕션 계정에 로그인하지 않고도 프로덕션 계정에서 배포를 시작할 수 있기를 원합니다.

교차 계정 구성 단계를 수행한 후 다른 계정에 대한 전체 자격 증명 세트 없이 조직의 다른 계정에 속하는 배포를 시작할 수 있습니다. 부분적으로 해당 계정에 대한 임시 액세스 권한을 부여하는 AWS Security Token Service (AWS STS)에서 제공하는 기능을 사용하여 이 작업을 수행합니다.

1단계: 두 계정 중 하나에 S3 버킷 생성

개발 계정 또는 프로덕션 계정에서 다음을 수행합니다.

- 아직 생성하지 않은 경우 프로덕션 계정에 대한 애플리케이션 개정 버전이 저장될 Amazon S3 버킷을 생성합니다. 자세한 정보는 Amazon S3에서 버킷 생성을 참조하세요. 두 계정 모두에 동일한 버킷 및 애플리케이션 개정 버전을 사용하여 개발 계정에서 테스트하고 확인한 동일한 파일을 프로덕션 환경에 배포할 수도 있습니다.

2단계: Amazon S3 버킷에 프로덕션 계정의 IAM 인스턴스 프로필에 대한 권한 부여

1단계에서 생성한 Amazon S3 버킷이 프로덕션 계정에 있는 경우 이 단계가 필요하지 않습니다. 나중에 수임하는 역할은 프로덕션 계정에도 있기 때문에 이미 이 버킷에 액세스할 수 있습니다.

개발 계정에서 Amazon S3 버킷을 생성한 경우 다음을 수행합니다.

- 프로덕션 계정에서 IAM 인스턴스 프로파일을 생성합니다. 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 단원을 참조하세요.

> **Note**
>
> 이 IAM 인스턴스 프로파일의 ARN을 기록해 둡니다. 다음에 생성하는 교차 버킷 정책에 추가해야 합니다.

- 개발 계정에서 개발 계정으로 생성한 Amazon S3 버킷에 대한 액세스 권한을 프로덕션 계정에서 방금 생성한 IAM 인스턴스 프로파일에 부여합니다. 자세한 내용은 예제 2: 버킷 소유자가 교차 계정 버킷 권한 부여를 참조하세요.

  교차 계정 버킷 권한 부여 프로세스를 완료할 때 다음 사항에 유의하세요.

  - 예제 연습에서 계정 A는 개발 계정을 나타내고 계정 B는 프로덕션 계정을 나타냅니다.
  - 계정 A(개발 계정) 작업에서 다음 버킷 정책을 수정하여 연습에 제공된 정책 예제를 사용하는 대신 교차 계정 권한을 부여하도록 합니다.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Cross-account permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/role-name"
            },
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

    *account-id*는 방금 IAM 인스턴스 프로파일을 생성한 프로덕션 계정의 계정 번호를 나타냅니다.

    *role-name*은 방금 생성한 IAM 인스턴스 프로파일의 이름을 나타냅니다.

*bucket-name*은 1단계에서 생성한 버킷의 이름을 나타냅니다. 버킷 내의 각 파일에 대한 액세스를 제공할 수 있도록 버킷 이름 뒤에 /*을(를) 포함해야 합니다.

3단계: 프로덕션 계정에서 리소스 및 교차 계정 역할 생성

프로덕션 계정에서 다음을 수행합니다.

- 이 가이드의 지침에 따라 CodeDeploy 리소스(애플리케이션, 배포 그룹, 배포 구성, Amazon EC2 인스턴스, Amazon EC2 인스턴스 프로파일, 서비스 역할 등)를 생성합니다.
- 개발 계정의 사용자가 이 프로덕션 계정에서 CodeDeploy 작업을 수행하도록 수임할 수 있는 추가 역할인 교차 계정 IAM 역할을 생성합니다.

  연습: IAM 역할을 사용하여 AWS 계정 간 액세스 권한 위임을 가이드로 사용하여 교차 계정 역할을 생성할 수 있습니다. 연습의 샘플 권한을 정책 문서에 추가하는 대신 최소한 다음 두 개의 AWS 제공된 정책을 역할에 연결해야 합니다.

  - `AmazonS3FullAccess`: S3 버킷이 개발 계정에 있는 경우에만 필요합니다. 위임된 프로덕션 계정 역할에 개정 버전이 저장된 개발 계정의 Amazon S3 서비스 및 리소스에 대한 전체 액세스 권한을 제공합니다.
  - `AWSCodeDeployDeployerAccess`: 사용자가 개정 버전을 등록 및 배포할 수 있도록 합니다.

  배포를 시작하는 것만이 아니라 배포 그룹을 만들고 관리하려는 경우 `AWSCodeDeployDeployerAccess` 정책 대신 `AWSCodeDeployFullAccess` 정책을 추가하세요. IAM 관리형 정책을 사용하여 CodeDeploy 작업에 대한 권한을 부여하는 방법에 대한 자세한 내용은 AWS CodeDeploy에 대한 관리형(미리 정의된) 정책 단원을 참조하세요.

  이 교차 계정 역할을 사용하는 동안 다른 AWS 서비스에서 작업을 수행하려는 경우 추가 정책을 연결할 수 있습니다.

> **Important**
>
> 교차 계정 IAM 역할을 생성할 때 프로덕션 계정에 액세스하는 데 필요한 세부 정보를 기록해 둡니다.
> AWS Management Console 를 사용하여 역할을 전환하려면 다음 중 하나를 제공해야 합니다.
>
> - 위임된 역할의 자격 증명을 사용하여 프로덕션 계정에 액세스하기 위한 URL. 검토 페이지에서 이 URL을 찾을 수 있습니다. 이는 교차 계정 역할 생성 프로세스가 끝날 때 표시됩니다.

- 교차 계정 역할의 이름과 계정 ID 번호 또는 별칭.

AWS CLI 를 사용하여 역할을 전환하려면 다음을 제공해야 합니다.

- 위임할 교차 계정 역할의 ARN.

4단계: Amazon S3 버킷에 애플리케이션 개정 버전 업로드.

Amazon S3 버킷을 생성한 계정에서 다음을 수행합니다.

- Amazon S3 버킷에 애플리케이션 개정 버전을 업로드합니다. 자세한 내용은 Amazon S3에 CodeDeploy의 개정 푸시(EC2 온프레미스 배포 전용) 단원을 참조하세요.

5단계: 교차 계정 역할을 위임하고 애플리케이션 배포

개발 계정에서 AWS CLI 또는를 사용하여 교차 계정 역할을 AWS Management Console 수임하고 프로덕션 계정에서 배포를 시작할 수 있습니다.

를 사용하여 역할을 AWS Management Console 전환하고 배포를 시작하는 방법에 대한 지침은 역할로 전환(AWS Management Console) 및 섹션을 참조하세요EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔).

AWS CLI 를 사용하여 교차 계정 역할을 수임하고 배포를 시작하는 방법에 대한 지침은 IAM 역할로 전환(AWS Command Line Interface) 및 섹션을 참조하세요EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(CLI).

역할 수임에 대한 자세한 AWS STS내용은 AWS Security Token Service 사용 설명서의 AssumeRole 및 AWS CLI 명령 참조의 assume-role을 참조하세요.

관련 주제:

- CodeDeploy: 개발 계정에서 프로덕션 계정으로 배포

CodeDeploy 에이전트를 사용하여 로컬 시스템에서 배포 패키지의 유효성을 검사합니다.

CodeDeploy 에이전트를 사용하여 로그인한 인스턴스에 콘텐츠를 배포할 수 있습니다. 이를 통해 배포에 사용하려는 애플리케이션 사양 파일(AppSpec 파일)의 무결성 및 배포하려는 콘텐츠를 테스트할 수 있습니다.

애플리케이션 및 배포 그룹을 만들 필요가 없습니다. 로컬 인스턴스에 저장된 콘텐츠를 배포하려는 경우 AWS 계정이 필요하지 않습니다. 가장 간단한 테스트를 위해 AppSpec 파일과 배포할 콘텐츠가 포함된 디렉터리에서 옵션을 지정하지 않고 codedeploy-agent deploy-local 명령을 실행할 수 있습니다. 도구에는 다른 테스트 사례에 대한 옵션이 있습니다.

로컬 컴퓨터에서 배포 패키지의 유효성을 검사하여 다음을 수행할 수 있습니다.

- 애플리케이션 개정 버전의 무결성 테스트.
- AppSpec 파일의 내용을 테스트합니다.
- 처음으로 기존 애플리케이션 코드로 CodeDeploy를 사용해 보세요.
- 인스턴스에 이미 로그인하지 않았다면 콘텐츠를 빠르게 배포합니다.

로컬 인스턴스 또는 지원되는 원격 리포지토리 유형(Amazon S3 버킷 또는 퍼블릭 GitHub 리포지토리)에 저장된 콘텐츠를 배포할 수 있습니다.

사전 조건

로컬 배포를 시작하기 전에 다음 단계를 완료해야 합니다.

- CodeDeploy 에이전트에서 지원하는 인스턴스 유형을 생성하거나 사용합니다. 자세한 내용은 CodeDeploy 에이전트에서 지원하는 운영 체제 단원을 참조하세요.
- CodeDeploy 에이전트 버전 2.0.0 이상을 설치합니다. `deploy-local` 하위 명령에는 버전 2.0.0 이상이 필요합니다. 버전 1.8.x 이하의 경우 독립 실행형 `codedeploy-local` 명령을 대신 사용합니다. 버전 정보는 CodeDeploy 에이전트의 버전 기록를 참조하세요. 자세한 설치 정보는 CodeDeploy 에이전트 설치 단원을 참조하십시오.
- Amazon S3 버킷 또는 GitHub 리포지토리에서 콘텐츠를 배포하는 경우 CodeDeploy와 함께 사용하도록 사용자를 프로비저닝합니다. 자세한 내용은 1단계: 설정 단원을 참조하세요.
- Amazon S3 버킷에서 애플리케이션 개정을 배포하는 경우 작업 중인 리전에 Amazon S3 버킷을 생성하고 버킷에 Amazon S3 버킷 정책을 적용합니다. 이 정책은 애플리케이션 개정을 다운로드하는 데 필요한 권한을 인스턴스에 부여합니다.

예를 들어, 다음 Amazon S3 버킷 정책은 ARN `arn:aws:iam::444455556666:role/CodeDeployDemo`이(가) 포함된 IAM 인스턴스 프로필이 연결되어 있는 Amazon EC2 인스턴스가 이름이 `amzn-s3-demo-bucket`인 Amazon S3 버킷에서 위치에 관계없이 다운로드할 수 있도록 허용합니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/CodeDeployDemo"
                ]
            }
        }
    ]
}
```

다음 Amazon S3 버킷 정책은 ARN `arn:aws:iam::444455556666:user/CodeDeployUser`이(가) 포함된 IAM 사용자가 연결되어 있는 온프레미스 인스턴스가 이름이 `amzn-s3-demo-bucket`인 Amazon S3 버킷에서 위치에 관계없이 다운로드할 수 있도록 허용합니다.

```
{
    "Statement": [
        {
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:user/CodeDeployUser"
                ]
            }
```

```
        }
    ]
}
```

Amazon S3 버킷 정책 생성 및 연결에 대한 자세한 내용은 버킷 정책 예제를 참조하세요.

- Amazon S3 버킷 또는 GitHub 리포지토리에서 애플리케이션 개정 버전을 배포하는 경우 IAM 인스턴스 프로파일을 설정하고 인스턴스에 연결합니다. 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기, CodeDeploy용 Amazon EC2 인스턴스 생성(AWS CLI 또는 Amazon EC2 콘솔), 및 CodeDeploy용 Amazon EC2 인스턴스 생성(CloudFormation 템플릿)을(를) 참조하세요.
- GitHub에서 콘텐츠를 배포하는 경우 GitHub 계정과 퍼블릭 리포지토리를 만듭니다. GitHub 계정을 만들려면 GitHub 조인을 참조하세요. GitHub 리포지토리를 만들려면 리포지토리 생성을 참조하세요.

> **Note**
>
> 프라이빗 리포지토리는 현재 지원되지 않습니다. 콘텐츠가 프라이빗 GitHub 리포지토리에 저장되어 있는 경우 해당 콘텐츠를 인스턴스로 다운로드하고 `--bundle-location` 옵션을 사용하여 로컬 경로를 지정할 수 있습니다.

- 인스턴스에 배포하려는 콘텐츠(AppSpec 파일 포함)를 준비하고 로컬 인스턴스, Amazon S3 버킷 또는 GitHub 리포지토리에 배치합니다. 자세한 내용은 CodeDeploy의 애플리케이션 개정 작업 단원을 참조하세요.
- 다른 구성 옵션의 기본값 이외의 값을 사용하려면 구성 파일을 생성하여 인스턴스(Amazon Linux, RHEL 또는 Ubuntu Server 인스턴스의 경우 `/etc/codedeploy-agent/conf/codedeployagent.yml`, Windows Server 인스턴스의 경우 `C:\ProgramData\Amazon\CodeDeploy\conf.yml`)에 배치합니다. 자세한 내용은 CodeDeploy 에이전트 구성 참조 단원을 참조하세요.

> **Note**
>
> Amazon Linux, RHEL 또는 Ubuntu Server 인스턴스에서 구성 파일을 사용하는 경우 다음 중 하나를 수행해야 합니다.
>
> - `:root_dir:`와(과) `:log_dir:` 변수를 사용하여 배포 루트 및 로그 디렉토리 폴더의 기본값 이외의 위치를 지정할 수 있습니다.
> - `sudo`을(를) 사용하여 CodeDeploy 에이전트 명령을 실행합니다.

로컬 배포 생성

로컬 배포를 생성하려는 인스턴스에서 터미널 세션(Amazon Linux, RHEL 또는 Ubuntu Server 인스턴스) 또는 명령 프롬프트(Windows Server)를 열어 도구 명령을 실행합니다.

> **Note**
>
> codedeploy-agent deploy-local 명령은 다음 위치에서 `codedeploy-agent` 바이너리를 사용합니다.
>
> - Amazon Linux, RHEL 또는 Ubuntu Server의 경우: `/opt/codedeploy-agent/bin/codedeploy-agent`.
> - Windows Server의 경우: `C:\ProgramData\Amazon\CodeDeploy\bin\codedeploy-agent.exe`.
>
> 이전 버전과의 호환성을 위해 동일한 위치에서 codedeploy-local 명령을 사용할 수도 있습니다.
>
> - Amazon Linux, RHEL 또는 Ubuntu Server에서 명령은에 있습니다`/opt/codedeploy-agent/bin/codedeploy-local`.
> - Windows Server에서 명령은에 있습니다`C:\ProgramData\Amazon\CodeDeploy\bin\codedeploy-local.exe`.
>
> 구문은 계속 작동하며 codedeploy-local [options]와 동일한 옵션을 허용합니다codedeploy-agent deploy-local.

기본 명령 구문

```
codedeploy-agent deploy-local [options]
```

시놉시스

```
codedeploy-agent deploy-local
[--bundle-location <value>]
[--type <value>]
[--file-exists-behavior <value>]
[--deployment-group <value>]
```

```
[--deployment-group-name <value>]
[--application-name <value>]
[--events <comma-separated values>]
[--agent-configuration-file <value>]
[--appspec-filename <value>]
```

옵션

-l, --bundle-location

애플리케이션 개정 버전 번들의 위치. 위치를 지정하지 않으면 도구는 기본적으로 현재 작업 중인 디렉토리를 사용합니다. `--bundle-location`의 값을 지정하면 `--type`에 대한 값도 함께 지정해야 합니다.

번들 위치 형식 예:

- Local Amazon Linux, RHEL 또는 Ubuntu Server 인스턴스: `/path/to/local/bundle.tgz`
- 로컬 Windows Server 인스턴스: `C:/path/to/local/bundle`
- Amazon S3 버킷: `s3://amzn-s3-demo-bucket/bundle.tar`
- GitHub 리포지토리: `https://github.com/`*`account-name`*`/`*`repository-name`*`/`

-t,—type

애플리케이션 개정 버전 번들의 형식. 지원되는 형식은 `tgz`, `tar`, `zip`, `directory`입니다. 유형을 지정하지 않을 경우 도구는 기본적으로 `directory`을(를) 사용합니다. `--type`의 값을 지정하면 `--bundle-location`에 대한 값도 함께 지정해야 합니다.

-b, --file-exists-behavior

배포 대상 위치에 이미 있지만 이전에 성공한 배포의 일부가 아닌 파일이 처리되는 방식을 나타냅니다. 옵션에는 DISALLOW, OVERWRITE, RETAIN가 있습니다. 자세한 내용은 AWS CodeDeploy API 참조의 fileExistsBehavior을 참조하세요.

-g, --deployment-group

배포할 콘텐츠의 대상 위치인 폴더의 경로. 폴더를 지정하지 않을 경우 도구는 배포 루트 디렉토리 내에서 default-local-deployment-group이라는 이름의 폴더를 생성합니다. 생성하는 각 로컬 배포에 대해 도구는 local-1234(숫자가 프로세스 ID)와 같은 이름으로이 폴더 내에 하위 디렉터리를 생성합니다.

-d, --deployment-group-name

수명 주기 후크 스크립트에 DEPLOYMENT_GROUP_NAME 환경 변수로 노출된 배포 그룹 이름입니다. 기본값은 LocalFleet입니다.

-a, --application-name

수명 주기 후크 스크립트에 APPLICATION_NAME 환경 변수로 표시되는 애플리케이션 이름입니다. 기본값은 번들 위치입니다.

-e, --events

AppSpec 파일에 나열된 이벤트 대신 순서대로 실행하려는 재정의 수명 주기 이벤트 후크 세트. 쉼표로 구분된 여러 후크를 지정할 수 있습니다. 다음 경우에 이 옵션을 사용할 수 있습니다.

- AppSpec 파일을 업데이트하지 않고 다른 이벤트 집합을 실행하려고 합니다.
- AppSpec 파일에 있는 항목(예: ApplicationStop)에 대한 예외로 단일 이벤트 후크를 실행하려고 합니다

재정의 목록에 DownloadBundle 및 Install 이벤트를 지정하지 않는 경우 지정한 모든 이벤트 후크 전에 실행됩니다. --events 옵션 목록에 DownloadBundle과 Install이 포함되어 있는 경우 CodeDeploy 배포에서 일반적으로 해당 항목 이전에 실행되는 이벤트만 해당 항목 앞에 와야 합니다. 자세한 내용은 AppSpec 'hooks' 섹션 단원을 참조하세요.

-c, --agent-configuration-file

기본 위치가 아닌 다른 위치에 저장하는 경우 배포에 사용할 구성 파일의 위치. 구성 파일은 배포에 대한 다른 기본값 및 동작에 대한 대안을 지정합니다.

기본적으로 구성 파일은 /etc/codedeploy-agent/conf/codedeployagent.yml(Amazon Linux, RHEL 또는 Ubuntu Server 인스턴스) 또는 C:/ProgramData/Amazon/CodeDeploy/conf.yml(Windows Server)에 저장됩니다. 자세한 내용은 CodeDeploy 에이전트 구성 참조 단원을 참조하십시오.

-A, --appspec-filename

AppSpec 파일의 이름입니다. 로컬 배포의 경우 허용되는 값은 appspec.yml 및 appspec.yaml입니다. 기본적으로 AppSpec 파일의 이름은 appspec.yml입니다.

-h, --help

도움말 컨텐츠의 요약을 표시합니다.

-v, --version

도구의 버전 번호를 표시합니다.

예제

다음은 유효한 명령 형식의 예입니다.

```
codedeploy-agent deploy-local
```

```
codedeploy-agent deploy-local --bundle-location /path/to/local/bundle/directory
```

```
codedeploy-agent deploy-local --bundle-location C:/path/to/local/bundle.zip --type zip
 --deployment-group my-deployment-group
```

```
codedeploy-agent deploy-local --bundle-location /path/to/local/directory --type
 directory --deployment-group my-deployment-group
```

Amazon S3에서 번들 배포:

```
codedeploy-agent deploy-local --bundle-location s3://amzn-s3-demo-bucket/bundle.tgz --
type tgz
```

```
codedeploy-agent deploy-local --bundle-location s3://amzn-s3-demo-bucket/bundle.zip?
versionId=1234&etag=47e8 --type zip --deployment-group my-deployment-group
```

퍼블릭 GitHub 리포지토리에서 번들 배포:

```
codedeploy-agent deploy-local --bundle-location https://github.com/awslabs/aws-
codedeploy-sample-tomcat --type zip
```

```
codedeploy-agent deploy-local --bundle-location https://api.github.com/repos/awslabs/
aws-codedeploy-sample-tomcat/zipball/master --type zip
```

```
codedeploy-agent deploy-local --bundle-location https://api.github.com/repos/awslabs/
aws-codedeploy-sample-tomcat/zipball/HEAD --type zip
```

```
codedeploy-agent deploy-local --bundle-location https://api.github.com/repos/awslabs/
aws-codedeploy-sample-tomcat/zipball/1a2b3c4d --type zip
```

여러 수명 주기 이벤트를 지정하는 번들 배포:

```
codedeploy-agent deploy-local --bundle-location /path/to/local/
bundle.tar --type tar --deployment-group my-deployment-group --events
 DownloadBundle,Install,ApplicationStart,HealthCheck
```

ApplicationStop 수명 주기 이벤트를 사용하여 이전에 배포한 애플리케이션 중지:

```
codedeploy-agent deploy-local --bundle-location /path/to/local/bundle.tgz --type tgz --
deployment-group my-deployment-group --events ApplicationStop
```

특정 배포 그룹 ID를 사용하여 배포:

```
codedeploy-agent deploy-local --bundle-location C:/path/to/local/bundle/directory --
deployment-group 1234abcd-5dd1-4774-89c6-30b107ac5dca
```

```
codedeploy-agent deploy-local --bundle-location C:/path/to/local/bundle.zip --type zip
 --deployment-group 1234abcd-5dd1-4774-89c6-30b107ac5dca
```

CodeDeploy에서 배포 모니터링

모니터링은 CodeDeploy 및 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 다중 지점 장애가 발생할 경우 보다 쉽게 디버깅할 수 있도록 AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다. 하지만 CodeDeploy 모니터링을 시작하기 전에 다음 질문에 대한 답변을 포함하는 모니터링 계획을 작성해야 합니다.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

다음 단계에서는 다양한 시간과 다양한 부하 조건에서 성능을 측정하여 환경에서 일반 CodeDeploy 성능의 기준선을 설정합니다. CodeDeploy가 과거 모니터링 데이터를 저장하는 것을 모니터링하면서 현재 성능 데이터를 이 과거 데이터와 비교하면 일반적인 성능 패턴과 성능 이상을 식별하고 이를 해결할 방법을 고안할 수 있습니다.

예를 들어 CodeDeploy를 사용하는 경우 배포 및 대상 인스턴스 상태를 모니터링할 수 있습니다. 배포 또는 인스턴스가 실패할 경우 애플리케이션 사양 파일을 재구성하거나, CodeDeploy 에이전트를 다시 설치 또는 업데이트하거나, 애플리케이션 또는 배포 그룹의 설정을 업데이트하거나, 인스턴스 설정 또는 AppSpec 파일을 변경해야 할 수 있습니다.

기준선을 설정하려면 최소한 다음 항목을 모니터링해야 합니다.

- 배포 이벤트 및 상태
- 인스턴스 이벤트 및 상태

자동 모니터링 도구

AWS 는 CodeDeploy를 모니터링하는 데 사용할 수 있는 다양한 도구를 제공합니다. 이들 도구 중에는 모니터링을 자동으로 수행하도록 구성할 수 있는 도구도 있지만, 수동 작업이 필요한 도구도 있습니다. 모니터링 작업은 최대한 자동화하는 것이 좋습니다.

다음과 같은 자동 모니터링 도구를 사용하여 CodeDeploy를 관찰하고 문제 발생 시 보고할 수 있습니다.

- Amazon CloudWatch 경보 – 지정한 기간 동안 단일 지표를 감시하고, 여러 기간에 대해 지정된 임곗값과 관련하여 지표 값을 기준으로 하나 이상의 작업을 수행합니다. 이 작업은 Amazon Simple Notification Service(Amazon SNS) 주제 또는 Amazon EC2 Auto Scaling 정책에 전송되는 알림입니다. CloudWatch 경보는 특정 상태에 있다는 이유만으로는 작업을 호출하지 않습니다. 상태가 변경되고 지정한 기간 동안 유지되어야 합니다. 자세한 내용은 Monitoring Deployments with Amazon CloudWatch Tools 단원을 참조하십시오.

  CloudWatch 경보 모니터링을 사용하도록 서비스 역할을 업데이트하는 방법에 대한 자세한 내용은 CodeDeploy 서비스 역할에 CloudWatch 권한 부여 단원을 참조하세요. CloudWatch 경보 모니터링을 CodeDeploy 작업에 추가하는 방법에 대한 자세한 내용은 CodeDeploy를 사용하여 애플리케이션 생성, CodeDeploy에서 배포 그룹 만들기 또는 CodeDeploy에서 배포 그룹 설정 변경 단원을 참조하세요.

- Amazon CloudWatch Logs – AWS CloudTrail 또는 기타 소스의 로그 파일을 모니터링, 저장 및 액세스합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 로그 파일 모니터링을 참조하세요.

  CloudWatch 콘솔을 사용하여 CodeDeploy 로그를 보는 방법에 대한 자세한 내용은 CloudWatch Logs 콘솔에서 CodeDeploy 로그 보기를 참조하세요.

- Amazon CloudWatch Events – 이벤트를 일치시키고 하나 이상의 대상 함수 또는 스트림으로 라우팅하여 값을 변경하거나 상태 정보를 캡처하거나 수정 작업을 수행합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch Events란 무엇인가요를 참조하세요.

  CodeDeploy 작업에서 CloudWatch Events를 사용하는 방법에 대한 자세한 내용은 Amazon CloudWatch Events를 사용하여 배포 모니터링 단원을 참조하세요.

- AWS CloudTrail 로그 모니터링 - 계정 간에 로그 파일을 공유하고, CloudTrail 로그 파일을 CloudWatch Logs로 전송하여 실시간으로 모니터링하고, Java로 로그 처리 애플리케이션을 작성하고, CloudTrail에서 전송한 후 로그 파일이 변경되지 않았는지 확인합니다. 자세한 내용은AWS CloudTrail 사용 설명서의 CloudTrail 로그 파일 작업을 참조하세요.

  CodeDeploy에서 CloudTrail 사용에 관한 자세한 내용은 Monitoring Deployments 단원을 참조하세요.

- Amazon Simple Notification Service — 성공 또는 실패와 같은 배포 및 인스턴스 이벤트에 대한 SMS 또는 이메일 알림을 수신하도록 이벤트 기반 트리거를 구성합니다. 자세한 내용은 주제 생성 및 Amazon Simple Notification Service는 무엇입니까를 참조하세요.

CodeDeploy에 대한 Amazon SNS 알림을 설정하는 방법에 대한 자세한 내용은 Monitoring Deployments with Amazon SNS Event Notifications 단원을 참조하세요.

수동 모니터링 도구

CodeDeploy 모니터링의 또 한 가지 중요한 부분은 CloudWatch 경보에 포함되지 않는 항목을 수동으로 모니터링해야 한다는 점입니다. CodeDeploy, CloudWatch 및 기타 AWS 콘솔 대시보드는 AWS 환경 상태를 at-a-glance 볼 수 있습니다. 또한 CodeDeploy 배포에서 로그 파일을 확인하는 것이 좋습니다.

- CodeDeploy 콘솔에는 다음이 표시됩니다.
  - 배포 상태
  - 수정 버전의 마지막 배포 시도 및 마지막으로 성공한 각 날짜 및 시간
  - 배포에서 성공, 실패, 건너뛴 인스턴스 또는 진행 중인 인스턴스 수
  - 온프레미스 인스턴스 상태
  - 온프레미스 인스턴스 등록 또는 등록 취소된 날짜 및 시간
- CloudWatch 홈 페이지에 표시되는 항목은 다음과 같습니다.
  - 현재 경보 및 상태
  - 경보 및 리소스 그래프
  - 서비스 상태

  또한 CloudWatch를 사용하여 다음을 수행할 수 있습니다.
  - 맞춤 대시보드를 생성하여 관심 있는 서비스 모니터링
  - 지표 데이터를 그래프로 작성하여 문제를 해결하고 추세 파악
  - 모든 AWS 리소스 지표 검색 및 찾아보기
  - 문제에 대해 알려주는 경보 생성 및 편집

주제

- Monitoring Deployments with Amazon CloudWatch Tools
- Monitoring Deployments
- Monitoring Deployments with Amazon SNS Event Notifications

Amazon CloudWatch 도구를 사용하여 배포 모니터링

CodeDeploy 배포는 Amazon CloudWatch Events, CloudWatch 경보 및 Amazon CloudWatch Logs 등의 CloudWatch 도구를 사용하여 모니터링할 수 있습니다.

CodeDeploy 에이전트 및 배포에서 생성된 로그를 검토하면 배포 실패의 원인을 파악해 문제를 해결할 수 있습니다. 한 번에 하나의 인스턴스에 대해 CodeDeploy 로그를 검토하는 대신 CloudWatch Logs를 사용하여 중앙 위치에서 모든 로그를 모니터링할 수 있습니다.

CloudWatch 경보 및 CloudWatch Events를 사용하여 CodeDeploy의 배포를 모니터링하는 방법을 자세히 알아보려면 다음 주제를 참조하세요.

주제

- CodeDeploy에서 CloudWatch 경보를 사용하여 배포 모니터링
- Amazon CloudWatch Events를 사용하여 배포 모니터링

CodeDeploy에서 CloudWatch 경보를 사용하여 배포 모니터링

CodeDeploy 작업에서 사용 중인 인스턴스 또는 Amazon EC2 Auto Scaling 그룹에 대한 CloudWatch 경보를 만들 수 있습니다. 경보는 지정한 기간에 단일 지표를 감시하고 여러 기간에 지정된 임계값에 대한 지표 값을 기준으로 작업을 하나 이상 수행합니다. CloudWatch 경보는 상태가 변경(예: `OK`에서 `ALARM`으로)될 때 작업을 호출합니다.

기본 CloudWatch 경보 기능을 사용하면 배포에 사용 중인 인스턴스가 실패할 경우 CloudWatch에서 지원하는 어느 작업이든지 지정할 수 있습니다(예: Amazon SNS 알림 보내기 또는 인스턴스 중지, 종료, 재부팅 또는 복구). CodeDeploy 작업의 경우 배포 그룹과 연결한 CloudWatch 경보 중 어느 것이든지 활성화될 때마다 배포가 중지되도록 배포 그룹을 구성할 수 있습니다.

CodeDeploy 배포 그룹 하나에 CloudWatch 경보를 최대 10개까지 연결할 수 있습니다. 지정한 경보 중 하나가 활성화되면 배포가 중지되고 상태가 중지됨으로 업데이트됩니다. 이 옵션을 사용하려면 CodeDeploy 서비스 역할에 CloudWatch 권한을 부여해야 합니다.

CloudWatch 콘솔에서 CloudWatch 경보를 설정하는 방법에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 생성을 참조하세요.

CodeDeploy에서 CloudWatch 경보를 배포 그룹과 연결하는 방법은 CodeDeploy에서 배포 그룹 만들기 및 CodeDeploy에서 배포 그룹 설정 변경 섹션을 참조하세요.

주제

- CodeDeploy 서비스 역할에 CloudWatch 권한 부여

CodeDeploy 서비스 역할에 CloudWatch 권한 부여

배포에 CloudWatch 경보 모니터링을 사용하려면 먼저 CodeDeploy 작업에서 사용하는 서비스 역할에 CloudWatch 리소스에 액세스할 수 있는 권한을 부여해야 합니다.

서비스 역할에 CloudWatch 권한 부여하기

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할을 선택합니다.
3. AWS CodeDeploy 작업에 사용하는 서비스 역할의 이름을 선택합니다.
4. [Permissions] 탭의 [Inline Policies]에서, [Create Role Policy]를 선택합니다.

   -또는-

   [Create Role Policy] 버튼을 사용할 수 없으면 [Inline Policies] 영역을 확장한 후 [click here]를 선택합니다.
5. [Set Permissions] 페이지에서 [Custom Policy]와 [Select]를 차례로 선택합니다.
6. 정책 검토 페이지의 정책 이름 필드에 이 정책을 식별할 수 있는 이름을 입력합니다(예: `CWAlarms`).
7. 다음 내용을 [Policy Document] 필드에 붙여 넣습니다.

   JSON

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

8. 정책 적용을 선택합니다.

Amazon CloudWatch Events를 사용하여 배포 모니터링

Amazon CloudWatch Events를 사용하여 CodeDeploy 작업에서 인스턴스 또는 배포의 상태 변화("이벤트")를 감지하고 그에 대응할 수 있습니다. 그런 다음 CloudWatch Events는 사용자가 만든 규칙에 따라 배포 또는 인스턴스가 규칙에서 지정된 상태로 전환되면 하나 이상의 대상 작업을 호출합니다. 상태 변경 유형에 따라 알림을 보내거나, 상태 정보를 캡처하거나, 교정 작업을 수행하거나, 이벤트를 시작하거나, 기타 작업을 수행할 수 있습니다. CloudWatch Events를 CodeDeploy 작업의 일부로 사용하는 경우 다음과 같은 대상 유형을 선택할 수 있습니다.

- AWS Lambda 함수
- Kinesis 스트림
- Amazon SQS 대기열
- 기본 제공 대상(`EC2 CreateSnapshot API call`, `EC2 RebootInstances API call`, `EC2 StopInstances API call`, `EC2 TerminateInstances API call`)
- Amazon SNS 주제

다음은 몇 가지 사용 사례입니다.

- 배포에 실패할 때마다 Lambda 함수를 사용하여 Slack 채널에 알림을 전달합니다.
- 배포 또는 인스턴스에 대한 데이터를 Kinesis 스트림으로 푸시하여 포괄적인 실시간 상태 모니터링을 지원합니다.
- CloudWatch 경보 작업을 사용하여 지정한 배포 또는 인스턴스 이벤트가 발생할 경우 Amazon EC2 인스턴스를 자동으로 중지, 종료, 재부팅 또는 복구할 수 있습니다.

이 주제의 나머지 부분에서는 CodeDeploy에 대한 CloudWatch Events 규칙을 만드는 기본 절차를 설명합니다. CodeDeploy 작업에 사용할 이벤트 규칙을 만들기 전에 다음을 수행해야 합니다.

- CloudWatch Events 사전 요구 사항을 완료합니다. 자세한 내용은 Amazon CloudWatch Events 사전 요구 사항을 참조하세요.
- CloudWatch Events의 이벤트, 규칙 및 대상을 익힙니다. 자세한 내용은 Amazon CloudWatch Events란 무엇입니까?를 참조하세요. 및 새 CloudWatch 이벤트 - AWS 리소스의 변경 사항을 추적하고 이에 대응합니다.
- 대상, 즉 이벤트 규칙에 사용할 대상을 생성합니다.

CodeDeploy에 대한 CloudWatch Events 규칙을 생성하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 이벤트를 선택합니다.
3. 규칙 생성(Create rule)을 선택한 다음 이벤트 선택기(Event selector)에서 AWS CodeDeploy를 선택합니다.
4. 세부 정보를 지정합니다.
   - 인스턴스 및 배포의 모든 상태 변경에 적용되는 규칙을 만들려면 모든 세부 정보 유형(Any detail type)을 선택한 다음 6단계로 건너뜁니다.
   - 인스턴스에만 적용되는 규칙을 만들려면 특정 상세 유형(Specific detail type)을 선택한 다음 CodeDeploy 인스턴스 상태 변경 알림(CodeDeploy Instance State-change Notification)을 선택합니다.
   - 배포에만 적용되는 규칙을 만들려면 특정 상세 유형(Specific detail type)을 선택한 다음 CodeDeploy 배포 상태 변경 알림(CodeDeploy Deployment State-change Notification)을 선택합니다.
5. 규칙을 적용할 상태 변경을 지정합니다.
   - 모든 상태 변경에 적용되는 규칙을 만들려면 모든 상태(Any state)를 선택합니다.
   - 일부 상태 변경에만 적용되는 규칙을 만들려면 특정 상태(Specific state(s))를 선택한 다음 목록에서 상태 값을 한 개 이상 선택합니다. 다음 표에서는 선택할 수 있는 상태 값을 나열합니다.

배포 상태 값	인스턴스 상태 값
실패	실패
START	START
중지(STOP)	준비
대기됨	성공
준비	
성공	

6. 규칙을 적용할 CodeDeploy 애플리케이션을 지정합니다.

- 모든 애플리케이션에 적용되는 규칙을 만들려면 모든 애플리케이션(Any application)을 선택한 다음 8단계로 건너뜁니다.
- 하나의 애플리케이션에만 적용되는 규칙을 만들려면 특정 애플리케이션(Specific application)을 선택한 다음 목록에서 애플리케이션의 이름을 선택합니다.

7. 규칙을 적용할 배포 그룹을 지정합니다.

   - 선택한 애플리케이션과 연결된 모든 배포 그룹에 적용되는 규칙을 만들려면 모든 배포 그룹(Any deployment group)을 선택합니다.
   - 선택한 애플리케이션과 연결된 배포 그룹 중 하나에만 적용되는 규칙을 만들려면 특정 배포 그룹(Specific deployment group(s))을 선택한 후 목록에서 배포 그룹의 이름을 선택합니다.

8. 이벤트 모니터링 요구 사항을 충족하도록 규칙 설정을 검토하세요.
9. 대상(Targets) 영역에서 대상 추가(Add target)*를 선택하세요.
10. [Select target type] 목록에서 이 규칙을 사용하도록 설정한 대상 유형을 선택한 후, 해당 유형에 필요한 모든 추가 옵션을 구성합니다.
11. 세부 정보 구성을 선택합니다.
12. 규칙 세부 정보 구성 페이지에서 규칙의 이름과 설명을 입력한 다음 상태 상자를 선택하여 지금 규칙을 활성화합니다.
13. 규칙이 만족스러우면 규칙 생성(Create rule)을 선택하세요.

를 사용하여 배포 모니터링 AWS CloudTrail

CodeDeploy는 AWS 계정에서 CodeDeploy에 의해 또는 이를 대신하여 수행된 API 호출을 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송하는 서비스인 CloudTrail과 통합됩니다. CloudTrail은 CodeDeploy 콘솔,를 통한 CodeDeploy 명령 AWS CLI또는 CodeDeploy API에서 직접 APIs. CloudTrail에서 수집하는 정보를 사용하여 CodeDeploy에 어떤 요청이 이루어졌는지, 어떤 소스 IP 주소에서 요청했는지, 누가 언제 요청했는지 등을 확인할 수 있습니다. 구성 및 사용 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 CodeDeploy 정보

AWS 계정에서 CloudTrail 로깅이 활성화되면 CodeDeploy 작업에 대한 API 호출이 로그 파일에서 추적됩니다. CodeDeploy 레코드는 로그 파일의 다른 AWS 서비스 레코드와 함께 작성됩니다. CloudTrail은 기간 및 파일 크기를 기준으로 새 파일을 만들고 기록하는 시점을 결정합니다.

모든 CodeDeploy 작업은 AWS CodeDeploy 명령줄 레퍼런스 및 AWS CodeDeploy API 참조에 로깅되고 문서화됩니다. 예를 들어, 배포를 생성하고, 애플리케이션을 삭제하고, 애플리케이션 개정 버전을 등록하도록 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 로그 항목은 누가 요청을 생성했는가에 대한 정보가 들어 있습니다. 로그의 사용자 자격 증명 정보는 요청이 루트 또는 사용자 자격 증명, 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명 또는 다른 AWS 서비스에 의해 이루어졌는지 확인하는 데 도움이 됩니다. 자세한 내용은 CloudTrail 이벤트 참조의 userIdentity 필드를 참조하세요.

원하는 기간만큼 버킷에 로그 파일을 저장할 수 있습니다. 그러나 Amazon S3 수명 주기 규칙을 정의하여 자동으로 로그 파일을 보관하거나 삭제할 수도 있습니다. 기본적으로 로그 파일은 Amazon S3 서버측 암호화(SSE)를 사용하여 암호화합니다.

새 로그 파일이 전송될 때 CloudTrail에서 Amazon SNS 알림을 게시하도록 할 수 있습니다. 자세한 내용은 CloudTrail용 Amazon SNS 알림 구성을 참조하십시오.

또한 여러 AWS 리전 및 여러 AWS 계정의 CodeDeploy 로그 파일을 단일 Amazon S3 버킷으로 집계할 수 있습니다. 자세한 내용은 여러 리전에서 CloudTrail 로그 파일 수신을 참조하세요.

CodeDeploy 로그 파일 항목 이해

CloudTrail 로그 파일은 하나 이상의 로그 항목이 있으며, 각 항목은 여러 개의 JSON 형식 이벤트로 구성됩니다. 로그 항목은 어떤 소스로부터의 요청 하나를 나타내며 요청된 작업, 모든 파라미터, 작업 날짜와 시간 등에 대한 정보가 들어 있습니다. 로그 항목의 순서가 정해져 있는 것은 아닙니다. 즉 순서가 지정된 퍼블릭 API 호출의 스택 추적이 아닙니다.

다음은 CodeDeploy가 배포 그룹을 생성하는 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예제입니다.

```
{
 "Records": [{
  "eventVersion": "1.02",
  "userIdentity": {
   "type": "AssumedRole",
   "principalId": "AKIAI44QH8DHBEXAMPLE:203.0.113.11",
   "arn": "arn:aws:sts::123456789012:assumed-role/example-role/203.0.113.11",
   "accountId": "123456789012",
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "sessionContext": {
    "attributes": {
     "mfaAuthenticated": "false",
     "creationDate": "2014-11-27T03:57:36Z"
```

```
    },
    "sessionIssuer": {
     "type": "Role",
     "principalId": "AKIAI44QH8DHBEXAMPLE",
     "arn": "arn:aws:iam::123456789012:role/example-role",
     "accountId": "123456789012",
     "userName": "example-role"
    }
   }
  },
  "eventTime": "2014-11-27T03:57:36Z",
  "eventSource": "codedeploy.amazonaws.com",
  "eventName": "CreateDeploymentGroup",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.11",
  "userAgent": "example-user-agent-string",
  "requestParameters": {
   "applicationName": "ExampleApplication",
   "serviceRoleArn": "arn:aws:iam::123456789012:role/example-instance-group-role",
   "deploymentGroupName": "ExampleDeploymentGroup",
   "ec2TagFilters": [{
               "value": "CodeDeployDemo",
    "type": "KEY_AND_VALUE",
    "key": "Name"
            }],
            "deploymentConfigName": "CodeDeployDefault.HalfAtATime"
  },
  "responseElements": {
   "deploymentGroupId": "7d64e680-e6f4-4c07-b10a-9e117EXAMPLE"
  },
  "requestID": "86168559-75e9-11e4-8cf8-75d18EXAMPLE",
  "eventID": "832b82d5-d474-44e8-a51d-093ccEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
 },
    ... additional entries ...
    ]
}
```

Amazon SNS 이벤트 알림으로 배포 모니터링

CodeDeploy 배포 그룹에 트리거를 추가하여 배포 또는 배포 그룹의 인스턴스와 관련된 이벤트에 대한 알림을 수신할 수 있습니다. 이러한 알림은 트리거 작업의 일부로 만든 Amazon SNS 주제를 구독하는 수신자에게 전송됩니다.

CodeDeploy 이벤트에 대한 알림은 SMS 메시지 또는 이메일 메시지로 수신할 수 있습니다. 지정된 이벤트가 발생할 때 생성되는 JSON 데이터는 Amazon SQS 대기열로 메시지 보내기 또는 AWS Lambda의 함수 호출 등 다른 방법으로 사용할 수 있습니다. 배포 및 인스턴스 트리거에 대해 제공되는 JSON 데이터의 구조를 보려면 CodeDeploy 트리거에 대한 JSON 데이터 형식 단원을 참조하세요.

다음과 같은 경우 트리거를 사용하여 알림을 수신하도록 선택할 수 있습니다.

- 문제를 해결할 수 있도록 배포에 실패하거나 배포가 중지된 경우를 알아야 하는 개발자인 경우
- Amazon EC2 플릿의 상태를 모니터링하기 위해 실패한 인스턴스 수를 알아야 하는 시스템 관리자인 경우
- 배포 및 인스턴스 이벤트의 개수를 한 눈에 파악하길 원하는 관리자인 경우. 이러한 정보는 여러 알림 유형을 데스크톱 이메일 클라이언트의 폴더로 라우팅하는 규칙을 필터링해 파악할 수 있습니다.

다음 모든 이벤트 유형에 대해 각 CodeDeploy 배포 그룹에 트리거를 최대 10개까지 만들 수 있습니다.

배포 이벤트	인스턴스 이벤트
• Success • 실패 • 시작됨 • 중지됨 • 롤백 • 준비[1] • 모든 배포 이벤트	• Success • 실패 • 시작됨 • 준비[1] • 모든 인스턴스 이벤트

[1]블루/그린 배포에만 적용됩니다. 대체 환경의 인스턴스에 최신 애플리케이션 수정이 설치되었으며 원본 환경에서의 트래픽을 이제 로드 밸런서 뒤로 다시 라우팅할 수 있음을 나타냅니다. 자세한 내용은 CodeDeploy에서 배포 작업 단원을 참조하세요.

주제

- CodeDeploy 서비스 역할에 Amazon SNS 권한 부여
- CodeDeploy 이벤트에 대한 트리거 만들기
- CodeDeploy 배포 그룹에서 트리거 편집
- CodeDeploy 배포 그룹에서 트리거 삭제
- CodeDeploy 트리거에 대한 JSON 데이터 형식

CodeDeploy 서비스 역할에 Amazon SNS 권한 부여

트리거가 알림을 생성하려면 CodeDeploy 작업에서 사용하는 서비스 역할에 Amazon SNS 리소스에 액세스할 수 있는 권한이 부여되어 있어야 합니다.

서비스 역할에 Amazon SNS 권한을 부여하려면

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. IAM 콘솔의 탐색 창에서 역할(Roles)을 선택합니다.
3. AWS CodeDeploy 작업에서 사용한 서비스 역할 이름을 선택합니다.
4. [Permissions] 탭의 [Inline Policies]에서, [Create Role Policy]를 선택합니다.

   -또는-

   [Create Role Policy] 버튼을 사용할 수 없으면 [Inline Policies] 영역을 확장한 후 [click here]를 선택합니다.
5. [Set Permissions] 페이지에서 [Custom Policy]와 [Select]를 차례로 선택합니다.
6. 정책 검토 페이지의 정책 이름 필드에 이 정책을 식별할 수 있는 이름을 입력합니다(예: `SNSPublish`).
7. 다음 내용을 [Policy Document] 필드에 붙여 넣습니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
```

```
            "Resource": "*"
        }
    ]
}
```

8. 정책 적용을 선택합니다.

CodeDeploy 이벤트에 대한 트리거 만들기

배포 또는 인스턴스 이벤트에 대한 Amazon Simple Notification Service(Amazon SNS) 주제를 게시하는 트리거를 AWS CodeDeploy 생성할 수 있습니다. 그런 다음, 해당 이벤트가 발생하면 연결된 주제의 모든 구독자가 주제에 지정된 엔드포인트(예: SMS 메시지 또는 이메일 메시지)를 통해 알림을 수신합니다. Amazon SNS 주제를 구독하기 위한 다양한 방법을 제공합니다.

트리거를 생성하기 전에 트리거가 가리키도록 Amazon SNS 주제를 설정해야 합니다. 자세한 내용은 주제 만들기를 참조하세요. 주제를 만들면 목적을 식별하는 이름을 `Topic-group-us-west-3-deploy-fail` 또는 `Topic-group-project-2-instance-stop` 등의 형식으로 지정하는 것이 좋습니다.

또한 트리거에 대한 알림을 보내려면 CodeDeploy 서비스 역할에 Amazon SNS 권한을 부여해야 합니다. 자세한 내용은 CodeDeploy 서비스 역할에 Amazon SNS 권한 부여 단원을 참조하세요.

주제를 만든 후에는 구독자를 추가할 수 있습니다. 주제 생성, 관리 및 구독에 대한 자세한 내용은 Amazon Simple Notification Service란 무엇입니까?을 참조하세요.

CodeDeploy 이벤트에 대한 알림을 전송하는 트리거 만들기(콘솔)

CodeDeploy 콘솔을 사용하여 CodeDeploy 이벤트에 대한 트리거를 만들 수 있습니다. 설정 프로세스를 마치면 권한 및 트리거 세부 정보가 둘 다 제대로 설정되었는지 확인하기 위해 테스트 알림 메시지가 전송됩니다.

CodeDeploy 이벤트에 대한 트리거를 만들려면

1. 에서 AWS CodeDeploy 콘솔을 AWS Management Console엽니다.
2. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

> **Note**
>
> CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

3. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
4. 애플리케이션 페이지에서 트리거를 추가할 배포 그룹과 연결된 애플리케이션 이름을 선택합니다.
5. 애플리케이션 세부 정보 페이지에서 트리거를 추가할 배포 그룹을 선택합니다.
6. 편집을 선택합니다.
7. 고급 옵션을 확장합니다.
8. 트리거 영역에서 트리거 생성을 선택합니다.
9. 배포 트리거 생성 창에서 다음을 수행합니다.

    a. 트리거 이름에 용도를 쉽게 식별할 수 있도록 트리거 이름을 입력합니다. `Trigger-group-us-west-3-deploy-fail` 또는 `Trigger-group-eu-central-instance-stop` 등의 형식이 좋습니다.

    b. 이벤트에서 Amazon SNS 주제가 알림을 전송하도록 트리거할 이벤트 유형을 선택합니다.

    c. Amazon SNS 주제에서 이 트리거에 대한 알림을 전송하기 위해 만든 주제 이름을 선택합니다.

    d. 트리거 생성을 선택합니다. CodeDeploy는 테스트 알림을 전송하여 CodeDeploy와 Amazon SNS 주제 간의 액세스 구성이 올바른지 확인합니다. 선택한 주제 엔드포인트 유형에 따라 주제를 구독하는 경우, SMS 메시지 또는 이메일 메시지로 확인 메시지가 수신됩니다.

10. 변경 사항 저장을 선택합니다.

CodeDeploy 이벤트에 대한 알림을 전송하는 트리거 만들기(CLI)

CLI를 사용하면 배포 그룹을 만들 때 트리거를 포함하거나 기본 배포 그룹에 트리거를 추가할 수 있습니다.

새 배포 그룹에 대한 알림을 전송하는 트리거를 만들려면

JSON 파일을 만들어 배포 그룹을 구성한 다음 `--cli-input-json` 옵션을 사용하여 create-deployment-group 명령을 실행합니다.

`--generate-cli-skeleton` 옵션을 통해 JSON 형식의 사본을 가져와 일반 텍스트 편집기에 필수 값을 입력하면 JSON 파일을 아주 쉽게 만들 수 있습니다.

1. 다음 명령을 실행한 후 일반 텍스트 편집기에 결과를 복사합니다.

```
aws deploy create-deployment-group --generate-cli-skeleton
```

2. 출력에 기존 CodeDeploy 애플리케이션의 이름을 추가합니다.

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentGroupName": "",
    "deploymentConfigName": "",
    "ec2TagFilters": [
        {
            "Key": "",
            "Value": "",
            "Type": ""
        }
    ],
    "onPremisesInstanceTagFilters": [
        {
            "Key": "",
            "Value": "",
            "Type": ""
        }
    ],
    "autoScalingGroups": [
        ""
    ],
    "serviceRoleArn": "",
    "triggerConfigurations": [
        {
            "triggerName": "",
            "triggerTargetArn": "",
            "triggerEvents": [
                ""
            ]
        }
    ]
}
```

3. 구성할 파라미터 값을 입력합니다.

   create-deployment-group 명령을 사용할 때 최소한 다음 파라미터 값을 입력해야 합니다.

- `applicationName`: 이미 계정에 생성된 애플리케이션 이름
- `deploymentGroupName`: 생성 중인 배포 그룹의 이름
- `serviceRoleArn`: 계정에서 CodeDeploy에 대해 설정된 기존 서비스 역할의 ARN 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하세요.

`triggerConfigurations` 섹션에서 다음 파라미터 값을 입력합니다.

- `triggerName`: 쉽게 식별할 수 있도록 트리거 이름을 지정합니다. `Trigger-group-us-west-3-deploy-fail` 또는 `Trigger-group-eu-central-instance-stop` 등의 형식이 좋습니다.
- `triggerTargetArn`: 트리거에 연결하기 위해 생성한 Amazon SNS주제의 ARN으로, 형식은 `arn:aws:sns:us-east-2:444455556666:NewTestTopic`입니다.
- `triggerEvents`: 알림을 트리거하려는 이벤트 유형 하나 이상의 이벤트 유형을 지정할 수 있으며, 이벤트 유형 이름이 여러 개인 경우 쉼표로 구분합니다(예: `"triggerEvents": ["DeploymentSuccess","DeploymentFailure","InstanceFailure"]`). 하나 이상의 이벤트 유형을 추가할 때, 해당되는 모든 유형 알림이 각기 다른 주제가 아닌 지정된 주제로 전송됩니다. 다음 이벤트 유형 중에서 선택할 수 있습니다.
  - DeploymentStart
  - DeploymentSuccess
  - DeploymentFailure
  - DeploymentStop
  - DeploymentRollback
  - DeploymentReady(블루/그린 배포의 대체 인스턴스에만 적용됨)
  - InstanceStart
  - InstanceSuccess
  - InstanceFailure
  - InstanceReady(블루/그린 배포의 대체 인스턴스에만 적용됨)

다음 구성 예제는 `TestApp-us-east-2`라는 애플리케이션에 대해 `dep-group-ghi-789-2`라는 배포 그룹을 생성하고, 배포가 시작, 성공 또는 실패할 때마다 알림 메시지를 전송하는 트리거를 생성합니다.

```
{
```

```
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "deploymentGroupName": "dep-group-ghi-789-2",
    "ec2TagFilters": [
        {
            "Key": "Name",
            "Value": "Project-ABC",
            "Type": "KEY_AND_VALUE"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "triggerConfigurations": [
        {
            "triggerName": "Trigger-group-us-east-2",
            "triggerTargetArn": "arn:aws:sns:us-east-2:444455556666:us-east-
deployments",
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure"
            ]
        }
    ]
}
```

4. 업데이트를 JSON 파일로 저장한 후 create-deployment-group 명령을 실행할 때 `--cli-input-json` 옵션을 사용하여 해당 파일을 호출합니다.

> **Important**
>
> 파일 이름 앞에 `file://`를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws deploy create-deployment-group --cli-input-json file://filename.json
```

만들기 프로세스를 마치면 권한 및 트리거 세부 정보가 둘 다 제대로 설정되었음을 나타내는 테스트 알림 메시지가 수신됩니다.

기존 배포 그룹에 대한 알림을 전송하는 트리거를 만들려면

AWS CLI 를 사용하여 기존 배포 그룹에 CodeDeploy 이벤트에 대한 트리거를 추가하려면 JSON 파일을 생성하여 배포 그룹을 업데이트한 다음 `--cli-input-json` 옵션을 사용하여 update-deployment-group 명령을 실행합니다.

get-deployment-group 명령을 실행하여 배포 그룹 구성의 사본을 JSON 형식으로 가져온 다음 일반 텍스트 편집기에서 파라미터 값을 업데이트하면 JSON 파일을 아주 쉽게 만들 수 있습니다.

1. 다음 명령을 실행한 후 일반 텍스트 편집기에 결과를 복사합니다.

```
aws deploy get-deployment-group --application-name application --deployment-group-
name deployment-group
```

2. 출력에서 다음을 삭제합니다.

   - 출력 시작 부분에서 `{ "deploymentGroupInfo":`를 삭제합니다.
   - 출력 끝부분에서 `}`를 삭제합니다.
   - `deploymentGroupId`을 포함하는 행을 삭제합니다.
   - `deploymentGroupName`을 포함하는 행을 삭제합니다.

   텍스트 파일의 내용이 다음과 같아야 합니다.

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "Project-ABC",
            "Key": "Name"
        }
    ],
    "triggerConfigurations": [],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

3. `triggerConfigurations` 섹션에, `triggerEvents`, `triggerTargetArn` 및 `triggerName` 파라미터에 대한 데이터를 추가합니다. 트리거 구성 파라미터에 대한 자세한 내용은 TriggerConfig를 참조하세요.

   텍스트 파일의 내용이 다음과 같아야 합니다. 이 코드는 배포가 시작, 성공 또는 실패할 때마다 전송할 알림 메시지를 표시합니다.

```
{
    "applicationName": "TestApp-us-east-2",
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "Project-ABC",
            "Key": "Name"
        }
    ],
    "triggerConfigurations": [
        {
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure"
            ],
            "triggerTargetArn": "arn:aws:sns:us-east-2:444455556666:us-east-
deployments",
            "triggerName": "Trigger-group-us-east-2"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

4. 업데이트를 JSON 파일로 저장한 후 `--cli-input-json` 옵션을 사용하여 update-deployment-group 명령을 실행합니다. `--current-deployment-group-name` 옵션을 포함하고 *`filename`*을 JSON 파일 이름으로 바꿔야 합니다.

> **Important**
>
> 파일 이름 앞에 `file://`를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws deploy update-deployment-group --current-deployment-group-name deployment-group-name --cli-input-json file://filename.json
```

만들기 프로세스를 마치면 권한 및 트리거 세부 정보가 둘 다 제대로 설정되었음을 나타내는 테스트 알림 메시지가 수신됩니다.

CodeDeploy 배포 그룹에서 트리거 편집

알림 요구 사항을 변경한 경우 트리거를 새로 만드는 대신 트리거를 수정할 수 있습니다.

CodeDeploy 수정(CLI)

배포 그룹을 업데이트할 때 AWS CLI 를 사용하여 CodeDeploy 이벤트의 트리거 세부 정보를 변경하려면 JSON 파일을 생성하여 배포 그룹의 속성에 대한 변경 사항을 정의한 다음 `--cli-input-json` 옵션을 사용하여 update-deployment-group 명령을 실행합니다.

get-deployment-group 명령을 실행하여 현재 배포 그룹 세부 정보를 JSON 형식으로 가져온 다음 일반 텍스트 편집기에서 필수 값을 편집하면 JSON 파일을 아주 쉽게 만들 수 있습니다.

1. 다음 명령을 실행하고 *application* 및 *deployment-group*을 해당 애플리케이션 및 배포 그룹의 이름으로 바꿉니다.

   ```
   aws deploy get-deployment-group --application-name application --deployment-group-name deployment-group
   ```

2. 일반 텍스트 편집기에 명령 결과를 복사한 후 다음을 삭제합니다.

   - 출력 시작 부분에서 `{ "deploymentGroupInfo":`를 삭제합니다.
   - 출력 끝부분에서 `}`를 삭제합니다.
   - `deploymentGroupId`을 포함하는 행을 삭제합니다.
   - `deploymentGroupName`을 포함하는 행을 삭제합니다.

   텍스트 파일의 내용이 다음과 같아야 합니다.

   ```
   {
       "applicationName": "TestApp-us-east-2",
   ```

```
    "deploymentConfigName": "CodeDeployDefault.OneAtATime",
    "autoScalingGroups": [],
    "ec2TagFilters": [
        {
            "Type": "KEY_AND_VALUE",
            "Value": "East-1-Instances",
            "Key": "Name"
        }
    ],
    "triggerConfigurations": [
        {
            "triggerEvents": [
                "DeploymentStart",
                "DeploymentSuccess",
                "DeploymentFailure",
                "DeploymentStop"
            ],
            "triggerTargetArn": "arn:aws:sns:us-east-2:111222333444:Trigger-group-
us-east-2",
            "triggerName": "Trigger-group-us-east-2"
        }
    ],
    "serviceRoleArn": "arn:aws:iam::444455556666:role/AnyCompany-service-role",
    "onPremisesInstanceTagFilters": []
}
```

3. 필요한 경우 파라미터를 변경합니다. 트리거 구성 파라미터에 대한 자세한 내용은 TriggerConfig를 참조하세요.
4. 업데이트를 JSON 파일로 저장한 후 `--cli-input-json` 옵션을 사용하여 update-deployment-group 명령을 실행합니다. `--current-deployment-group-name` 옵션을 포함하고 *`filename`*을 JSON 파일 이름으로 바꿔야 합니다.

> **Important**
>
> 파일 이름 앞에 `file://`를 포함해야 합니다. 이 명령에 필수적입니다.

```
aws deploy update-deployment-group --current-deployment-group-name deployment-
group-name --cli-input-json file://filename.json
```

만들기 프로세스를 마치면 권한 및 트리거 세부 정보가 둘 다 제대로 설정되었음을 나타내는 테스트 알림 메시지가 수신됩니다.

CodeDeploy 배포 그룹에서 트리거 삭제

배포 그룹당 트리거 개수 제한이 10개이므로 더 이상 사용하지 않는 트리거를 삭제하려고 할 수 있습니다. 트리거 삭제는 실행 취소할 수 없지만 트리거를 다시 생성할 수 있습니다.

배포 그룹에서 트리거 삭제(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/codedeploy CodeDeploy 콘솔을 엽니다.

   > **Note**
   >
   > CodeDeploy 시작하기에서 설정한 사용자와 동일한 사용자로 로그인합니다.

2. 탐색 창에서 배포를 확장하고 애플리케이션을 선택합니다.
3. 애플리케이션 페이지에서 트리거를 삭제할 배포 그룹과 연결된 애플리케이션 이름을 선택합니다.
4. 애플리케이션 세부 정보 페이지에서 트리거를 삭제할 배포 그룹을 선택합니다.
5. 편집을 선택합니다.
6. 고급 옵션을 확장합니다.
7. 트리거 영역에서 삭제할 트리거를 선택한 다음 트리거 삭제를 선택합니다.
8. 변경 사항 저장을 선택합니다.

배포 그룹에서 트리거 삭제(CLI)

CLI를 사용하여 트리거를 삭제하려면 빈 트리거 구성 파라미터를 사용하여 update-deployment-group 명령을 호출해 다음을 지정합니다.

- 배포 그룹과 연결된 애플리케이션 이름. 애플리케이션 이름 목록을 보려면 list-applications 명령을 호출합니다.
- 애플리케이션과 연결된 배포 그룹 이름. 배포 그룹 이름 목록을 보려면 list-deployment-groups 명령을 호출합니다.

예:

```
aws deploy update-deployment-group --application-name application-name --current-
deployment-group-name deployment-group-name --trigger-configurations
```

CodeDeploy 트리거에 대한 JSON 데이터 형식

배포 또는 인스턴스에 대한 트리거가 사용자 지정 알림 워크플로우(예: Amazon SQS 대기열로 메시지 전송 또는 AWS Lambda에서 함수 호출)에서 활성화될 때 생성되는 JSON 출력을 사용할 수 있습니다.

> **Note**
>
> 이 설명서에서는 JSON을 사용하여 알림을 구성하는 방법을 다루지 않습니다. Amazon SNS를 사용하여 Amazon SQS 대기열에 메시지를 보내는 방법에 대한 자세한 내용은 Amazon SQS 대기열로 Amazon SNS 메시지 전송을 참조하세요. Amazon SNS 사용하여 Lambda 함수를 호출하는 방법에 대한 자세한 내용은 Amazon SNS 알림을 사용하여 Lambda 함수 호출을 참조하세요.

다음 예에서는 CodeDeploy 트리거에서 사용할 수 있는 JSON 출력의 구조를 보여 줍니다.

인스턴스 기반 트리거에 대한 샘플 JSON 출력

```
{
    "region": "us-east-2",
    "accountId": "111222333444",
    "eventTriggerName": "trigger-group-us-east-instance-succeeded",
    "deploymentId": "d-75I7MBT7C",
    "instanceId": "arn:aws:ec2:us-east-2:444455556666:instance/i-496589f7",
    "lastUpdatedAt": "1446744207.564",
    "instanceStatus": "Succeeded",
    "lifecycleEvents": [
        {
            "LifecycleEvent": "ApplicationStop",
            "LifecycleEventStatus": "Succeeded",
            "StartTime": "1446744188.595",
            "EndTime": "1446744188.711"
        },
        {
            "LifecycleEvent": "BeforeInstall",
            "LifecycleEventStatus": "Succeeded",
            "StartTime": "1446744189.827",
            "EndTime": "1446744190.402"
```

```
        }
//More lifecycle events might be listed here
    ]
}
```

배포 기반 트리거에 대한 샘플 JSON 출력

```
{
    "region": "us-west-1",
    "accountId": "111222333444",
    "eventTriggerName": "Trigger-group-us-west-3-deploy-failed",
    "applicationName": "ProductionApp-us-west-3",
    "deploymentId": "d-75I7MBT7C",
    "deploymentGroupName": "dep-group-def-456",
    "createTime": "1446744188.595",
    "completeTime": "1446744190.402",
    "deploymentOverview": {
        "Failed": "10",
        "InProgress": "0",
        "Pending": "0",
        "Skipped": "0",
        "Succeeded": "0"
    },
    "status": "Failed",
    "errorInformation": {
        "ErrorCode": "IAM_ROLE_MISSING",
        "ErrorMessage": "IAM Role is missing for deployment group: dep-group-def-456"
    }
}
```

보안 in AWS CodeDeploy

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이를 클라우드의 보안과 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 정기적으로 AWS 규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. AWS CodeDeploy에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 AWS 규정 준수 프로그램 제공 범위 내 서비스를 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 CodeDeploy 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 CodeDeploy를 구성하는 방법을 보여줍니다. 또한 CodeDeploy 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- in AWS CodeDeploy의 데이터 보호
- 에 대한 자격 증명 및 액세스 관리 AWS CodeDeploy
- CodeDeploy에서 로깅 및 모니터링
- AWS CodeDeploy에 대한 규정 준수 검증
- Resilience in AWS CodeDeploy
- 인프라 보안 in AWS CodeDeploy

in AWS CodeDeploy의 데이터 보호

AWS 공동 책임 모델 in AWS CodeDeploy의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성

과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 일반 데이터 보호 규정 (GDPR) 센터를 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 CodeDeploy 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 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.

인터네트워크 트래픽 개인 정보

CodeDeploy는 EC2 인스턴스, Lambda 함수, Amazon ECS 및 온프레미스 서버를 지원하는 완전 관리형 배포 서비스입니다. EC2 인스턴스 및 온프레미스 서버의 경우 호스트 기반 에이전트는 TLS를 사용하여 CodeDeploy와 통신합니다.

현재 에이전트에서 서비스로의 통신에는 에이전트가 퍼블릭 CodeDeploy 및 Amazon S3 서비스 엔드포인트와 통신할 수 있도록 아웃바운드 인터넷 연결이 필요합니다. Virtual Private Cloud(VPC)에서는 인터넷 게이트웨이, 회사 네트워크에 대한 사이트 간 VPN 연결 또는 직접 연결을 사용하여 이 작업을 수행할 수 있습니다.

CodeDeploy 에이전트는 HTTP 프록시를 지원합니다.

기반 Amazon VPC 엔드포인트 AWS PrivateLink는 특정 리전의 CodeDeploy에서 사용할 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud와 함께 CodeDeploy 사용 단원을 참조하세요.

> **Note**
>
> CodeDeploy 에이전트는 Amazon EC2/온프레미스 컴퓨팅 플랫폼에 배포하는 경우에만 필요합니다. Amazon ECS 또는 AWS Lambda 컴퓨팅 플랫폼을 사용하는 배포에는 에이전트가 필요하지 않습니다.

저장 데이터 암호화

고객 코드는 CodeDeploy에 저장되지 않습니다. 배포 서비스로서 CodeDeploy는 EC2 인스턴스 또는 온프레미스 서버에서 실행되는 CodeDeploy 에이전트에 명령을 디스패치합니다. 그런 다음 CodeDeploy 에이전트는 TLS를 사용하여 명령을 실행합니다. 배포, 배포 구성, 배포 그룹, 애플리케이션 및 애플리케이션 개정에 대한 서비스 모델 데이터는 Amazon DynamoDB에 저장되며 CodeDeploy에서 AWS 소유 키소유하고 관리하는를 사용하여 저장 시 암호화됩니다. 자세한 정보는 AWS 소유 키를 참조하세요.

전송 중 암호화

CodeDeploy 에이전트는 포트 443을 통해 CodeDeploy와 모든 통신을 시작합니다. 이 에이전트는 CodeDeploy를 폴링하고 명령을 수신 대기합니다. CodeDeploy 에이전트는 오픈 소스입니다. 모든 서비스 대 서비스 및 클라이언트 대 서비스 통신은 TLS를 사용하여 전송 중에 암호화됩니다. 이렇게 하면 CodeDeploy와 AmazonS3 등의 다른 서비스 간에 전송되는 고객 데이터를 보호할 수 있습니다.

암호화 키 관리

관리할 암호화 키가 없습니다. CodeDeploy 서비스 모델 데이터는 CodeDeploy에서 AWS 소유 키소유하고 관리하는를 사용하여 암호화됩니다. 자세한 정보는 AWS 소유 키를 참조하세요.

에 대한 자격 증명 및 액세스 관리 AWS CodeDeploy

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 어떤 사용자가 CodeDeploy 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- AWS CodeDeploy 에서 IAM을 사용하는 방법
- AWS CodeDeploy에 대한 관리형(미리 정의된) 정책
- AWS 관리형 정책에 대한 CodeDeploy 업데이트
- AWS CodeDeploy 자격 증명 기반 정책 예제
- AWS CodeDeploy 자격 증명 및 액세스 문제 해결
- CodeDeploy 권한 참조
- 교차 서비스 혼동된 대리인 방지

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조AWS CodeDeploy 자격 증명 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(AWS CodeDeploy 에서 IAM을 사용하는 방법 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(AWS CodeDeploy 자격 증명 기반 정책 예제 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 연동을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS CodeDeploy 에서 IAM을 사용하는 방법

IAM을 사용하여 CodeDeploy에 대한 액세스를 관리하려면 먼저 어떤 IAM 기능을 CodeDeploy와 함께 사용할 수 있는지를 이해해야 합니다. 자세한 내용은 IAM 사용 설명서의 IAM으로 작업하는AWS 서비스를 참조하세요.

주제

- CodeDeploy 자격 증명 기반 정책
- CodeDeploy 리소스 기반 정책
- CodeDeploy 태그 기반 권한 부여
- IAM CodeDeploy 역할

CodeDeploy 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스 및 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. CodeDeploy는 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

CodeDeploy의 정책 작업은 작업 앞에 `codedeploy:` 접두사를 사용합니다. 예를 들어, `codedeploy:GetApplication` 권한은 사용자에게 `GetApplication` 작업을 수행할 수 있는 권한을 부여합니다. 정책 문에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. CodeDeploy는 이 서비스로 수행할 수 있는 작업을 설명하는 고유한 작업 세트를 정의합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "codedeploy:action1",
      "codedeploy:action2"
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `Describe`라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "ec2:Describe*"
```

CodeDeploy 작업 목록에 대해서는 IAM 사용 설명서의 AWS CodeDeploy에서 정의한 작업을 참조하세요.

모든 CodeDeploy API 작업과 해당 작업이 적용되는 리소스를 나열하는 표는 CodeDeploy 권한 참조 섹션을 참조하세요.

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. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

예를 들어 명령문에서 다음과 같이 ARN을 사용하여 배포 그룹(*myDeploymentGroup*)을 지정할 수 있습니다.

```
"Resource": "arn:aws:codedeploy:us-
west-2:123456789012:deploymentgroup:myApplication/myDeploymentGroup"
```

다음과 같이 와일드카드 문자(*)를 사용하여 계정에 속한 모든 배포 그룹을 지정할 수도 있습니다.

```
"Resource": "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*"
```

모든 리소스를 지정해야 하거나, API 작업이 ARN을 지원하지 않는 경우 다음과 같이 Resource 요소에 와일드카드 문자(*)를 사용합니다.

```
"Resource": "*"
```

일부 CodeDeploy API 작업에서는 여러 리소스 사용을 허용합니다(예: BatchGetDeploymentGroups). 명령문 하나에 여러 리소스를 지정하려면 다음과 같이 각 ARN을 쉼표로 구분합니다.

```
"Resource": ["arn1", "arn2"]
```

CodeDeploy는 CodeDeploy 리소스를 사용하기 위한 일련의 작업을 제공합니다. 사용 가능한 작업 목록은 CodeDeploy 권한 참조 단원을 참조하세요.

CodeDeploy 리소스 유형 및 해당 ARN의 목록을 보려면 IAM 사용 설명서의 AWS CodeDeploy에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업에 대한 자세한 내용은 AWS CodeDeploy에서 정의한 작업을 참조하세요.

CodeDeploy 리소스 및 작업

CodeDeploy에서 기본 리소스는 배포 그룹입니다. 정책에서 Amazon 리소스 이름(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다. CodeDeploy는 애플리케이션, 배포 구성 및 인스턴스를 비롯하여 배포 그룹에 사용할 수 있는 기타 리소스를 지원합니다. 이러한 리소스를 가리켜 하위 리소스라 합니다. 이러한 리소스와 하위 리소스에는 고유한 ARN이 연결되어 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 Amazon 리소스 이름(ARN)을 참조하세요.

리소스 유형	ARN 형식
배포 그룹	`arn:aws:codedeploy:` *`region`*`:`*`account-id`* `:deploymentgroup:` *`application-name`* `/`*`deployment-group-name`*
애플리케이션	`arn:aws:codedeploy:` *`region`*`:`*`account-id`* `:application:` *`application-name`*
배포 구성	`arn:aws:codedeploy:` *`region`*`:`*`account-id`* `:deploymentconfig:` *`deployment-configuration-name`*
Instance	`arn:aws:codedeploy:` *`region`*`:`*`account-id`* `:instance/` *`instance-ID`*
모든 CodeDeploy 리소스	`arn:aws:codedeploy:*`
지정한 리전에서 지정한 계정이 소유한 모든 CodeDeploy 리소스	`arn:aws:codedeploy:` *`region`*`:`*`account-id`* `:*`

> **Note**
>
> 의 대부분의 서비스는 콜론(:) 또는 슬래시(/)를 ARN에서 동일한 문자로 AWS 취급합니다. ARNs 그러나 CodeDeploy는 리소스 패턴 및 규칙에서 정확한 일치를 사용합니다. 따라서 이벤트 패턴을 만들 때 리소스에서 ARN 구문이 일치하도록 정확한 ARN 문자를 사용해야 합니다.

조건 키

CodeDeploy는 서비스별 조건 키를 제공하지 않지만, 일부 전역 조건 키 사용을 지원합니다. 자세한 내용은 IAM 사용 설명서의 AWS 글로벌 조건 컨텍스트 키를 참조하세요.

예제

CodeDeploy 자격 증명 기반 정책 예제를 보려면 AWS CodeDeploy 자격 증명 기반 정책 예제 섹션을 참조하세요.

CodeDeploy 리소스 기반 정책

CodeDeploy는 리소스 기반 정책을 지원하지 않습니다. 자세한 리소스 기반 정책 페이지의 예를 보려면 리소스 기반 정책 사용을 참조하세요 AWS Lambda.

CodeDeploy 태그 기반 권한 부여

CodeDeploy는 리소스 태깅 또는 태그 기반 액세스 제어를 지원하지 않습니다.

IAM CodeDeploy 역할

IAM 역할은 특정 권한이 있는 AWS 계정의 엔터티입니다.

CodeDeploy에서 임시 자격 증명 사용

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken과 같은 AWS STS API 작업을 호출하여 임시 보안 자격 증명을 얻습니다.

CodeDeploy에서 임시 자격 증명 사용을 지원합니다.

서비스 연결 역할

CodeDeploy는 서비스 연결 역할을 지원하지 않습니다.

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 서비스 역할을 수임할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역

할은 AWS 계정에 나타나며 계정이 소유합니다. 즉, 사용자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

CodeDeploy는 서비스 역할을 지원합니다.

CodeDeploy에서 IAM 역할 선택

CodeDeploy에서 배포 그룹 리소스를 생성할 경우 CodeDeploy가 사용자 대신 Amazon EC2에 액세스할 수 있도록 하는 역할을 선택해야 합니다. 이전에 서비스 역할 또는 서비스 연결 역할을 생성한 경우 CodeDeploy는 선택할 수 있는 역할 목록을 제공합니다. EC2 인스턴스 시작 및 중지에 대한 액세스를 허용하는 역할을 선택하는 것이 중요합니다.

AWS CodeDeploy에 대한 관리형(미리 정의된) 정책

AWS 는에서 생성하고 관리하는 독립 실행형 IAM 정책을 제공하여 많은 일반적인 사용 사례를 처리합니다 AWS. 이러한 AWS관리형 정책은 일반적인 사용 사례에 대한 권한을 부여하므로 필요한 권한을 조사할 필요가 없습니다. 자세한 내용은 IAM 사용자 가이드의 AWS 관리형 정책을 참조하세요.

주제

- CodeDeploy에 대한 AWS 관리형 정책 목록
- CodeDeploy 관리형 정책 및 알림

CodeDeploy에 대한 AWS 관리형 정책 목록

계정의 사용자에게 연결할 수 있는 다음 AWS 관리형 정책은 CodeDeploy에 고유합니다.

- `AWSCodeDeployFullAccess`: CodeDeploy에 전체 액세스 권한 부여

> **Note**
>
> AWSCodeDeployFullAccess는 애플리케이션 배포에 필요한 기타 서비스(예: Amazon EC2 및Amazon S3)에서 작업을 수행할 수 있는 권한이 아니라 CodeDeploy 관련 작업을 수행할 수 있는 권한만 제공합니다.

- `AWSCodeDeployDeployerAccess`: 개정 버전을 등록 및 배포할 수 있는 권한을 부여합니다.

- `AWSCodeDeployReadOnlyAccess`: CodeDeploy에 대한 읽기 전용 액세스 권한을 부여합니다.

- `AWSCodeDeployRole`: CodeDeploy가 다음 작업을 수행할 수 있도록 합니다.
  - Amazon EC2 Auto Scaling 그룹 이름으로 인스턴스의 태그를 읽거나 Amazon EC2 인스턴스를 식별합니다.
  - Amazon EC2 Auto Scaling 그룹, 수명 주기 후크, 조정 정책 및 웜 풀 기능을 읽고, 생성하고, 업데이트하고, 삭제합니다.
  - Amazon SNS 주제로 정보를 게시합니다.
  - Amazon CloudWatch 경보에 관한 정보를 검색합니다.
  - Elastic Load Balancing 서비스에서 리소스 읽고 업데이트합니다.

  정책에는 다음 코드가 포함됩니다.

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction",
        "autoscaling:DeleteLifecycleHook",
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeLifecycleHooks",
        "autoscaling:PutLifecycleHook",
        "autoscaling:RecordLifecycleActionHeartbeat",
        "autoscaling:CreateAutoScalingGroup",
        "autoscaling:CreateOrUpdateTags",
        "autoscaling:UpdateAutoScalingGroup",
        "autoscaling:EnableMetricsCollection",
        "autoscaling:DescribePolicies",
        "autoscaling:DescribeScheduledActions",
        "autoscaling:DescribeNotificationConfigurations",
        "autoscaling:SuspendProcesses",
        "autoscaling:ResumeProcesses",
        "autoscaling:AttachLoadBalancers",
        "autoscaling:AttachLoadBalancerTargetGroups",
        "autoscaling:PutScalingPolicy",
        "autoscaling:PutScheduledUpdateGroupAction",
        "autoscaling:PutNotificationConfiguration",
```

```
        "autoscaling:DescribeScalingActivities",
        "autoscaling:DeleteAutoScalingGroup",
        "autoscaling:PutWarmPool",
        "ec2:DescribeInstances",
        "ec2:DescribeInstanceStatus",
        "ec2:TerminateInstances",
        "tag:GetResources",
        "sns:Publish",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:PutMetricAlarm",
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeLoadBalancerAttributes",
        "elasticloadbalancing:DescribeInstanceHealth",
        "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
        "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetGroupAttributes",
        "elasticloadbalancing:DescribeTargetHealth",
        "elasticloadbalancing:RegisterTargets",
        "elasticloadbalancing:DeregisterTargets"
      ],
      "Resource": "*"
    }
  ]
}
```

- `AWSCodeDeployRoleForLambda`: CodeDeploy에 배포에 필요한 AWS Lambda 및 기타 리소스에 액세스할 수 있는 권한을 부여합니다.

- `AWSCodeDeployRoleForECS`: CodeDeploy에 Amazon ECS 및 배포에 필요한 기타 리소스에 액세스할 수 있는 권한을 부여합니다.

- `AWSCodeDeployRoleForECSLimited`: CodeDeploy에 Amazon ECS 및 배포에 필요한 기타 리소스에 액세스할 수 있는 권한을 부여하지만, 다음과 같은 예외가 있습니다.
  - AppSpec 파일의 `hooks` 섹션에서 `CodeDeployHook_`로 시작하는 이름의 Lambda 함수만 사용할 수 있습니다. 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션 섹션을 참조하세요.

  - S3 버킷 액세스는 값이 `true`인 등록 태그 `UseWithCodeDeploy`가 있는 S3 버킷으로 제한됩니다. 자세한 내용은 객체 태그 지정을 참조하세요.
- `AmazonEC2RoleforAWSCodeDeployLimited`: CodeDeploy Amazon S3 버킷에서 객체를 가져오고 나열할 수 있는 CodeDeploy 권한을 부여합니다. 정책에는 다음 코드가 포함됩니다.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*/CodeDeploy/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/UseWithCodeDeploy": "true"
                }
            }
        }
    ]
}
```

배포 프로세스의 일부 측면에 대한 권한은 CodeDeploy을 대신하는 다른 두 역할 유형에 부여됩니다.

- IAM 인스턴스 프로파일이란 Amazon EC2 인스턴스에 연결하는 IAM 역할입니다. 이 프로파일에는 애플리케이션이 저장되는 Amazon S3 버킷 또는 GitHub 리포지토리에 액세스할 때 필요한 권한이 포함됩니다. 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 섹션을 참조하세요.

- 서비스 역할은 AWS 리소스에 액세스할 수 있도록 AWS 서비스에 권한을 부여하는 IAM 역할입니다. 서비스 역할에 연결하는 정책에 따라 서비스가 액세스할 수 있는 AWS 리소스와 해당 리소스로 수행할 수 있는 작업이 결정됩니다. CodeDeploy의 경우 서비스 역할은 다음 작업에 사용됩니다.
  - 인스턴스에 적용된 태그 또는 인스턴스와 연결된 Amazon EC2 Auto Scaling 그룹 이름을 읽습니다. 이를 통해 CodeDeploy는 애플리케이션을 배포할 수 있는 인스턴스를 식별할 수 있습니다.
  - Amazon EC2 Auto Scaling 그룹 및 Elastic Load Balancing 로드 밸런서의 인스턴스에 대한 작업을 수행합니다.
  - 지정된 배포 또는 인스턴스 이벤트가 발생할 때 알림을 전송할 수 있도록 Amazon SNS 주제에 정보를 게시합니다.
  - CloudWatch 경보에 대한 정보를 검색하여 배포에 대한 경보 모니터링을 설정합니다.

  자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 섹션을 참조하세요.

사용자 지정 IAM 정책을 생성하여 CodeDeploy 작업 및 리소스에 대한 권한을 부여할 수도 있습니다. IAM 역할에 이 사용자 지정 정책을 연결한 다음 해당 권한이 필요한 사용자 또는 그룹에 해당 역할을 할당합니다.

CodeDeploy 관리형 정책 및 알림

CodeDeploy에서는 사용자가 배포에 대한 중요한 변경 사항을 알 수 있도록 알림을 지원합니다. CodeDeploy에 대한 관리형 정책에는 알림 기능에 대한 정책 설명이 포함되어 있습니다. 자세한 내용은 알림이란 무엇입니까?를 참조하세요.

전체 액세스 관리형 정책의 알림에 대한 권한

`AWSCodeDeployFullAccess` 관리형 정책에는 알림에 대한 전체 액세스를 허용하는 다음 설명이 포함되어 있습니다. 이러한 관리형 정책이 적용된 사용자는 알림에 대한 Amazon SNS 주제를 생성 및 관리하고, 주제에 대해 사용자를 구독 및 구독 취소 처리하고, 알림 규칙의 대상으로 선택할 주제를 나열할 수도 있습니다.

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
```

```
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "ArnLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*:*:application:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
      "Sid" : "CodeStarNotificationsChatbotAccess",
      "Effect" : "Allow",
      "Action" : [
        "chatbot:DescribeSlackChannelConfigurations"
      ],
      "Resource" : "*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    }
```

읽기 전용 관리형 정책의 알림에 대한 권한

AWSCodeDeployReadOnlyAccess 관리형 정책에는 알림에 대한 읽기 전용 액세스를 허용하는 다음 설명이 포함되어 있습니다. 이 관리형 정책이 적용된 사용자는 리소스에 대한 알림을 볼 수 있지만 리소스를 생성, 관리 또는 구독할 수는 없습니다.

```
  {
       "Sid": "CodeStarNotificationsPowerUserAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:DescribeNotificationRule"
       ],
       "Resource": "*",
       "Condition" : {
           "ArnLike" : {"codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*:*:application:*"}
       }
   },
   {
       "Sid": "CodeStarNotificationsListAccess",
       "Effect": "Allow",
       "Action": [
           "codestar-notifications:ListNotificationRules"
       ],
       "Resource": "*"
   }
```

다른 관리형 정책의 알림에 대한 권한

AWSCodeDeployDeployerAccess 관리형 정책에는 사용자가 리소스에 대한 알림을 생성, 업데이트, 구독 및 볼 수 있도록 허용하는 다음 문이 포함되지만, 알림을 삭제할 수는 없습니다. 이 관리형 정책이 적용된 사용자는 알림을 위한 Amazon SNS 주제를 생성하고 관리할 수도 있습니다.

이 정책에는 다음을 할 수 있는 권한이 포함되어 있습니다.

- codestar-notifications:CreateNotificationRule - 보안 주체가 알림을 생성할 수 있도록 허용합니다.
- codestar-notifications:DescribeNotificationRule – 보안 주체가 알림에 대한 정보를 검색할 수 있도록 허용합니다.
- codestar-notifications:UpdateNotificationRule - 보안 주체가 알림을 업데이트할 수 있도록 허용합니다.

- codestar-notifications:Subscribe - 보안 주체가 알림 업데이트를 구독하도록 허용합니다.
- codestar-notifications:Unsubscribe - 보안 주체가 알림 업데이트 구독을 취소할 수 있습니다.
- codestar-notifications:ListNotificationRules - 보안 주체가 알림 규칙 목록을 검색할 수 있도록 허용합니다.
- codestar-notifications:ListTargets - 보안 주체가 대상 목록을 검색할 수 있도록 허용합니다.
- codestar-notifications:ListTagsforResource - 보안 주체가 태그 목록을 검색할 수 있도록 허용합니다.
- codestar-notifications:ListEventTypes - 보안 주체가 이벤트 유형 목록을 검색할 수 있도록 허용합니다.
- chatbot:DescribeSlackChannelConfiguration – 보안 주체가 Slack 채널 구성에 대한 정보를 검색할 수 있도록 합니다.
- sns:ListTopics - 보안 주체가 알림에 대한 Amazon SNS 주제 목록을 검색할 수 있도록 허용합니다.

```
  {
     "Sid" : "CodeStarNotificationsReadWriteAccess",
     "Effect" : "Allow",
     "Action" : [
       "codestar-notifications:CreateNotificationRule",
       "codestar-notifications:DescribeNotificationRule",
       "codestar-notifications:UpdateNotificationRule",
       "codestar-notifications:Subscribe",
       "codestar-notifications:Unsubscribe"
     ],
     "Resource" : "*",
     "Condition" : {
       "ArnLike" : {
         "codestar-notifications:NotificationsForResource" :
"arn:aws:codedeploy:*:*:application:*"
       }
     }
   },
   {
     "Sid" : "CodeStarNotificationsListAccess",
     "Effect" : "Allow",
     "Action" : [
```

```
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListTargets",
        "codestar-notifications:ListTagsforResource",
        "codestar-notifications:ListEventTypes"
      ],
      "Resource" : "*"
    },
    {
      "Sid" : "CodeStarNotificationsChatbotAccess",
      "Effect" : "Allow",
      "Action" : [
        "chatbot:DescribeSlackChannelConfigurations"
      ],
      "Resource" : "*"
    },
    {
      "Sid" : "SNSTopicListAccess",
      "Effect" : "Allow",
      "Action" : [
        "sns:ListTopics"
      ],
      "Resource" : "*"
    }
```

자세한 내용은 AWS CodeStar 알림의 Identity and Access Management를 참조하세요.

AWS 관리형 정책에 대한 CodeDeploy 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 CodeDeploy의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 CodeDeploy 문서 기록에서 RSS 피드를 구독하세요.

변경	설명	Date
AWSCodeDeployDeployerAccess 관리형 정책 – 기존 정책에 대한 업데이트	IAM 정책 검증 변경을 지원하도록 codestar-notifications:NotificationsForResource 작업을 업데이트했습니다. 원본 리소스 arn:aws:codedeploy:* 가	2024년 12월 16일

변경	설명	Date
	`arn:aws:codedeploy:*:*:application:*` 으로 업데이트되었습니다. 이 정책에 대한 자세한 내용은 다른 관리형 정책의 알림에 대한 권한 섹션을 참조하세요.	
`AWSCodeDeployReadOnlyAccess` 관리형 정책 – 기존 정책에 대한 업데이트	IAM 정책 검증 변경을 지원하도록 `codestar-notifications:NotificationsForResource` 작업을 업데이트했습니다. 원본 리소스 `arn:aws:codedeploy:*` 가 `arn:aws:codedeploy:*:*:application:*` 으로 업데이트되었습니다. 이 정책에 대한 자세한 내용은 읽기 전용 관리형 정책의 알림에 대한 권한 섹션을 참조하세요.	2024년 12월 16일

변경	설명	Date
AWSCodeDeployFullAccess 관리형 정책 – 기존 정책에 대한 업데이트	IAM 정책 검증 변경을 지원하도록 codestar-notifications:NotificationsForResource 작업을 업데이트했습니다. 원본 리소스 arn:aws:codedeploy:* 가 arn:aws:codedeploy:*:*:application:* 으로 업데이트되었습니다. 이 정책에 대한 자세한 내용은 전체 액세스 관리형 정책의 알림에 대한 권한 섹션을 참조하세요.	2024년 12월 16일
AWSCodeDeployRole 관리형 정책 – 기존 정책에 대한 업데이트	Elastic Load Balancing 변경을 지원하는 elasticloadbalancing:DescribeLoadBalancerAttributes 및 elasticloadbalancing:DescribeTargetGroupAttributes 작업을 정책 문에 추가했습니다. 이 정책에 대한 자세한 내용은 AWSCodeDeployRole 섹션을 참조하세요.	2023년 8월 16일

변경	설명	Date
AWSCodeDeployFullAccess 관리형 정책 – 기존 정책에 대한 업데이트	알림 변경을 지원하는 chatbot:ListMicrosoftTeamsChannelConfigurations 작업을 정책 문에 추가했습니다. 이 정책에 대한 자세한 내용은 AWSCodeDeployRole 섹션을 참조하세요.	2023년 5월 11일
AWSCodeDeployRole 관리형 정책 – 기존 정책에 대한 업데이트	Amazon EC2 Auto Scaling 권한 변경을 지원하기 위해 autoscaling:CreateOrUpdateTags 작업을 정책 설명에 추가했습니다. 이 정책에 대한 자세한 내용은 AWSCodeDeployRole 섹션을 참조하세요.	2023년 2월 3일
AmazonEC2RoleforAWSCodeDeployLimited 관리형 정책 – 기존 정책에 대한 업데이트	정책 문에서 s3:ExistingObjectTag/UseWithCodeDeploy 조건을 포함하는 s3:ListBucket 작업을 제거합니다. 이 정책에 대한 자세한 내용은 AmazonEC2RoleforAWSCodeDeployLimited 섹션을 참조하세요.	2021년 11월 22일

변경	설명	Date
`AWSCodeDeployRole` 관리형 정책 – 기존 정책에 대한 업데이트	블루/그린 배포를 위해 Amazon EC2 Auto Scaling 그룹에 웜 풀 추가를 지원하는 `autoscaling:PutWarmPool` 작업이 추가되었습니다. 불필요한 중복 작업이 제거되었습니다.	2021년 5월 18일
CodeDeploy가 변경 내용 추적 시작	CodeDeploy는 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 5월 18일

AWS CodeDeploy 자격 증명 기반 정책 예제

기본적으로 사용자에게는 CodeDeploy 리소스를 생성하거나 수정하는 권한이 없습니다. 또한 AWS Management Console AWS CLI또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 필요한 지정된 리소스에서 API 작업을 수행할 수 있는 권한을 IAM 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 그 IAM 역할을 해당 권한이 필요한 사용자 또는 그룹에 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하세요.

CodeDeploy에서는 자격 증명 기반 정책을 사용하여 배포 프로세스에 관련된 다양한 리소스에 대한 권한을 관리합니다. 다음 리소스 유형에 대한 액세스를 제어할 수 있습니다.

- 애플리케이션 및 애플리케이션 개정
- 배포
- 배포 구성
- 인스턴스 및 온프레미스 인스턴스

다음 표에서 보여주는 것처럼 리소스 정책에 따라 제어되는 기능은 리소스 유형에 따라 다릅니다.

조건 키	기능
모두	리소스에 대한 세부 정보 보기 및 나열
애플리케이션 배포 구성 배포 그룹	리소스 만들기 리소스 삭제하기
배포	배포 만들기 배포 중지
애플리케이션 개정	애플리케이션 개정 등록
애플리케이션 배포 그룹	리소스 업데이트
온프레미스 인스턴스	인스턴스에 태그 추가 인스턴스에서 태그 제거 인스턴스 등록 인스턴스 등록 취소

다음 예제는 사용자가 **us-west-2** 리전의 **WordPress_App**이라는 애플리케이션과 연결된 **WordPress_DepGroup**이라는 배포 그룹을 삭제할 수 있도록 허용하는 권한 정책을 보여줍니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:DeleteDeploymentGroup"
```

```
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      ]
    }
  ]
}
```

주제

- 고객 관리형 정책 예제
- 정책 모범 사례
- CodeDeploy 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

고객 관리형 정책 예제

이 섹션에서는 다양한 CodeDeploy 작업에 대한 권한을 부여하는 정책의 예를 제공합니다. 이러한 정책은 CodeDeploy API, AWS SDK 또는 AWS CLI를 사용하는 경우에 적용됩니다. 콘솔에서 수행하는 작업에 대한 추가 권한을 부여해야 합니다. 콘솔 권한 부여에 대한 자세한 내용은 CodeDeploy 콘솔 사용 단원을 참조하세요.

> **Note**
>
> 모든 예에서는 미국 서부(오리건) 리전(us-west-2)을 사용하며 가상의 계정 ID를 포함합니다.

예제

- 예 1: 단일 리전에서 CodeDeploy 작업을 수행하도록 권한 허용
- 예 2: 단일 애플리케이션에 대한 개정 버전을 등록하도록 권한 허용
- 예 3: 단일 배포 그룹에 대한 배포를 만들도록 권한 허용

예 1: 단일 리전에서 CodeDeploy 작업을 수행하도록 권한 허용

다음 예제에서는 **us-west-2** 리전에서만 CodeDeploy 작업을 수행할 수 있는 권한을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:*"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:*"
      ]
    }
  ]
}
```

예 2: 단일 애플리케이션에 대한 개정 버전을 등록하도록 권한 허용

다음 예제에서는 **us-west-2** 리전에서 **Test**로 시작하는 모든 애플리케이션의 애플리케이션 개정을 등록할 수 있는 권한을 부여합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:RegisterApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:application:Test*"
      ]
    }
  ]
```

```
}
```

예 3: 단일 배포 그룹에 대한 배포를 만들도록 권한 허용

다음 예제에서는 권한이 **WordPress_App**이라는 애플리케이션과 관련된 배포 그룹 **WordPress_DepGroup**에 대한 배포, **ThreeQuartersHealthy**라는 사용자 지정 배포 구성, **WordPress_App**이라는 애플리케이션과 연결된 모든 애플리케이션 개정 버전을 만들도록 허용합니다. 이러한 모든 리소스는 **us-west-2** 리전에 있습니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:CreateDeployment"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:deploymentgroup:WordPress_App/
WordPress_DepGroup"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetDeploymentConfig"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-
west-2:111122223333:deploymentconfig:ThreeQuartersHealthy"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "codedeploy:GetApplicationRevision"
      ],
      "Resource" : [
        "arn:aws:codedeploy:us-west-2:111122223333:application:WordPress_App"
```

```
      ]
    }
  ]
}
```

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 CodeDeploy 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

CodeDeploy 콘솔 사용

CodeDeploy 콘솔을 사용하는 경우 AWS 계정의 다른 AWS 리소스를 설명할 수 있는 최소 권한 집합이 있어야 합니다. CodeDeploy 콘솔에서 CodeDeploy를 사용하려면 다음 서비스의 권한이 있어야 합니다.

- Amazon EC2 오토 스케일링
- AWS CodeDeploy
- - Amazon Elastic Compute Cloud
- Elastic Load Balancing
- AWS Identity and Access Management
- Amazon Simple Storage Service
- Amazon Simple Notification Service
- Amazon CloudWatch

최소 필수 권한보다 더 제한적인 IAM 정책을 만들면 해당 IAM 정책을 보유한 사용자의 경우 콘솔이 의도대로 작동하지 않습니다. 해당 사용자가 여전히 CodeDeploy 콘솔을 사용할 수 있도록 하려면 AWS CodeDeploy에 대한 관리형(미리 정의된) 정책에 설명된 대로 해당 사용자에게 할당된 역할에 `AWSCodeDeployReadOnlyAccess` 관리형 정책도 연결하세요.

AWS CLI 또는 CodeDeploy API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
```

```
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS CodeDeploy 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 CodeDeploy 및 IAM 작업 시 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- iam:PassRole을 수행하도록 인증되지 않음
- 내 AWS 계정 외부의 사람이 내 CodeDeploy 리소스에 액세스하도록 허용하고 싶습니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 CodeDeploy에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 CodeDeploy에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 내 CodeDeploy 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- CodeDeploy에서 이러한 기능을 지원하는지 여부를 알아보려면 AWS CodeDeploy 에서 IAM을 사용하는 방법 섹션을 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

CodeDeploy 권한 참조

IAM 자격 증명에 연결할 수 있는 액세스 및 쓰기 권한 정책(자격 증명 기반 정책)을 설정할 때 다음 표를 사용합니다. 이 표에는 각 CodeDeploy API 작업, 수행 권한을 부여할 수 있는 작업, 권한을 부여하는 데 사용할 리소스 ARN의 형식이 나열되어 있습니다. 정책의 Action 필드에 작업을 지정합니다. 정

책의 Resource 필드에 리소스 값으로 와일드카드 문자(*)를 사용하거나 사용하지 않고 ARN을 지정합니다.

CodeDeploy 정책에서 AWS전체 조건 키를 사용하여 조건을 표시할 수 있습니다. AWS전체 키의 전체 목록은 IAM 사용 설명서의 사용 가능한 키를 참조하세요.

작업을 지정하려면 codedeploy: 접두사 다음에 API 작업 이름을 사용합니다(예: codedeploy:GetApplication 및 codedeploy:CreateApplication). 문장 하나에 여러 작업을 지정하려면 쉼표로 구분합니다(예: "Action": ["codedeploy:action1", "codedeploy:action2"]).

와일드카드 문자 사용

ARN에 와일드카드(*)를 사용하여 여러 작업 또는 리소스를 지정할 수 있습니다. 예를 들어, codedeploy:*는 모든 CodeDeploy 작업을 지정하고, codedeploy:Get*는 Get이라는 단어로 시작하는 모든 CodeDeploy 작업을 지정합니다. 다음 예제에서는 이름이 West로 시작하는 배포 그룹 중에서 이름이 Test로 시작하는 애플리케이션과 연결된 모든 배포 그룹에 대한 액세스 권한을 부여합니다.

```
arn:aws:codedeploy:us-west-2:444455556666:deploymentgroup:Test*/West*
```

표에 나열된 다음 리소스에 와일드카드를 사용할 수 있습니다.

- *application-name*
- *deployment-group-name*
- *deployment-configuration-name*
- *instance-ID*

와일드카드는 *region* 또는 *account-id*에는 사용할 수 없습니다. 와일드카드에 대한 자세한 내용은 IAM 사용 설명서에서 IAM 식별자를 참조하세요.

> **Note**
>
> 각 작업의 ARN에는 리소스 다음에 콜론(:)이 있습니다. 이 리소스 다음에 슬래시(/)가 올 수도 있습니다. 자세한 내용은 CodeDeploy 예제 ARN을 참조하세요.

CodeDeploy API 작업 및 작업에 필요한 권한

AddTagsToOnPremisesInstances

작업: codedeploy:AddTagsToOnPremisesInstances

하나 이상의 온프레미스 인스턴스에 태그를 추가하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

BatchGetApplicationRevisions

작업: codedeploy:BatchGetApplicationRevisions

사용자와 연결된 여러 애플리케이션 개정에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

BatchGetApplications

작업: codedeploy:BatchGetApplications

사용자와 연결된 여러 애플리케이션에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*

BatchGetDeploymentGroups

작업: codedeploy:BatchGetDeploymentGroups

사용자와 연결된 여러 배포 그룹에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeploymentInstances

작업: codedeploy:BatchGetDeploymentInstances

배포 그룹의 일부인 하나 이상의 인스턴스에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetDeployments

작업: codedeploy:BatchGetDeployments

사용자와 연결된 여러 배포에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

BatchGetOnPremisesInstances

작업: codedeploy:BatchGetOnPremisesInstances

하나 이상의 온프레미스 인스턴스에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:*

ContinueDeployment

작업: codedeploy:ContinueDeployment

블루/그린 배포 중 대체 환경에서 Elastic Load Balancing 로드 밸런서를 사용하여 인스턴스 등록을 시작하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

CreateApplication

작업: codedeploy:CreateApplication

사용자와 연결된 애플리케이션을 만드는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

CreateDeployment [1]

작업: codedeploy:CreateDeployment

사용자와 연결된 애플리케이션에 대한 배포를 만드는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

CreateDeploymentConfig

작업: codedeploy:CreateDeploymentConfig

사용자와 연결된 사용자 지정 배포 구성을 만드는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

CreateDeploymentGroup

작업: codedeploy:CreateDeploymentGroup

사용자와 연결된 애플리케이션에 대한 배포 그룹을 만드는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

DeleteApplication

작업: codedeploy:DeleteApplication

사용자와 연결된 애플리케이션을 삭제하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

DeleteDeploymentConfig

작업: codedeploy:DeleteDeploymentConfig

사용자와 연결된 사용자 지정 배포 구성을 삭제하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

DeleteDeploymentGroup

작업: codedeploy:DeleteDeploymentGroup

사용자와 연결된 애플리케이션에 대한 배포 그룹을 삭제하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

DeregisterOnPremisesInstance

작업: codedeploy:DeregisterOnPremisesInstance

온프레미스 인스턴스를 등록 취소하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

GetApplication

작업: codedeploy:GetApplication

사용자와 연결된 단일 애플리케이션에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

GetApplicationRevision

작업: codedeploy:GetApplicationRevision

사용자와 연결된 애플리케이션의 단일 애플리케이션 개정에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

GetDeployment

작업: codedeploy:GetDeployment

사용자와 연결된 애플리케이션의 배포 그룹에 대한 단일 배포의 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetDeploymentConfig

작업: codedeploy:GetDeploymentConfig

사용자와 연결된 단일 배포 구성에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*deployment-configuration-name*

GetDeploymentGroup

작업: codedeploy:GetDeploymentGroup

사용자와 연결된 애플리케이션의 단일 배포 그룹에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetDeploymentInstance

작업: codedeploy:GetDeploymentInstance

사용자와 연결된 배포의 단일 인스턴스에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

GetOnPremisesInstance

작업: codedeploy:GetOnPremisesInstance

단일 온프레미스 인스턴스에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

ListApplicationRevisions

작업: codedeploy:ListApplicationRevisions

사용자와 연결된 애플리케이션의 모든 애플리케이션 개정에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*

ListApplications

작업: codedeploy:ListApplications

사용자와 연결된 모든 애플리케이션에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*

ListDeploymentConfigs

작업: codedeploy:ListDeploymentConfigs

사용자와 연결된 모든 배포 구성에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentconfig/*

ListDeploymentGroups

작업: codedeploy:ListDeploymentGroups

사용자와 연결된 애플리케이션의 모든 배포 그룹에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*

ListDeploymentInstances

작업: codedeploy:ListDeploymentInstances

사용자와 연결된 배포의 모든 인스턴스에 대한 정보를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListDeployments

작업: codedeploy:ListDeployments

사용자와 연결된 배포 그룹에 대한 모든 배포의 정보를 가져오거나 사용자와 연결된 모든 배포를 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

ListGitHubAccountTokenNames

작업: codedeploy:ListGitHubAccountTokenNames

GitHub 계정에 대한 저장된 연결의 이름 목록을 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:*

ListOnPremisesInstances

작업: codedeploy:ListOnPremisesInstances

하나 이상의 온프레미스 인스턴스 이름 목록을 가져오는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:*

RegisterApplicationRevision

작업: codedeploy:RegisterApplicationRevision

사용자와 연결된 애플리케이션의 애플리케이션 개정에 대한 정보를 등록하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

RegisterOnPremisesInstance

작업: codedeploy:RegisterOnPremisesInstance

CodeDeploy에 온프레미스 인스턴스를 등록하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

RemoveTagsFromOnPremisesInstances

작업: codedeploy:RemoveTagsFromOnPremisesInstances

하나 이상의 온프레미스 인스턴스에서 태그를 제거하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

SkipWaitTimeForInstanceTermination

작업: codedeploy:SkipWaitTimeForInstanceTermination

블루/그린 배포에서 트래픽이 성공적으로 라우팅된 후 즉시 지정된 대기 시간을 재정의하고 원래 환경에서 인스턴스 종료를 시작하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:instance/*instance-ID*

StopDeployment

작업: codedeploy:StopDeployment

사용자와 연결된 애플리케이션의 배포 그룹에 대해 진행 중인 배포를 중지하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

UpdateApplication [3]

작업: codedeploy:UpdateApplication

사용자와 연결된 애플리케이션에 대한 정보를 변경하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:application:*application-name*

UpdateDeploymentGroup [3]

작업: codedeploy:UpdateDeploymentGroup

사용자와 연결된 애플리케이션의 단일 배포 그룹에 대한 정보를 변경하는 데 필요합니다.

리소스: arn:aws:codedeploy:*region*:*account-id*:deploymentgroup:*application-name*/*deployment-group-name*

[1] CreateDeployment 권한을 지정할 때 배포 구성에 대한 GetDeploymentConfig 권한과 애플리케이션 개정에 대한 GetApplicationRevision 또는 RegisterApplicationRevision 권한도 지정해야 합니다.

[2] 배포 그룹을 제공하는 경우 ListDeployments에 유효하지만, 사용자와 연결된 모든 배포를 나열하는 경우에는 유효하지 않습니다.

[3] UpdateApplication의 경우 이전 애플리케이션 이름과 새 애플리케이션 이름 모두에 대한 UpdateApplication 권한이 있어야 합니다. 배포 그룹의 이름 변경과 관련된

UpdateDeploymentGroup 작업의 경우 이전 배포 그룹 이름과 새 배포 그룹 이름 모두에 대한 UpdateDeploymentGroup 권한이 있어야 합니다.

교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 AWS 제공합니다.

CodeDeploy가 리소스에 다른 서비스를 제공하는 권한을 제한하려면 리소스 정책에서 aws:SourceArn 및 aws:SourceAccount 글로벌 조건 컨텍스트 키를 사용하는 것이 좋습니다. 두 전역 조건 컨텍스트 키와 계정을 포함한 aws:SourceArn 값을 모두 사용하는 경우, aws:SourceAccount 값 및 aws:SourceArn 값의 계정은 동일한 정책 명령문에서 사용할 경우 반드시 동일한 계정 ID를 사용해야 합니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 aws:SourceArn을 사용하세요. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 하려면 aws:SourceAccount를 사용하세요.

EC2/온프레미스, AWS Lambda 및 일반 Amazon ECS 배포의 경우의 값에는 CodeDeploy가 IAM 역할을 수임할 수 있는 CodeDeploy 배포 그룹 ARN이 포함되어야 aws:SourceArn 합니다.

를 통해 생성된 Amazon ECS 블루/그린 배포 CloudFormation의 경우의 값에 CodeDeploy가 IAM 역할을 수임할 수 있는 CloudFormation 스택 ARN이 포함되어야 aws:SourceArn 합니다.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 aws:SourceArn 키를 사용하는 것입니다. 전체 ARN을 모르거나 여러 리소스를 지정하는 경우, 알 수 없는 부분에 대해 와일드카드 문자(*)를 사용합니다.

예를 들어 EC2/온프레미스, AWS Lambda 또는 일반 Amazon ECS 배포와 함께 다음 신뢰 정책을 사용할 수 있습니다.

JSON

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

```
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:codedeploy:us-
east-1:111122223333:deploymentgroup:myApplication/*"
                }
            }
        }
    ]
}
```

를 통해 생성된 Amazon ECS 블루/그린 배포 CloudFormation의 경우 다음을 사용할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "codedeploy.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudformation:us-
east-1:111122223333:stack/MyCloudFormationStackName/*"
                }
```

```
            }
        }
    ]
}
```

CodeDeploy에서 로깅 및 모니터링

이 섹션에서는 CodeDeploy의 모니터링, 로깅 및 인시던트 대응에 대한 개요를 제공합니다.

CodeDeploy와의 모든 상호 작용에 대한 감사

CodeDeploy는 AWS 계정에서 CodeDeploy에 의해 또는 CodeDeploy를 대신하여 수행된 API 호출을 캡처하고 지정한 S3 버킷으로 로그 파일을 전송하는 AWS CloudTrail서비스와 통합됩니다. CloudTrail은 CodeDeploy 콘솔, AWS CLI를 통한 CodeDeploy 명령 또는 CodeDeploy API에서 직접 API 호출을 캡처합니다. CloudTrail에서 수집하는 정보를 사용하여 CodeDeploy에 어떤 요청이 이루어졌는지, 어떤 소스 IP 주소에서 요청했는지, 누가 언제 요청했는지 등을 확인할 수 있습니다. CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 로그 파일 작업을 참조하세요.

CloudWatch 콘솔에서 집계 데이터를 보도록 Amazon CloudWatch 에이전트를 설정하거나 로그 파일을 검토하기 위해 인스턴스에 로그인하여 CodeDeploy 배포에서 생성된 로그 데이터를 볼 수 있습니다. 자세한 내용은 CloudWatch로 CodeDeploy 에이전트 로그 전송 섹션을 참조하세요.

알림 및 인시던트 관리

Amazon CloudWatch Events를 사용하여 CodeDeploy 작업에서 인스턴스 또는 배포의 상태 변화(이벤트)를 감지하고 그에 대응할 수 있습니다. 그런 다음 CloudWatch Events는 사용자가 만든 규칙에 따라 배포 또는 인스턴스가 규칙에 지정된 상태로 전환되면 하나 이상의 대상 작업을 호출합니다. 상태 변경 유형에 따라 알림을 보내거나, 상태 정보를 캡처하거나, 교정 작업을 수행하거나, 이벤트를 시작하거나, 기타 작업을 수행할 수 있습니다. CloudWatch Events를 CodeDeploy 작업의 일부로 사용하는 경우 다음과 같은 대상 유형을 선택할 수 있습니다.

- AWS Lambda 함수
- Kinesis 스트림
- Amazon SQS 대기열
- 기본 제공 대상(CloudWatch 경보 작업)
- Amazon SNS 주제

다음은 몇 가지 사용 사례입니다.

- 배포에 실패할 때마다 Lambda 함수를 사용하여 Slack 채널에 알림을 전달합니다.
- 배포 또는 인스턴스에 대한 데이터를 Kinesis 스트림으로 푸시하여 포괄적인 실시간 상태 모니터링을 지원합니다.
- CloudWatch 경보 작업을 사용하여, 지정한 배포 또는 인스턴스 이벤트가 발생하면 EC2 인스턴스를 자동으로 중지, 종료, 재부팅 또는 복구합니다.

자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch Events란 무엇인가요를 참조하세요.

AWS CodeDeploy에 대한 규정 준수 검증

새로운 AWS 가입 환경은 규제 워크로드용으로 설계되지 않았습니다. 새로운 AWS 가입 환경을 사용하고 있지만 규제 대상 워크로드 AWS 에를 사용하려는 경우 AWS 환경에 가입 AWS (고급)하거나 고급 기능을 활성화할 수 있습니다.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

Resilience in AWS CodeDeploy

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 많으며 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.

인프라 보안 in AWS CodeDeploy

관리형 서비스인 AWS CodeDeploy는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차로 보호됩니다. https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 CodeDeploy에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. TLS 1.3 이상을 권장합니다. 클라이언트는 DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Diffie-Hellman Ephemeral)와 같은 PFS(전달 완전 보안)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

요청은 액세스 키 ID 및 IAM 보안 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

레퍼런스

참조

주제

- CodeDeploy AppSpec 파일 참조
- CodeDeploy 에이전트 구성 참조
- CloudFormation CodeDeploy 참조용 템플릿
- Amazon Virtual Private Cloud와 함께 CodeDeploy 사용
- CodeDeploy 리소스 키트 참조
- CodeDeploy 할당량

CodeDeploy AppSpec 파일 참조

이 단원은 참조용입니다. AppSpec 파일에 대한 개념적 개요는 Application Specification Files 단원을 참조하세요.

애플리케이션 사양 파일(AppSpec file)은 CodeDeploy에서 배포를 관리하는 데 사용하는 YAML 형식 또는 JSON 형식의 파일입니다.

> **Note**
>
> 로컬 배포를 수행하지 않는 한 EC2/온프레미스 배포에 대한 AppSpec 파일의 이름은 `appspec.yml`입니다. 자세한 내용은 로컬 배포 생성 섹션을 참조하세요.

주제

- Amazon ECS 컴퓨팅 플랫폼에 대한 AppSpec 파일
- AWS Lambda 컴퓨팅 플랫폼의 AppSpec 파일
- EC2/온프레미스 컴퓨팅 플랫폼에 대한 AppSpec 파일
- AppSpec 파일 구조
- AppSpec 파일 예
- AppSpec 파일 간격

- AppSpec 파일 및 파일 위치 확인

Amazon ECS 컴퓨팅 플랫폼에 대한 AppSpec 파일

Amazon ECS 컴퓨팅 플랫폼 애플리케이션의 경우 CodeDeploy에서 AppSpec 파일을 사용하여 다음을 결정합니다.

- Amazon ECS 작업 정의 파일 AppSpec 파일에서 `TaskDefinition` 지침의 ARN으로 지정됩니다.
- 배포 중에 Application Load Balancer 또는 Network Load Balancer가 트래픽을 다시 라우팅하는 대체 작업 세트의 컨테이너 및 포트. AppSpec 파일의 `LoadBalancerInfo` 지침으로 지정됩니다.
- Amazon ECS 서비스에 대한 선택적 정보(예: 실행되는 플랫폼 버전, 서브넷, 보안 그룹).
- Amazon ECS 배포 중에 수명 주기 이벤트에 해당하는 후크 중에 실행할 Lambda 함수(선택 사항). 자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션 섹션을 참조하세요.

AWS Lambda 컴퓨팅 플랫폼의 AppSpec 파일

AWS Lambda 컴퓨팅 플랫폼 애플리케이션의 경우 CodeDeploy에서 AppSpec 파일을 사용하여 다음을 결정합니다.

- 배포할 Lambda 함수 버전
- 확인 테스트로 사용할 Lambda 함수

AppSpec 파일은 YAML 형식 또는 JSON 형식일 수 있습니다. 배포를 생성할 때 AppSpec 파일의 내용을 CodeDeploy 콘솔에 직접 입력할 수도 있습니다.

EC2/온프레미스 컴퓨팅 플랫폼에 대한 AppSpec 파일

애플리케이션이 EC2/온프레미스 컴퓨팅 플랫폼을 사용하는 경우 AppSpec 파일은 `appspec.yml`이라는 YAML 형식의 파일이며 애플리케이션의 소스 코드 디렉터리 구조의 루트에 배치해야 합니다. 그렇지 않으면 배포에 실패합니다. CodeDeploy에서 사용하여 다음 사항을 확인합니다.

- Amazon S3 또는 GitHub의 애플리케이션 수정에서 인스턴스에 설치해야 할 항목
- 배포 수명 주기 이벤트에 대한 응답으로 실행될 수명 주기 이벤트 후크

AppSpec 파일을 완성한 후 배포할 콘텐츠와 함께 아카이브 파일(zip, tar 또는 압축 tar)로 번들링합니다. 자세한 내용은 CodeDeploy의 애플리케이션 개정 작업 섹션을 참조하세요.

> **Note**
>
> tar 및 압축된 tar 아카이브 파일 형식(.tar 및 .tar.gz)은 Windows Server 인스턴스에서 지원되지 않습니다.

번들링된 아카이브 파일(CodeDeploy에서는 수정 버전이라고 함)이 생성되면 Amazon S3 버킷 또는 Git 리포지토리에 업로드합니다. 그런 다음 CodeDeploy를 사용하여 수정을 배포합니다. 지침은 CodeDeploy에서 배포 만들기 단원을 참조하세요.

EC2/온프레미스 컴퓨팅 플랫폼 배포용 appspec.yml은 수정 버전의 루트 디렉터리에 저장됩니다. 자세한 내용은 EC2 온프레미스 배포용 AppSpec 파일 추가 및 CodeDeploy의 개정 계획 단원을 참조하세요.

AppSpec 파일 구조

다음은 AWS Lambda 및 EC2/온프레미스 컴퓨팅 플랫폼으로의 배포에 사용되는 을 간단히 나타낸 것입니다.

문자열인 YAML 형식 AppSpec 파일의 값은 다른 방식으로 지정되지 않는 한 따옴표("")로 둘러싸면 안 됩니다.

Amazon ECS 배포의 AppSpec 파일 구조

> **Note**
>
> 이 AppSpec 파일은 YAML로 작성되지만, 동일한 구조를 사용하여 JSON으로 작성할 수 있습니다. JSON 형식 AppSpec 파일의 문자열은 항상 따옴표("")로 둘러싸야 합니다.

```
version: 0.0
resources:
  ecs-service-specifications
hooks:
  deployment-lifecycle-event-mappings
```

이 구조에서:

version

이 섹션에서는 AppSpec 파일의 버전을 지정합니다. 이 값은 변경하지 마세요. 필수 항목입니다. 현재, 유일하게 허용되는 값은 **0.0**입니다. 이 값은 이후에 사용하기 위해 CodeDeploy에서 예약한 값입니다.

문자열로 version을 지정합니다.

resources

이 섹션은 배포할 Amazon ECS 애플리케이션에 대한 정보를 지정합니다.

자세한 내용은 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션 섹션을 참조하세요.

hooks

이 섹션은 배포를 확인하기 위해 특정 배포 수명 주기 이벤트 후크에서 실행할 Lambda 함수를 지정합니다.

자세한 내용은 Amazon ECS 배포를 위한 수명 주기 이벤트 후크 목록 섹션을 참조하세요.

AWS Lambda 배포를 위한 AppSpec 파일 구조

> **Note**
>
> 이 AppSpec 파일은 YAML로 작성되지만, 동일한 구조를 사용하여 JSON으로 Lambda 배포용 AppSpec 파일을 작성할 수 있습니다. JSON 형식 AppSpec 파일의 문자열은 항상 따옴표("")로 둘러싸야 합니다.

```
version: 0.0
resources:
  lambda-function-specifications
hooks:
  deployment-lifecycle-event-mappings
```

이 구조에서:

version

이 섹션에서는 AppSpec 파일의 버전을 지정합니다. 이 값은 변경하지 마세요. 필수 항목입니다. 현재, 유일하게 허용되는 값은 **0.0**입니다. 이 값은 이후에 사용하기 위해 CodeDeploy에서 예약한 값입니다.

문자열로 version을 지정합니다.

resources

이 섹션은 배포할 Lambda 함수에 대한 정보를 지정합니다.

자세한 내용은 AppSpec 'resources' 섹션(Amazon ECS 및 AWS Lambda 배포만 해당) 섹션을 참조하세요.

hooks

이 섹션은 배포를 확인하기 위해 특정 배포 수명 주기 이벤트에서 실행할 Lambda 함수를 지정합니다.

자세한 내용은 AppSpec 'hooks' 섹션 섹션을 참조하세요.

EC2 온프레미스 배포용 AppSpec 파일 구조

```
version: 0.0
os: operating-system-name
files:
  source-destination-files-mappings
permissions:
  permissions-specifications
hooks:
  deployment-lifecycle-event-mappings
```

이 구조에서:

version

이 섹션에서는 AppSpec 파일의 버전을 지정합니다. 이 값은 변경하지 마세요. 필수 항목입니다. 현재, 유일하게 허용되는 값은 **0.0**입니다. 이 값은 이후에 사용하기 위해 CodeDeploy에서 예약한 값입니다.

문자열로 version을 지정합니다.

os

이 섹션은 배포할 인스턴스의 운영 체제 값을 지정합니다. 필수 항목입니다. 다음 값을 지정할 수 있습니다.

- Linux - 인스턴스가 Amazon Linux, Ubuntu Server, RHEL 또는 macOS 인스턴스입니다.
- Windows – Windows Server 인스턴스입니다.

문자열로 os를 지정합니다.

files

이 섹션에서는 배포의 설치 이벤트 중 인스턴스에 복사해야 하는 파일의 이름을 지정합니다.

자세한 내용은 AppSpec 'files' 섹션(EC2/온프레미스 배포만 해당) 섹션을 참조하세요.

권한

이 섹션은 `files` 섹션의 파일이 인스턴스에 복사되는 경우 이러한 파일에 특수 권한(있는 경우)이 어떻게 적용되어야 하는지를 지정합니다. 이 섹션은 Amazon Linux, Ubuntu Server, Red Hat Enterprise Linux(RHEL) 및 macOS 인스턴스에만 적용됩니다.

자세한 내용은 단원을 참조하십시오AppSpec 'permissions' 섹션(EC2/온프레미스 배포만 해당).

hooks

이 섹션은 배포 중 특정 배포 수명 주기 이벤트에서 실행되는 스크립트를 지정합니다.

자세한 내용은 AppSpec 'hooks' 섹션 섹션을 참조하세요.

주제

- AppSpec 'files' 섹션(EC2/온프레미스 배포만 해당)
- AppSpec 'resources' 섹션(Amazon ECS 및 AWS Lambda 배포만 해당)
- AppSpec 'permissions' 섹션(EC2/온프레미스 배포만 해당)
- AppSpec 'hooks' 섹션

AppSpec 'files' 섹션(EC2/온프레미스 배포만 해당)

배포의 설치이벤트 동안 인스턴스에 설치해야 하는 애플리케이션 수정 버전의 파일에 대한 정보를 CodeDeploy에 제공합니다. 이 섹션은 배포 중 수정의 파일을 인스턴스의 위치로 복사하는 경우에만 필요합니다.

이 섹션의 구조는 다음과 같습니다.

```
files:
  - source: source-file-location-1
    destination: destination-file-location-1
file_exists_behavior: DISALLOW|OVERWRITE|RETAIN
```

여러 개의 source 및 destination 페어를 설정할 수 있습니다.

source 명령은 인스턴스에 복사할 수정의 파일 또는 디렉터리를 식별합니다.

- source가 파일을 나타내는 경우 지정한 파일만 인스턴스에 복사됩니다.
- source가 디렉터리를 나타내는 경우 디렉터리 내의 모든 파일이 인스턴스에 복사됩니다.
- source이(가) 슬래시 하나인 경우(Amazon Linux, RHEL 및 Ubuntu Server 인스턴스의 경우: "/", Windows Server 인스턴스의 경우: "\") 수정 버전의 모든 파일이 인스턴스에 복사됩니다.

source에 사용된 경로는 appspec.yml 파일에 상대적이며 수정 버전의 루트에 있어야 합니다. 수정 버전의 파일 구조에 대한 자세한 내용은 CodeDeploy의 개정 계획 단원을 참조하세요.

destination 명령은 인스턴스에서 파일이 복사되어야 하는 위치를 식별합니다. /root/destination/directory(Linux, RHEL, Ubuntu) 또는 c:\destination\folder(Windows)와 같은 정규화된 경로여야 합니다.

source 및 destination은 각각 문자열을 사용하여 지정됩니다.

file_exists_behavior 지침은 선택 사항이며, CodeDeploy가 배포 대상 위치에 이미 존재하지만 이전에 성공한 배포의 일부가 아닌 파일을 처리하는 방식을 지정합니다. 이 설정은 다음 값 중 하나일 수 있습니다.

- DISALLOW: 배포가 실패합니다. 이는 옵션을 지정하지 않은 경우의 기본 동작입니다.
- OVERWRITE: 현재 배포 중인 애플리케이션 수정 버전의 파일 버전이 인스턴스에 이미 있는 버전을 대체합니다.
- RETATE: 인스턴스에 이미 있는 파일의 버전이 유지되고 새 배포의 일부로 사용됩니다.

file_exists_behavior 설정을 사용하는 경우 다음 설정을 이해합니다.

- 한 번만 지정할 수 있으며 files: 아래에 나열된 모든 파일 및 디렉터리에 적용됩니다.

- 는 `--file-exists-behavior` AWS CLI 옵션 및 `fileExistsBehavior` API 옵션보다 우선합니다(둘 다 선택 사항임).

다음은 Amazon Linux, Ubuntu Server 또는 RHEL 인스턴스에 대한 `files` 섹션의 예입니다.

```
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
```

이 예에서는 설치 이벤트 중 다음 두 가지 작업이 수행됩니다.

1. 수정의 `Config/config.txt` 파일을 인스턴스의 `/webapps/Config/config.txt` 경로에 복사합니다.
2. 수정의 `source` 디렉터리에 있는 파일을 인스턴스의 `/webapps/myApp` 디렉터리로 모두 복사합니다.

'Files' 섹션의 예

다음 예제에서는 `files` 섹션을 지정하는 방법을 보여줍니다. 이러한 예에서는 Windows Server 파일 및 디렉터리(폴더) 구조를 설명하고 있지만 이러한 구조는Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에 대해 쉽게 조정할 수 있습니다.

> **Note**
>
> EC2/온프레미스 배포만 `files` 섹션을 사용합니다. AWS Lambda 배포에는 적용되지 않습니다.

다음 예에서는 이러한 파일이 `source`의 루트에 번들로 나타난다고 가정합니다.

- `appspec.yml`
- `my-file.txt`
- `my-file-2.txt`
- `my-file-3.txt`

```
# 1) Copy only my-file.txt to the destination folder c:\temp.
#
files:
  - source: .\my-file.txt
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#
# ----------------------
#
# 2) Copy only my-file-2.txt and my-file-3.txt to the destination folder c:\temp.
#
files:
  - source: my-file-2.txt
    destination: c:\temp
  - source: my-file-3.txt
    destination: c:\temp
#
# Result:
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ----------------------
#
# 3) Copy my-file.txt, my-file-2.txt, and my-file-3.txt (along with the appspec.yml
 file) to the destination folder c:\temp.
#
files:
  - source: \
    destination: c:\temp
#
# Result:
#   c:\temp\appspec.yml
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
```

다음 예에서는 `appspec.yml`이 `source`의 루트에 번들로 나타나고, 여기에는 파일 3개가 포함된 `my-folder` 폴더가 함께 있습니다.

- `appspec.yml`

- my-folder\my-file.txt
- my-folder\my-file-2.txt
- my-folder\my-file-3.txt

```
# 4) Copy the 3 files in my-folder (but do not copy my-folder itself) to the
 destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp
#
# Result:
#   c:\temp\my-file.txt
#   c:\temp\my-file-2.txt
#   c:\temp\my-file-3.txt
#
# ---------------------
#
# 5) Copy my-folder and its 3 files to my-folder within the destination folder c:\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 6) Copy the 3 files in my-folder to other-folder within the destination folder c:
\temp.
#
files:
  - source: .\my-folder
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file.txt
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
```

```
#
# ---------------------
#
# 7) Copy only my-file-2.txt and my-file-3.txt to my-folder within the destination
 folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\my-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\my-folder
#
# Result:
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
#
# ---------------------
#
# 8) Copy only my-file-2.txt and my-file-3.txt to other-folder within the destination
 folder c:\temp.
#
files:
  - source: .\my-folder\my-file-2.txt
    destination: c:\temp\other-folder
  - source: .\my-folder\my-file-3.txt
    destination: c:\temp\other-folder
#
# Result:
#   c:\temp\other-folder\my-file-2.txt
#   c:\temp\other-folder\my-file-3.txt
#
# ---------------------
#
# 9) Copy my-folder and its 3 files (along with the appspec.yml file) to the
 destination folder c:\temp. If any of the files already exist on the instance,
 overwrite them.
#
files:
  - source: \
    destination: c:\temp
file_exists_behavior: OVERWRITE
#
# Result:
#   c:\temp\appspec.yml
```

```
#   c:\temp\my-folder\my-file.txt
#   c:\temp\my-folder\my-file-2.txt
#   c:\temp\my-folder\my-file-3.txt
```

AppSpec 'resources' 섹션(Amazon ECS 및 AWS Lambda 배포만 해당)

AppSpec 파일의 'resources' 섹션 내용은 해당 배포의 컴퓨팅 플랫폼에 따라 다릅니다. Amazon ECS 배포의 'resources' 섹션에는 Amazon ECS 작업 정의, 업데이트된 Amazon ECS 작업 세트로 트래픽을 라우팅하기 위한 포트 및 컨테이너, 기타 선택적 정보가 포함되어 있습니다. AWS Lambda 배포 'resources' 섹션에는 Lambda 함수의 이름, 별칭, 현재 버전 및 대상 버전이 포함되어 있습니다.

주제

- AWS Lambda 배포를 위한 AppSpec 'resources' 섹션
- Amazon ECS 배포를 위한 AppSpec 'resources' 섹션

AWS Lambda 배포를 위한 AppSpec 'resources' 섹션

'resources' 섹션은 배포할 Lambda 함수를 지정하며 다음과 같은 구조를 갖습니다.

YAML:

```
resources:
  - name-of-function-to-deploy:
      type: "AWS::Lambda::Function"
      properties:
        name: name-of-lambda-function-to-deploy
        alias: alias-of-lambda-function-to-deploy
        currentversion: version-of-the-lambda-function-traffic-currently-points-to
        targetversion: version-of-the-lambda-function-to-shift-traffic-to
```

JSON:

```
"resources": [
    {
        "name-of-function-to-deploy" {
            "type": "AWS::Lambda::Function",
            "properties": {
                "name": "name-of-lambda-function-to-deploy",
                "alias": "alias-of-lambda-function-to-deploy",
```

```
                "currentversion": "version-of-the-lambda-function-traffic-currently-
points-to",
                "targetversion": "version-of-the-lambda-function-to-shift-traffic-to"
            }
        }
    }
]
```

각 속성은 문자열로 지정됩니다.

- name - 필수. 배포할 Lambda 함수의 이름입니다.
- alias - 필수. Lambda 함수에 대한 별칭 이름입니다.
- currentversion - 필수. 현재 가리키는 Lambda 함수 트래픽의 버전입니다. 값은 유효한 양의 정수여야 합니다.
- targetversion - 필수. 전환되는 Lambda 함수 트래픽의 버전입니다. 값은 유효한 양의 정수여야 합니다.

Amazon ECS 배포를 위한 AppSpec 'resources' 섹션

'resources' 섹션은 배포할 Amazon ECS 서비스를 지정하며 다음과 같은 구조를 갖습니다.

YAML:

```
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "task-definition-arn"
        LoadBalancerInfo:
          ContainerName: "ecs-container-name"
          ContainerPort: "ecs-application-port"
# Optional properties
        PlatformVersion: "ecs-service-platform-version"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["ecs-subnet-1","ecs-subnet-n"]
            SecurityGroups: ["ecs-security-group-1","ecs-security-group-n"]
            AssignPublicIp: "ENABLED | DISABLED"
        CapacityProviderStrategy:
          - Base: integer
            CapacityProvider: "capacityProviderA"
```

```
            Weight: integer
          - Base: integer
            CapacityProvider: "capacityProviderB"
            Weight: integer
```

JSON:

```
"Resources": [
    {
        "TargetService": {
            "Type": "AWS::ECS::Service",
            "Properties": {
                "TaskDefinition": "task-definition-arn",
                "LoadBalancerInfo": {
                    "ContainerName": "ecs-container-name",
                    "ContainerPort": "ecs-application-port"
                },
                "PlatformVersion": "ecs-service-platform-version",
                "NetworkConfiguration": {
                    "AwsvpcConfiguration": {
                        "Subnets": [
                            "ecs-subnet-1",
                            "ecs-subnet-n"
                        ],
                        "SecurityGroups": [
                            "ecs-security-group-1",
                            "ecs-security-group-n"
                        ],
                        "AssignPublicIp": "ENABLED | DISABLED"
                    }
                },
                "CapacityProviderStrategy": [
                    {
                        "Base": integer,
                        "CapacityProvider": "capacityProviderA",
                        "Weight": integer
                    },
                    {
                        "Base": integer,
                        "CapacityProvider": "capacityProviderB",
                        "Weight": integer
                    }
                ]
```

```
            }
        }
    }
]
```

각 속성은 숫자인 ContainerPort을(를) 제외하고 문자열로 지정됩니다.

- TaskDefinition - 필수. 배포할 Amazon ECS 서비스의 작업 정의입니다. 작업 정의의 ARN으로 지정됩니다. ARN 형식은 arn:aws:ecs:*aws-region*:*account-id*:task-definition/*task-definition-family*:*task-definition-revision*입니다. 자세한 내용은 Amazon 리소스 이름(ARNs) 및 AWS 서비스 네임스페이스를 참조하세요.

  > **Note**
  >
  > ARN의 :*task-definition-revision* 부분은 선택 사항입니다. 생략할 경우 Amazon ECS는 작업 정의의 최신 ACTIVE 개정 버전을 사용합니다.

- ContainerName - 필수. Amazon ECS 애플리케이션이 포함된 Amazon ECS 컨테이너의 이름입니다. Amazon ECS 작업 정의에 지정된 컨테이너여야 합니다.
- ContainerPort - 필수. 트래픽이 라우팅되는 컨테이너의 포트입니다.
- PlatformVersion: 선택 사항입니다. 배포된 Amazon ECS 서비스의 Fargate 작업의 플랫폼 버전입니다. 자세한 내용은 AWS Fargate 플랫폼 버전을 참조하세요. 지정하지 않으면 기본적으로 LATEST이(가) 사용됩니다.
- NetworkConfiguration: 선택 사항입니다. AwsvpcConfiguration에서 다음을 지정할 수 있습니다. 자세한 내용은 Amazon ECS Container Service API 참조의 AwsVpcConfiguration을 참조하세요.
  - Subnets: 선택 사항입니다. Amazon ECS 서비스의 쉼표로 구분된 하나 이상의 서브넷 목록.
  - SecurityGroups: 선택 사항입니다. Amazon Elastic Container Service의 쉼표로 구분된 하나 이상의 보안 그룹 목록.
  - AssignPublicIp: 선택 사항입니다. Amazon ECS 서비스의 탄력적 네트워크 인터페이스가 퍼블릭 IP 주소를 수신하는지 여부를 지정하는 문자열. 유효 값은 ENABLED 및 DISABLED입니다.

  > **Note**
  >
  > NetworkConfiguration 아래의 설정을 모두 지정하거나 아무 것도 지정하지 않아야 합니다. 예를 들어, Subnets를 지정하려는 경우 SecurityGroups 및 AssignPublicIp도 지

정해야 합니다. 아무 것도 지정하지 않는 경우 CodeDeploy는 현재 네트워크 Amazon ECS 설정을 사용합니다.

- `CapacityProviderStrategy`: 선택 사항입니다. 배포에 사용할 Amazon ECS 용량 공급자 목록입니다. 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 Amazon ECS 용량 공급자를 참조하세요. 각 용량 공급자에 대해 다음 설정을 지정할 수 있습니다. 이러한 설정에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::ECS::ServiceCapacityProviderStrategyItem을 참조하세요.
  - `Base`: 선택 사항입니다. 최소한 기준 값은 지정된 용량 공급자에서 실행할 태스크 수를 지정합니다. 용량 공급자 전략에서 하나의 용량 공급자만 기준을 정의할 수 있습니다. 값을 지정하지 않으면 기본값 `0`이 사용됩니다.
  - `CapacityProvider`: 선택 사항입니다. 용량 공급자의 짧은 이름입니다. 예: capacityProviderA
  - `Weight`: 선택 사항입니다.

    가중치 값은 지정된 용량 공급자를 사용해야 하는 시작된 총 작업 수의 상대 백분율을 지정합니다. `base` 값이 정의되어 있다면 이 값이 만족된 다음 `weight` 값을 고려합니다.

    `weight` 값을 지정하지 않으면 `0`의 기본값이 사용됩니다. 용량 공급자 전략 내에서 여러 용량 공급자가 지정된 경우 하나 이상의 용량 공급자가 0보다 큰 가중치 값을 가져야 하며 가중치가 `0`인 모든 용량 공급자는 태스크를 배치하는 데 사용되지 않습니다. 가중치가 모두 `0`인 전략에 여러 용량 공급자를 지정하면 용량 공급자 전략을 사용하는 모든 `RunTask` 또는 `CreateService` 작업이 실패합니다.

    예를 들면 둘 다 1의 가중치를 갖는 경우 `base`가 충족되면 작업이 두 용량 공급자에 균등하게 분할되는 두 개의 용량 공급자를 포함하는 전략을 정의하는 가중치를 사용하는 시나리오입니다. 동일한 논리를 사용하여 capacityProviderA에 1의 가중치를 지정하고 capacityProviderB에 4의 가중치를 지정하면 capacityProviderA를 사용하여 실행되는 모든 태스크에 대해 네 가지 태스크에서 capacityProviderB를 사용합니다.

AppSpec 'permissions' 섹션(EC2/온프레미스 배포만 해당)

`'permissions'` 섹션은 `'files'` 섹션의 파일 및 디렉터리/폴더가 인스턴스에 복사된 후 이러한 파일 및 디렉터리/폴더에 특수 권한(있는 경우)이 어떻게 적용되어야 하는지를 지정합니다. 여러 개의 `object` 명령을 지정할 수 있습니다. 이 섹션은 선택 사항입니다. Amazon Linux, Ubuntu Server, RHEL 및 macOS 인스턴스에만 적용됩니다.

> **Note**
>
> macOS 인스턴스에서는 `acls` 및 `context` 지침이 지원되지 않습니다. AppSpec 파일이 macOS에서 둘 중 하나를 사용하는 배포는 파일이 설치되기 전에 실패합니다. 다른 지침은이 단원에 설명된 대로 작동합니다.

> **Note**
>
> EC2/온프레미스 배포용으로만 `'permissions'` 섹션을 사용합니다. AWS Lambda 또는 Amazon ECS 배포에는 사용되지 않습니다.

이 섹션의 구조는 다음과 같습니다.

```
permissions:
  - object: object-specification
    pattern: pattern-specification
    except: exception-specification
    owner: owner-account-name
    group: group-name
    mode: mode-specification
    acls:
      - acls-specification
    context:
      user: user-specification
      type: type-specification
      range: range-specification
    type:
      - object-type
```

명령은 다음과 같습니다.

- `object` - 필수. 파일 시스템 객체가 인스턴스로 복사된 후 지정한 권한이 적용되는 파일 시스템 객체 세트(파일 또는 디렉터리/폴더)입니다.

  문자열을 사용하여 `object`를 지정합니다.
- `pattern` – 선택 사항. 권한을 적용할 패턴을 지정합니다. 지정하지 않거나 특수 문자 `"**"`를 사용하여 지정하면 권한이 `type`에 따라 일치하는 모든 파일 또는 디렉터리에 적용됩니다.

따옴표("")가 있는 문자열을 사용하여 pattern을 지정합니다.

- except – 선택 사항. pattern에 대한 예외인 파일 또는 디렉터리를 지정합니다.

  대괄호 안에 있는 쉼표로 구분된 문자열 목록을 사용하여 except를 지정합니다.

- owner – 선택 사항. object의 소유자 이름입니다. 지정하지 않으면 원본 파일 또는 디렉터리/폴더 구조에 적용된 기존의 모든 소유자가 복사 작업 후에도 아무 것도 변경되지 않습니다.

  문자열을 사용하여 owner를 지정합니다.

- group – 선택 사항. object의 그룹 이름입니다. 지정하지 않으면 원본 파일 또는 디렉터리/폴더 구조에 적용된 기존의 모든 그룹이 복사 작업 후에도 아무 것도 변경되지 않습니다.

  문자열을 사용하여 group를 지정합니다.

- mode – 선택 사항. object에 적용할 권한을 지정하는 숫자 값. 모드 설정은 Linux chmod 명령 구문을 따릅니다.

> **Important**
>
> 값이 0으로 시작하는 경우 큰 따옴표로 묶거나 세 자리 숫자만 남도록 선행하는 0을 제거해야 합니다.

> **Note**
>
> **u+x**와(과) 같은 기호 표기법은 mode 설정에 대해 지원되지 않습니다.

예제:

- mode: "0644"은(는) 개체 소유자에게 읽기 및 쓰기 권한(6), 그룹에 읽기 전용 권한(4) 및 다른 모든 사용자에게 읽기 전용 권한(4)을 부여합니다
- mode: 644은(는) mode: "0644"와(과) 동일한 권한을 부여합니다.
- mode: 4755은(는) setuid 특성(4)을 설정하고 소유자에게 모든 제어 권한(7)을 부여하며 그룹에 읽기 및 실행 권한(5), 다른 모든 사용자에게 읽기 및 실행 권한(5)을 부여합니다.

  더 많은 예제를 보려면 Linux chmod 명령 설명서를 참조하세요.

모드를 지정하지 않으면 원본 파일 또는 폴더 구조에 적용된 기존의 모든 모드가 복사 작업 후에도 아무 것도 변경되지 않습니다.

- `acls` – 선택 사항. `object`에 적용된 하나 이상의 ACL(액세스 제어 목록) 항목을 나타내는 문자열 목록. 예를 들어, **`u:bob:rw`**는 사용자 **`bob`**에 대해 읽기 및 쓰기 권한을 나타냅니다. (더 많은 예제를 보려면 Linux `setfacl` 명령 설명서에서 ACL 항목 형식의 예를 참조하세요.) ACL 항목은 여러 개 지정할 수 있습니다. `acls`를 지정하지 않으면 원본 파일 또는 디렉터리/폴더 구조에 적용된 모든 기존 ACL이 복사 작업 후에 변경되지 않고 그대로 유지됩니다. 기존 ACL을 대체합니다.

  대시(-), 그 다음에 공백, 그 다음에 문자열을 사용하여 `acls`를 지정합니다(예: `- u:jane:rw`). 2개 이상의 ACL이 있는 경우 각각 별도의 줄에서 지정됩니다.

  > **Note**
  >
  > 이름이 없는 사용자, 이름이 없는 그룹 또는 기타 유사한 ACL 항목을 설정하면 AppSpec 파일이 실패합니다. 이러한 유형의 권한을 지정하려면 `mode`를 대신 사용하세요.

- `context` – 선택 사항. 보안 강화 Linux(SELinux) 기반 인스턴스의 경우 복사된 객체에 적용되는 보안 관련 컨텍스트 레이블 목록. 레이블은 `user`, `type` 및 `range`가 포함된 키로 지정됩니다. 자세한 내용은 SELinux 설명서를 참조하세요. 각 키는 문자열로 입력됩니다. 지정하지 않으면 원본 파일 또는 디렉터리/폴더 구조에 적용된 기존의 모든 레이블이 복사 작업 후에도 아무 것도 변경되지 않습니다.
  - `user` – 선택 사항. SELinux 사용자
  - `type` – 선택 사항입니다. SELinux 유형 이름.
  - `range` – 선택 사항입니다. SELinux 범위 지정자. 머신에서 MLS(다중 수준 보안) 및 MCS(다중 범주 보안)를 활성화하지 않으면 아무 영향이 없습니다. 활성화하지 않으면 `range`는 기본적으로 **`s0`**으로 지정됩니다.

  문자열을 사용하여 `context`를 지정합니다(예: `user: unconfined_u`). 각 `context`는 별도의 줄에서 지정됩니다.

- `type` – 선택 사항입니다. 지정한 권한을 적용할 객체 유형입니다. `type`은 **`file`** 또는 **`directory`**로 설정될 수 있는 문자열입니다. **`file`**을 지정하면 복사 작업 후 `object`에 즉시 포함되는 파일에만 권한이 적용됩니다(`object` 자체에는 적용되지 않음). **`directory`**를 지정하면 복사 작업 후 `object` 안의 어떤 위치에든 있는 모든 디렉터리/폴더에 권한이 반복적으로 적용됩니다(`object` 자체에는 적용되지 않음).

  대시(-), 그 다음에 공백, 그 다음에 문자열을 사용하여 `type`을 지정합니다(예: `- file`).

'permissions' 섹션의 예

다음 예제에서는 `object`, `pattern`, `except`, `owner`, `mode` 및 `type` 명령을 사용하여 `'permissions'` 섹션을 지정하는 방법을 보여 줍니다. 이 예제는 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에만 적용됩니다. 이 예에서는 다음 파일 및 폴더가 아래와 같은 계층 구조로 복사된다고 가정합니다.

```
/tmp
  `-- my-app
       |-- my-file-1.txt
       |-- my-file-2.txt
       |-- my-file-3.txt
       |-- my-folder-1
       |     |-- my-file-4.txt
       |     |-- my-file-5.txt
       |     `-- my-file-6.txt
       `-- my-folder-2
             |-- my-file-7.txt
             |-- my-file-8.txt
             |-- my-file-9.txt
            `-- my-folder-3
```

다음 AppSpec 파일은 파일 및 폴더가 복사된 후 이러한 파일 및 폴더에 대한 권한을 설정하는 방법을 보여줍니다.

```
version: 0.0
os: linux
# Copy over all of the folders and files with the permissions they
#  were originally assigned.
files:
  - source: ./my-file-1.txt
    destination: /tmp/my-app
  - source: ./my-file-2.txt
    destination: /tmp/my-app
  - source: ./my-file-3.txt
    destination: /tmp/my-app
  - source: ./my-folder-1
    destination: /tmp/my-app/my-folder-1
  - source: ./my-folder-2
    destination: /tmp/my-app/my-folder-2
# 1) For all of the files in the /tmp/my-app folder ending in -3.txt
#  (for example, just my-file-3.txt), owner = adm, group = wheel, and
```

```
#  mode = 464 (-r--rw-r--).
permissions:
  - object: /tmp/my-app
    pattern: "*-3.txt"
    owner: adm
    group: wheel
    mode: 464
    type:
      - file
# 2) For all of the files ending in .txt in the /tmp/my-app
#  folder, but not for the file my-file-3.txt (for example,
#  just my-file-1.txt and my-file-2.txt),
#  owner = ec2-user and mode = 444 (-r--r--r--).
  - object: /tmp/my-app
    pattern: "*.txt"
    except: [my-file-3.txt]
    owner: ec2-user
    mode: 444
    type:
      - file
# 3) For all the files in the /tmp/my-app/my-folder-1 folder except
#  for my-file-4.txt and my-file-5.txt, (for example,
#  just my-file-6.txt), owner = operator and mode = 646 (-rw-r--rw-).
  - object: /tmp/my-app/my-folder-1
    pattern: "**"
    except: [my-file-4.txt, my-file-5.txt]
    owner: operator
    mode: 646
    type:
      - file
# 4) For all of the files that are immediately under
#  the /tmp/my-app/my-folder-2 folder except for my-file-8.txt,
#  (for example, just my-file-7.txt and
#  my-file-9.txt), owner = ec2-user and mode = 777 (-rwxrwxrwx).
  - object: /tmp/my-app/my-folder-2
    pattern: "**"
    except: [my-file-8.txt]
    owner: ec2-user
    mode: 777
    type:
      - file
# 5) For all folders at any level under /tmp/my-app that contain
#  the name my-folder but not
#  /tmp/my-app/my-folder-2/my-folder-3 (for example, just
```

```
#  /tmp/my-app/my-folder-1 and /tmp/my-app/my-folder-2),
#  owner = ec2-user and mode = 555 (dr-xr-xr-x).
  - object: /tmp/my-app
    pattern: "*my-folder*"
    except: [tmp/my-app/my-folder-2/my-folder-3]
    owner: ec2-user
    mode: 555
    type:
      - directory
# 6) For the folder /tmp/my-app/my-folder-2/my-folder-3,
#  group = wheel and mode = 564 (dr-xrw-r--).
  - object: /tmp/my-app/my-folder-2/my-folder-3
    group: wheel
    mode: 564
    type:
      - directory
```

그 결과, 권한은 다음과 같습니다.

```
-r--r--r-- ec2-user root  my-file-1.txt
-r--r--r-- ec2-user root  my-file-2.txt
-r--rw-r-- adm      wheel my-file-3.txt

dr-xr-xr-x ec2-user root  my-folder-1
-rw-r--r-- root     root  my-file-4.txt
-rw-r--r-- root     root  my-file-5.txt
-rw-r--rw- operator root  my-file-6.txt

dr-xr-xr-x ec2-user root  my-folder-2
-rwxrwxrwx ec2-user root  my-file-7.txt
-rw-r--r-- root     root  my-file-8.txt
-rwxrwxrwx ec2-user root  my-file-9.txt

dr-xrw-r-- root     wheel my-folder-3
```

다음 예제에서는 `acls` 및 `context` 명령을 추가로 사용하여 `'permissions'` 섹션을 지정하는 방법을 보여 줍니다. 이 예제는 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에만 적용됩니다.

```
permissions:
  - object: /var/www/html/WordPress
    pattern: "**"
    except: [/var/www/html/WordPress/ReadMe.txt]
    owner: bob
```

```
    group: writers
    mode: 644
    acls:
      - u:mary:rw
      - u:sam:rw
      - m::rw
    context:
      user: unconfined_u
      type: httpd_sys_content_t
      range: s0
    type:
      - file
```

AppSpec 'hooks' 섹션

AppSpec 파일의 'hooks' 섹션 내용은 해당 배포의 컴퓨팅 플랫폼에 따라 다릅니다. EC2/온프레미스 배포에 대한 'hooks' 섹션에는 배포 수명 주기 이벤트 후크를 하나 이상의 스크립트에 연결하는 매핑이 포함되어 있습니다. Lambda 또는 Amazon ECS 배포에 대한 'hooks' 섹션은 배포 수명 주기 이벤트 중 실행하는 Lambda 확인 함수를 지정합니다. 이벤트 후크가 없는 경우 해당 이벤트에 대해 작업이 실행되지 않습니다. 이 섹션은 배포의 일부로 스크립트 또는 Lambda 확인 함수를 실행하는 경우에만 필요합니다.

주제

- Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션
- AWS Lambda 배포를 위한 AppSpec 'hooks' 섹션
- EC2/온프레미스 배포를 위한 AppSpec 'hooks' 섹션

Amazon ECS 배포를 위한 AppSpec 'hooks' 섹션

주제

- Amazon ECS 배포를 위한 수명 주기 이벤트 후크 목록
- Amazon ECS 배포 시 후크의 실행 순서입니다.
- 'hooks' 섹션의 구조
- 샘플 Lambda 'hooks' 함수

Amazon ECS 배포를 위한 수명 주기 이벤트 후크 목록

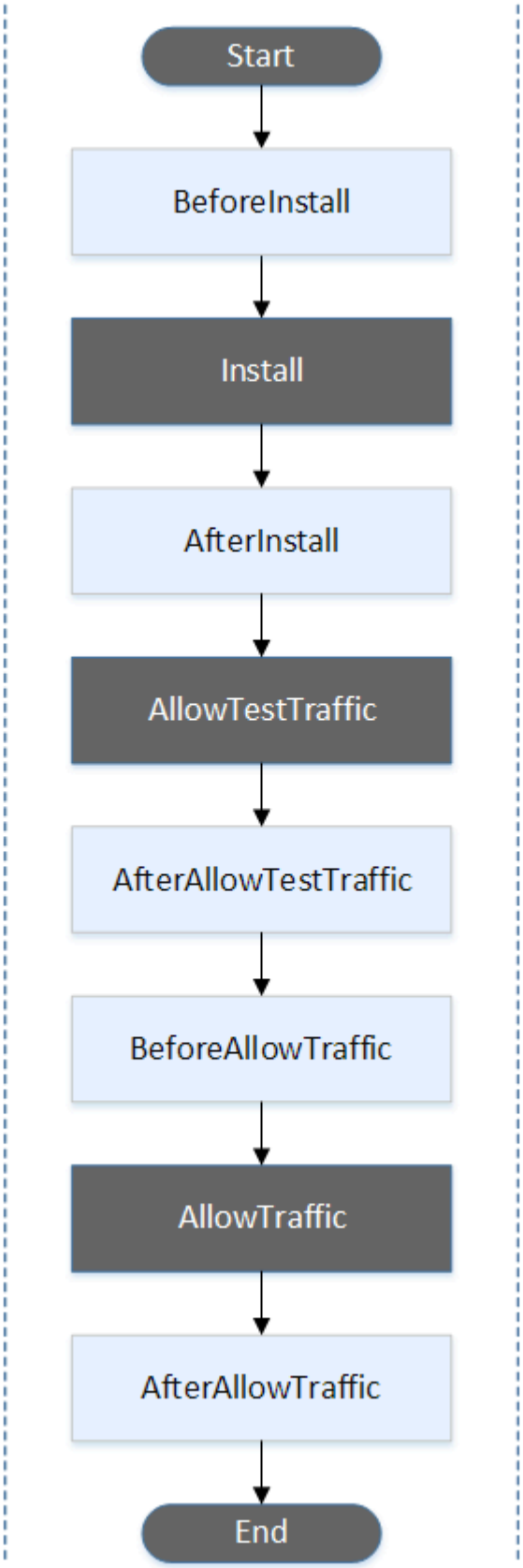
AWS Lambda 후크는 수명 주기 이벤트 이름 뒤의 새 줄에 문자열로 지정된 하나의 Lambda 함수입니다. 각 후크는 배포별로 한 번 실행됩니다. 다음은 Amazon ECS 배포 중에 후크를 실행할 수 있는 수명 주기 이벤트에 대한 설명입니다.

- `BeforeInstall` – 대체 작업 세트가 생성되기 전에 작업을 실행하려면 이 항목을 사용합니다. 대상 그룹 하나가 원래 작업 세트와 연결됩니다. 테스트 리스너(선택 사항)가 지정된 경우 원래 작업 세트와 연결됩니다. 이 시점에서는 롤백이 불가능합니다.
- `AfterInstall` – 대체 작업 세트가 생성되고 대상 그룹 중 하나가 연결된 후 작업을 실행하면 이 항목을 사용합니다. 테스트 리스너(선택 사항)가 지정된 경우 원래 작업 세트와 연결됩니다. 이 수명 주기 이벤트에서 후크 함수의 결과는 롤백을 트리거할 수 있습니다.
- `AfterAllowTestTraffic` – 테스트 리스너가 대체 작업 세트에 트래픽을 제공한 후 작업을 실행하려면 이 항목을 사용합니다. 이 시점에서 후크 함수의 결과는 롤백을 트리거할 수 있습니다.
- `BeforeAllowTraffic` – 두 번째 대상 그룹이 대체 작업 세트와 연결된 후 트래픽이 대체 작업 세트로 전환되기 전에 작업을 실행하려면 이 항목을 사용합니다. 이 수명 주기 이벤트에서 후크 함수의 결과는 롤백을 트리거할 수 있습니다.
- `AfterAllowTraffic` – 두 번째 대상 그룹이 대체 작업 세트에 트래픽을 제공한 후 작업을 실행하려면 이 항목을 사용합니다. 이 수명 주기 이벤트에서 후크 함수의 결과는 롤백을 트리거할 수 있습니다.

자세한 내용은 Amazon ECS 배포 중에 발생하는 일 및 튜토리얼: Amazon ECS 서비스 배포 및 확인 테스트 섹션을 참조하세요.

Amazon ECS 배포 시 후크의 실행 순서입니다.

Amazon ECS 배포에서 이벤트 후크는 다음 순서대로 실행됩니다.


Start
BeforeInstall
Install
AfterInstall
AllowTestTraffic
AfterAllowTestTraffic
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **Note**
>
> 배포의 Start, Install, TestTraffic, AllowTraffic 및 End 이벤트는 스크립팅할 수 없기 때문에 이 다이어그램에서 회색으로 표시됩니다.

'hooks' 섹션의 구조

다음은 'hooks' 섹션 구조의 예입니다.

YAML 사용:

```
Hooks:
  - BeforeInstall: "BeforeInstallHookFunctionName"
  - AfterInstall: "AfterInstallHookFunctionName"
  - AfterAllowTestTraffic: "AfterAllowTestTrafficHookFunctionName"
  - BeforeAllowTraffic: "BeforeAllowTrafficHookFunctionName"
  - AfterAllowTraffic: "AfterAllowTrafficHookFunctionName"
```

JSON 사용:

```
"Hooks": [
  {
   "BeforeInstall": "BeforeInstallHookFunctionName"
  },
  {
   "AfterInstall": "AfterInstallHookFunctionName"
  },
  {
   "AfterAllowTestTraffic": "AfterAllowTestTrafficHookFunctionName"
  },
  {
   "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
  },
  {
   "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
  }
 ]
}
```

샘플 Lambda 'hooks' 함수

'hooks' 섹션을 사용하여 CodeDeploy가 Amazon ECS 배포를 확인하기 위해 호출할 수 있는 Lambda 함수를 지정합니다. BeforeInstall, AfterInstall, AfterAllowTestTraffic, BeforeAllowTraffic 및 AfterAllowTraffic 배포 수명 주기 이벤트에 동일한 함수 또는 다른 함수를 사용할 수 있습니다. 검증 테스트가 완료되면 Lambda AfterAllowTraffic 함수는 CodeDeploy를 다시 호출하고 Succeeded 또는 Failed 결과를 전달합니다.

> **Important**
>
> CodeDeploy가 1시간 이내에 Lambda 확인 함수를 통해 알림을 받지 못하면 배포가 실패한 것으로 간주합니다.

Lambda 후크 함수 호출 전에 서버는 putLifecycleEventHookExecutionStatus 명령을 사용하여 배포 ID와 수명 주기 이벤트 후크 실행 ID를 통지받아야 합니다.

다음은 Node.js로 작성된 샘플 Lambda 후크 함수입니다.

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;

    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    /*
     Enter validation tests here.
    */

    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
```

```
    };

    // Pass CodeDeploy the prepared validation test results.
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
            callback(null, 'Validation test succeeded');
        }
    });
};
```

AWS Lambda 배포를 위한 AppSpec 'hooks' 섹션

주제

- AWS Lambda 배포를 위한 수명 주기 이벤트 후크 목록
- Lambda 함수 버전 배포에서 후크 실행 순서
- 'hooks' 섹션의 구조
- 샘플 Lambda 'hooks' 함수

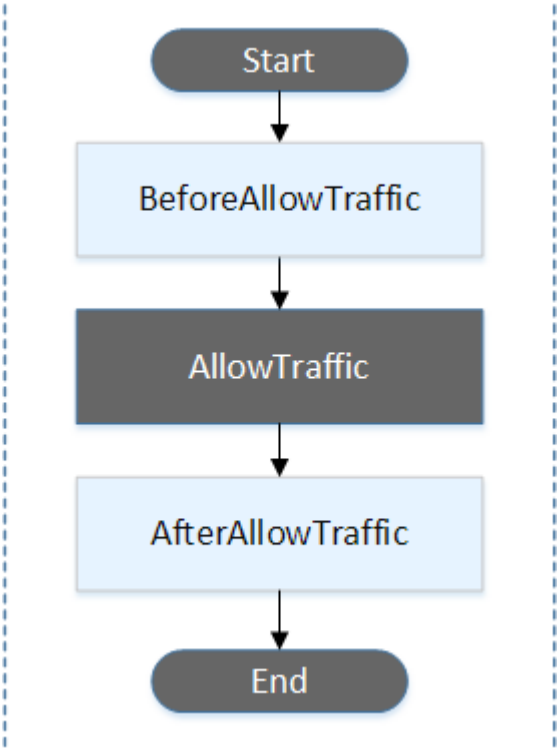
AWS Lambda 배포를 위한 수명 주기 이벤트 후크 목록

AWS Lambda 후크는 수명 주기 이벤트 이름 뒤의 새 줄에 문자열로 지정된 하나의 Lambda 함수입니다. 각 후크는 배포별로 한 번 실행됩니다. 다음은 AppSpec 파일에서 사용할 수 있는 후크에 대한 설명입니다.

- BeforeAllowTraffic – 배포된 Lambda 함수 버전으로 트래픽을 전환하기 전에 작업을 실행하려면 이 항목을 사용합니다.
- AfterAllowTraffic – 배포된 Lambda 함수 버전으로 모든 트래픽이 전환된 후에 작업을 실행하려면 이 항목을 사용합니다.

Lambda 함수 버전 배포에서 후크 실행 순서

서버리스 Lambda 함수 버전 배포에서 이벤트 후크는 다음 순서로 실행됩니다.



> **Note**
>
> 배포의 Start, AllowTraffic, End 이벤트는 스크립팅할 수 없기 때문에 이 다이어그램에서 회색으로 표시됩니다.

'hooks' 섹션의 구조

다음은 'hooks' 섹션 구조의 예입니다.

YAML 사용:

```
hooks:
   - BeforeAllowTraffic: BeforeAllowTrafficHookFunctionName
   - AfterAllowTraffic: AfterAllowTrafficHookFunctionName
```

JSON 사용:

```
"hooks": [{
    "BeforeAllowTraffic": "BeforeAllowTrafficHookFunctionName"
    },
    {
```

```
    "AfterAllowTraffic": "AfterAllowTrafficHookFunctionName"
}]
```

샘플 Lambda 'hooks' 함수

'hooks' 섹션을 사용하여 CodeDeploy가 Lambda 배포를 확인하기 위해 호출할 수 있는 Lambda 함수를 지정합니다. `BeforeAllowTraffic` 및 `AfterAllowTraffic` 배포 수명 주기 이벤트에 동일한 함수 또는 다른 함수를 사용할 수 있습니다. 검증 테스트가 완료되면 Lambda 확인 함수는 CodeDeploy를 다시 호출하고 `Succeeded` 또는 `Failed` 결과를 전달합니다.

> **Important**
>
> CodeDeploy가 1시간 이내에 Lambda 확인 함수를 통해 알림을 받지 못하면 배포가 실패한 것으로 간주합니다.

Lambda 후크 함수 호출 전에 서버는 `putLifecycleEventHookExecutionStatus` 명령을 사용하여 배포 ID와 수명 주기 이벤트 후크 실행 ID를 통지받아야 합니다.

다음은 Node.js로 작성된 샘플 Lambda 후크 함수입니다.

```
'use strict';

const aws = require('aws-sdk');
const codedeploy = new aws.CodeDeploy({apiVersion: '2014-10-06'});

exports.handler = (event, context, callback) => {
    //Read the DeploymentId from the event payload.
    var deploymentId = event.DeploymentId;

    //Read the LifecycleEventHookExecutionId from the event payload
    var lifecycleEventHookExecutionId = event.LifecycleEventHookExecutionId;

    /*
     Enter validation tests here.
    */

    // Prepare the validation test results with the deploymentId and
    // the lifecycleEventHookExecutionId for CodeDeploy.
    var params = {
        deploymentId: deploymentId,
```

```
        lifecycleEventHookExecutionId: lifecycleEventHookExecutionId,
        status: 'Succeeded' // status can be 'Succeeded' or 'Failed'
    };

    // Pass CodeDeploy the prepared validation test results.
    codedeploy.putLifecycleEventHookExecutionStatus(params, function(err, data) {
        if (err) {
            // Validation failed.
            callback('Validation test failed');
        } else {
            // Validation succeeded.
            callback(null, 'Validation test succeeded');
        }
    });
};
```

EC2/온프레미스 배포를 위한 AppSpec 'hooks' 섹션

주제

- 수명 주기 이벤트 후크 목록
- 수명 주기 이벤트 후크 가용성
- 배포 시 후크의 실행 순서
- 'hooks' 섹션의 구조
- 후크 스크립트에서 파일 참조
- 후크의 환경 변수 가용성
- 후크 예제

수명 주기 이벤트 후크 목록

EC2/온프레미스 배포 후크는 인스턴스에 대한 배포별로 한 번 실행됩니다. 후크에서 실행할 하나 이상의 스크립트를 지정할 수 있습니다. 수명 주기 이벤트에 대한 각 후크는 별도의 줄에서 문자열로 지정됩니다. 다음은 AppSpec 파일에서 사용할 수 있는 후크에 대한 설명입니다.

어떤 수명 주기 이벤트 후크가 어떤 배포 및 롤백 유형에 유효한지는 수명 주기 이벤트 후크 가용성 단원을 참조하세요.

- `ApplicationStop` – 이 배포 수명 주기 이벤트는 애플리케이션 수정이 다운로드되기 전에도 발생합니다. 이 이벤트에 대해서는 애플리케이션을 안전하게 종료하거나 배포 준비 중에 현재 설치된 패

키지를 제거하도록 스크립트를 지정할 수 있습니다. 이 배포 수명 주기 이벤트에 사용되는 AppSpec 파일 및 스크립트는 이전에 성공적으로 배포된 애플리케이션 수정에서 가져옵니다.

> **Note**
>
> AppSpec 파일은 인스턴스에 배포하기 전에는 인스턴스에 존재하지 않습니다. 이러한 이유로, 인스턴스에 처음으로 배포할 때는 `ApplicationStop` 후크가 실행되지 않습니다. 인스턴스에 두 번째로 배포할 때는 `ApplicationStop` 후크를 사용할 수 있습니다.

가장 마지막으로 배포가 완료된 애플리케이션 수정의 위치를 확인하기 위해 CodeDeploy 에이전트는 *deployment-group-id*`_last_successful_install` 파일에 나와 있는 위치를 찾습니다. 이 파일의 위치는 다음과 같습니다.

Amazon Linux, Ubuntu Server 및 RHEL Amazon EC2의 `/opt/codedeploy-agent/deployment-root/deployment-instructions` 폴더

Windows Server Amazon EC2 인스턴스에 대한 `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions` 폴더.

`ApplicationStop` 배포 수명 주기 이벤트 중 실패하는 배포 문제를 해결하려면 실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결 단원을 참조하세요.

- `DownloadBundle` – 이 배포 수명 주기 이벤트 중 에이전트는 애플리케이션 수정 파일을 다음 임시 위치로 복사합니다.

  Amazon Linux, Ubuntu Server 및 RHEL Amazon EC2의 `/opt/codedeploy-agent/deployment-root/`*deployment-group-id*`/`*deployment-id*`/deployment-archive` 폴더

  Windows Server Amazon EC2 인스턴스에 대한 `C:\ProgramData\Amazon\CodeDeploy\`*deployment-group-id*`\`*deployment-id*`\deployment-archive` 폴더.

  이 이벤트는 CodeDeploy 에이전트에 예약되어 있으므로 스크립트 실행에 사용할 수 없습니다.

  `DownloadBundle` 배포 수명 주기 이벤트 중 실패하는 배포 문제를 해결하려면 실패하고 UnknownError: not opened for reading이 표시되는 DownloadBundle 배포 수명 주기 이벤트 문제 해결 단원을 참조하세요.
- `BeforeInstall` – 파일 암호화 해제 및 현재 버전의 백업 만들기와 같은 사전 설치 작업에 이 배포 수명 주기 이벤트를 사용할 수 있습니다.

- `Install` – 이 배포 수명 주기 이벤트 중에 CodeDeploy 에이전트는 수정 파일을 임시 위치에서 최종 대상 폴더로 복사합니다. 이 이벤트는 CodeDeploy 에이전트에 예약되어 있으므로 스크립트 실행에 사용할 수 없습니다.
- `AfterInstall` – 애플리케이션 구성 또는 파일 권한 변경과 같은 작업에 이 배포 수명 주기 이벤트를 사용할 수 있습니다.
- `ApplicationStart` - `ApplicationStop` 중에 중지된 서비스를 다시 시작하려면 일반적으로 이 배포 수명 주기 이벤트를 사용합니다.
- `ValidateService` – 마지막 배포 수명 주기 이벤트입니다. 배포가 성공적으로 완료되었는지 확인하는 데 사용됩니다.
- `BeforeBlockTraffic` – 로드 밸런서에서 작업이 등록 취소되기 전에 인스턴스에서 작업을 실행하려면 이 배포 수명 주기 이벤트를 사용할 수 있습니다.

  `BeforeBlockTraffic` 배포 수명 주기 이벤트 중 실패하는 배포 문제를 해결하려면 실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결 단원을 참조하세요.
- `BlockTraffic` – 이 배포 수명 주기 이벤트 중에는 트래픽을 현재 제공하고 있는 인스턴스에 액세스할 수 없도록 인터넷 트래픽이 차단됩니다. 이 이벤트는 CodeDeploy 에이전트에 예약되어 있으므로 스크립트 실행에 사용할 수 없습니다.
- `AfterBlockTraffic` – 각 로드 밸런서에서 작업이 등록 취소된 후 인스턴스에서 작업을 실행하려면 이 배포 수명 주기 이벤트를 사용할 수 있습니다.

  `AfterBlockTraffic` 배포 수명 주기 이벤트 중 실패하는 배포 문제를 해결하려면 실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결 단원을 참조하세요.
- `BeforeAllowTraffic` – 로드 밸런서에 작업이 등록되기 전에 인스턴스에서 작업을 실행하려면 이 배포 수명 주기 이벤트를 사용할 수 있습니다.
- `AllowTraffic` – 이 배포 수명 주기 이벤트 중에는 배포 후 인터넷 트래픽이 인스턴스에 액세스할 수 있도록 허용됩니다. 이 이벤트는 CodeDeploy 에이전트에 예약되어 있으므로 스크립트 실행에 사용할 수 없습니다.
- `AfterAllowTraffic` – 로드 밸런서에 작업이 등록된 후 인스턴스에서 작업을 실행하려면 이 배포 수명 주기 이벤트를 사용할 수 있습니다.

수명 주기 이벤트 후크 가용성

다음 표에는 각 배포 및 롤백 시나리오에 사용할 수 있는 수명 주기 이벤트 후크가 나와 있습니다.

수명 주기 이벤트 이름	Auto Scaling 시작 배포[1]	Auto Scaling 종료 배포[1]	현재 위치 배포[1]	블루/그린 배포: 원본 인스턴스	블루/그린 배포: 대체 인스턴스	블루/그린 배포 롤백: 원본 인스턴스	블루/그린 배포 롤백: 대체 인스턴스
ApplicationStop	✓	✓	✓		✓		
DownloadBundle[3]	✓		✓		✓		
BeforeInstall	✓		✓		✓		
Install[3]	✓		✓		✓		
AfterInstall	✓		✓		✓		
ApplicationStart	✓		✓		✓		
ValidateService	✓		✓		✓		
BeforeBlockTraffic		✓	✓	✓			✓
BlockTraffic[3]		✓	✓	✓			✓
AfterBlockTraffic		✓	✓	✓			✓
BeforeAllowTraffic	✓		✓		✓	✓	
AllowTraffic[3]	✓		✓		✓	✓	

수명 주기 이벤트 이름	Auto Scaling 시작 배포 [1]	Auto Scaling 종료 배포 [1]	현재 위치 배포[1]	블루/그린 배포: 원본 인스턴스	블루/그린 배포: 대체 인스턴스	블루/그린 배포 롤백: 원본 인스턴스	블루/그린 배포 롤백: 대체 인스턴스
AfterAllowTraffic	✓		✓		✓	✓	

[1] Amazon EC2 Auto Scaling 배포에 대한 자세한 내용은 Amazon EC2 Auto Scaling에서 CodeDeploy를 사용하는 방식 단원을 참조하세요.

[2] 현재 위치 배포의 롤백에도 적용됩니다.

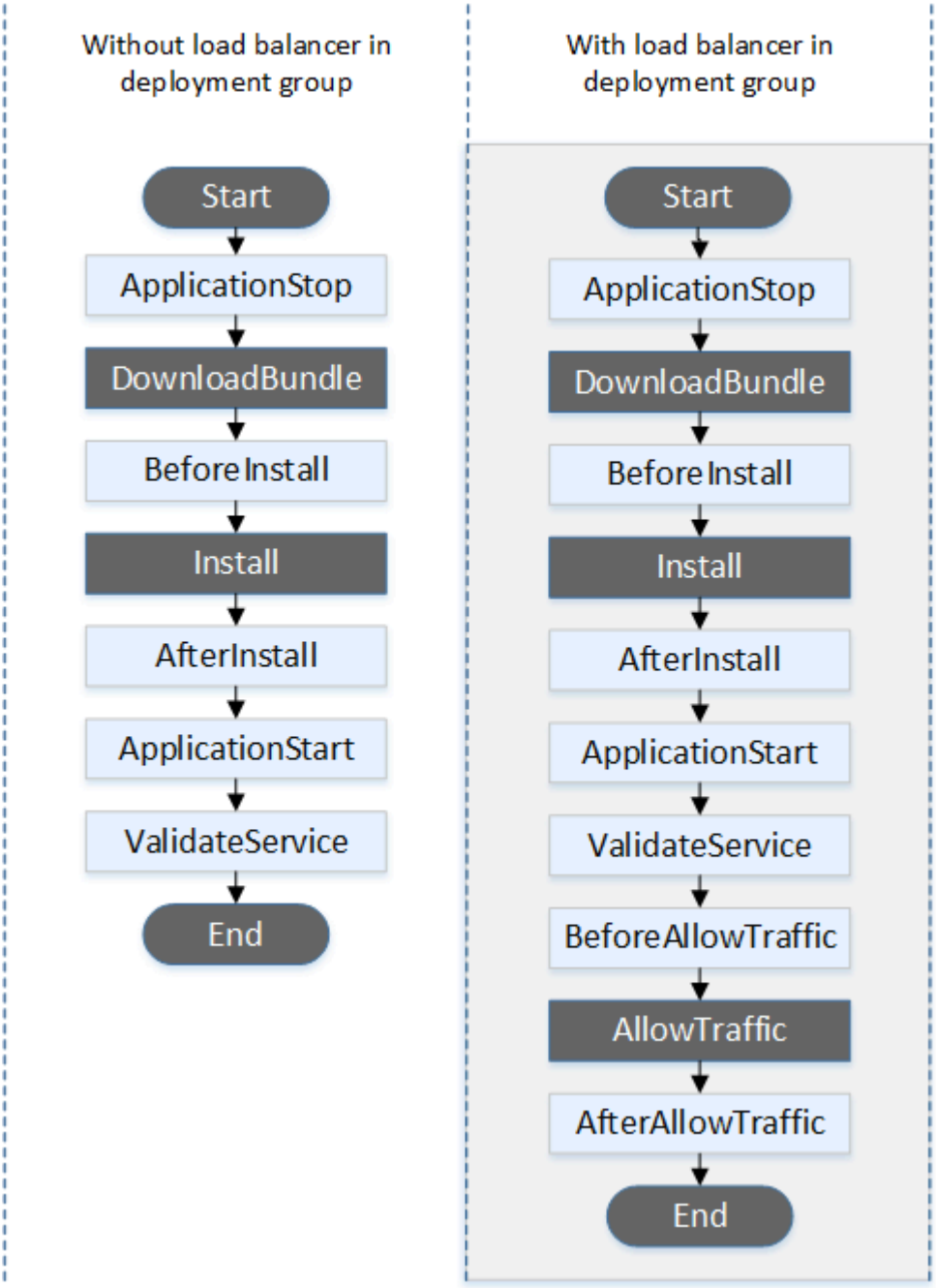
[3] CodeDeploy 작업을 위해 예약되어 있습니다. 스크립트를 실행하는 데 사용할 수 없습니다.

배포 시 후크의 실행 순서

Auto Scaling 시작 배포

Auto Scaling 시작 배포 중에 CodeDeploy는 다음 순서로 이벤트 후크를 실행합니다.

Amazon EC2 Auto Scaling 시작 배포에 대한 자세한 내용은 Amazon EC2 Auto Scaling에서 CodeDeploy를 사용하는 방식 단원을 참조하세요.

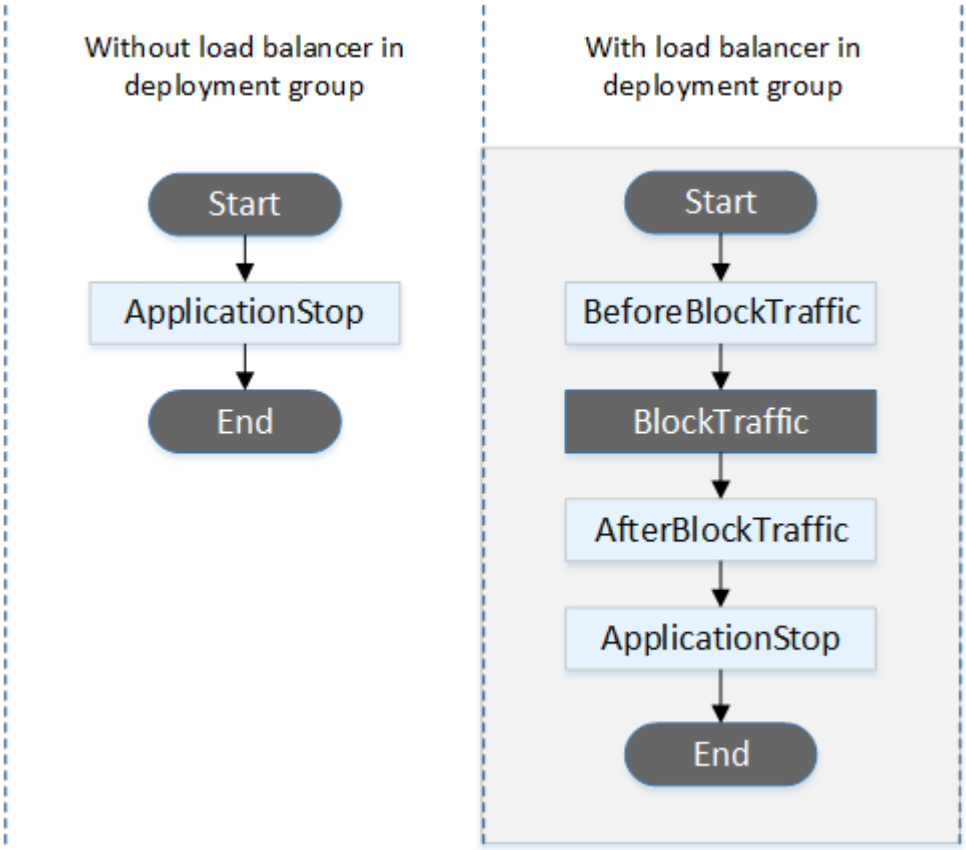


> **Note**
>
> 배포의 Start, Install, TestTraffic, AllowTraffic 및 End 이벤트는 스크립트화할 수 없으므로 이 다이어그램에서 회색으로 표시됩니다. 그러나 AppSpec 파일의 `'files'` 섹션을 편집하여 Install 이벤트 중 설치되는 항목을 지정할 수 있습니다.

Auto Scaling 종료 배포

Auto Scaling 종료 배포 중에 CodeDeploy는 다음 순서로 이벤트 후크를 실행합니다.

Amazon EC2 Auto Scaling 종료 배포에 대한 자세한 내용은 Auto Scaling 확장 이벤트 중 종료 배포 활성화 단원을 참조하세요.

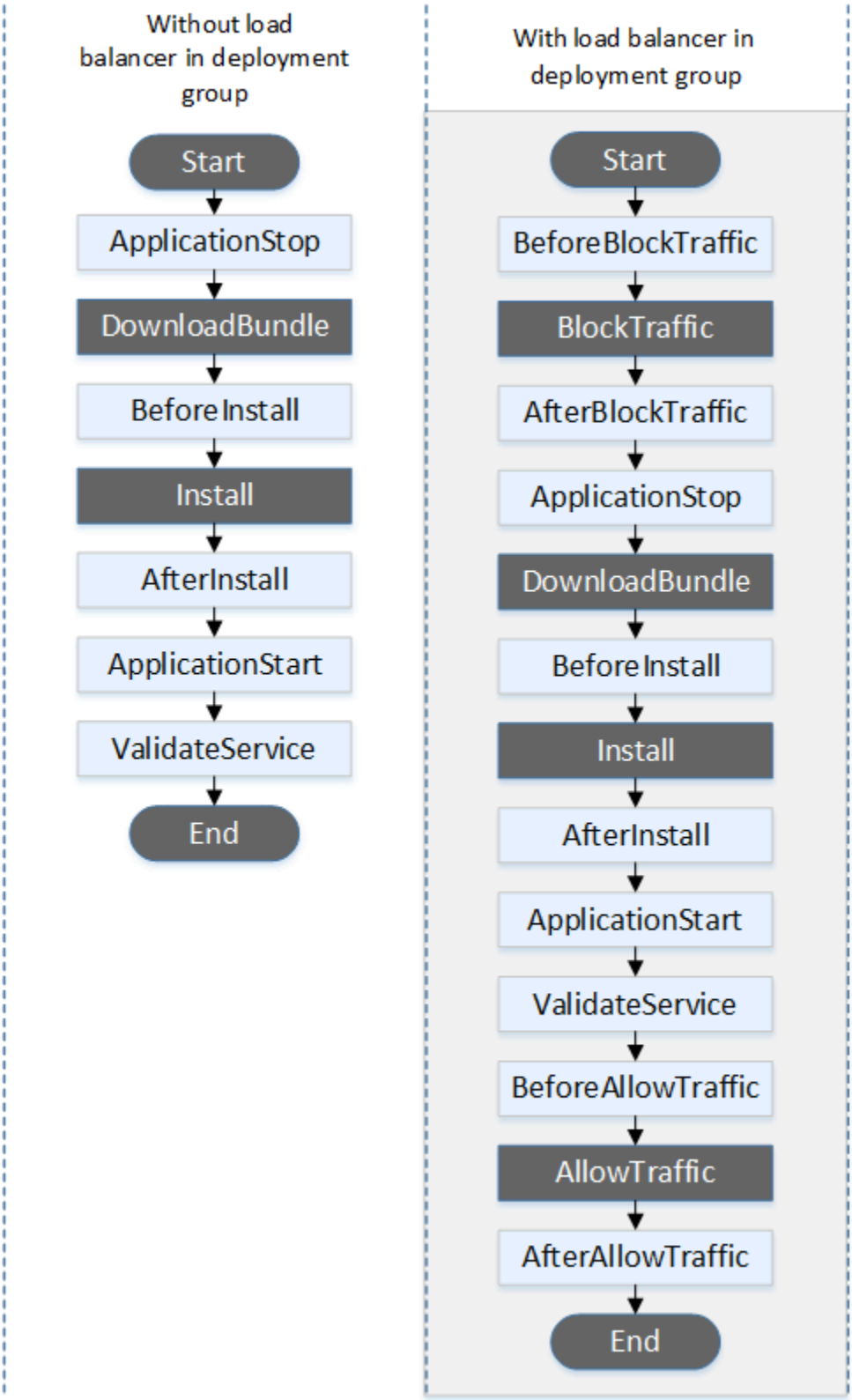


> **Note**
>
> 배포의 Start, BlockTraffic, End 이벤트는 스크립팅할 수 없기 때문에 이 다이어그램에서 회색으로 표시됩니다.

인 플레이스(in-place) 배포

인 플레이스(in-place) 배포 시(인 플레이스(in-place) 배포의 롤백 포함) 이벤트 후크는 다음 순서로 실행됩니다.

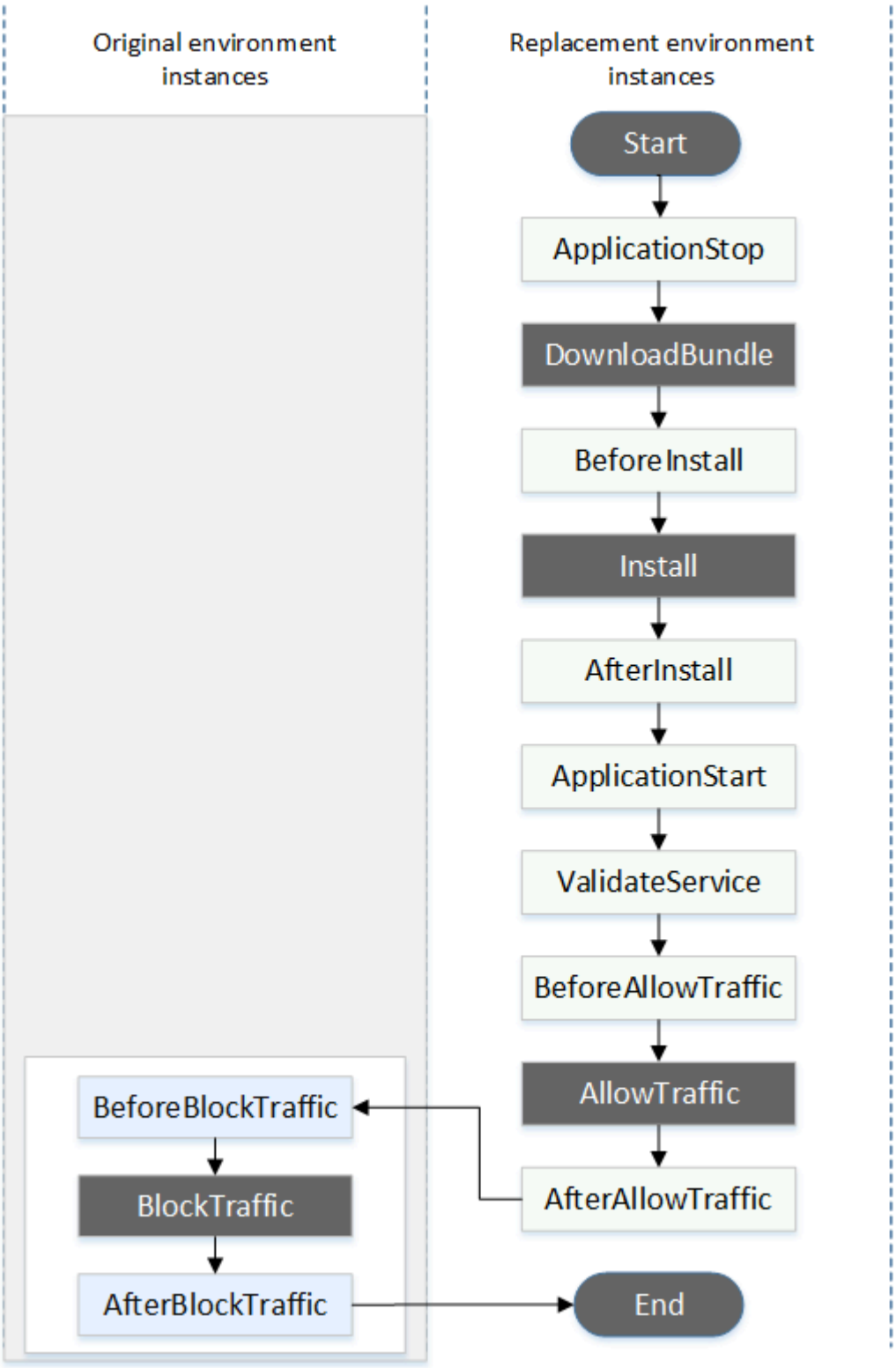
> **Note**
>
> 인 플레이스 배포의 경우 트래픽 차단 및 허용과 관련된 6개의 후크는 배포 그룹의 Elastic Load Balancing에서 Classic Load Balancer, Application Load Balancer 또는 Network Load Balancer를 지정한 경우에만 적용됩니다.


Without load balancer in deployment group
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
End
With load balancer in deployment group
Start
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
End

> **Note**
>
> 배포의 Start, DownloadBundle, Install 및 End 이벤트는 스크립팅할 수 없기 때문에 이 다이어그램에서 회색으로 표시됩니다. 그러나 AppSpec 파일의 `'files'` 섹션을 편집하여 Install 이벤트 중 설치되는 항목을 지정할 수 있습니다.

블루/그린 배포

블루/그림 배포에서 이벤트 후크는 다음 순서대로 실행됩니다.


Original environment instances
Replacement environment instances
Start
ApplicationStop
DownloadBundle
BeforeInstall
Install
AfterInstall
ApplicationStart
ValidateService
BeforeAllowTraffic
AllowTraffic
AfterAllowTraffic
BeforeBlockTraffic
BlockTraffic
AfterBlockTraffic
End

> **Note**
>
> 배포의 Start, DownloadBundle, Install, BlockTraffic, AllowTraffic 및 End 이벤트는 스크립팅할 수 없기 때문에 이 다이어그램에서 회색으로 표시됩니다. 그러나 AppSpec 파일의 'files' 섹션을 편집하여 Install 이벤트 중 설치되는 항목을 지정할 수 있습니다.

'hooks' 섹션의 구조

'hooks' 섹션의 구조는 다음과 같습니다.

```
hooks:
   deployment-lifecycle-event-name:
     - location: script-location
       timeout: timeout-in-seconds
       runas: user-name
```

배포 수명 주기 이벤트 이름 뒤의 hook 항목에 다음 요소를 포함할 수 있습니다.

location

필수 사항입니다. 수정의 스크립트 파일 번들 위치 hooks 섹션에서 지정한 스크립트 위치는 애플리케이션 수정 번들의 루트를 기준으로 합니다. 자세한 내용은 CodeDeploy의 개정 계획 섹션을 참조하세요.

제한 시간

선택 사항. 스크립트가 실패로 간주되기 전에 실행할 수 있는 기간(초). 기본값은 3600초(1시간)입니다.

> **Note**
>
> 3600초(1시간)은 각 배포 수명 주기 이벤트에 대한 스크립트 실행에 허용되는 최대 시간입니다. 스크립트가 이 한도를 초과하면 배포가 중지되고 인스턴스에 대한 배포가 실패합니다. 각 배포 수명 주기 이벤트의 모든 스크립트에 대해 timeout에 지정된 총 시간(초)이 이 한도를 초과하지 않아야 합니다.

runas

선택 사항. 스크립트 실행 시 가장하는 사용자. 기본적으로 인스턴스에서 실행 중인 CodeDeploy 에이전트입니다. CodeDeploy는 암호를 저장하지 않기 때문에 runas 사용자에게 암호가 필요합니다. 이 요소는 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에만 적용됩니다.

후크 스크립트에서 파일 참조

AppSpec AppSpec 'hooks' 섹션 섹션에 설명된 대로 스크립트를 CodeDeploy 수명 주기 이벤트에 연결하고 스크립트에서 파일(예: `helper.sh`)을 참조하려는 경우 다음을 사용하여 `helper.sh`를 지정해야 합니다.

- (권장) 절대 경로. 절대 경로 사용 섹션을 참조하세요.
- 상대 경로. 상대 경로 사용 섹션을 참조하세요.

절대 경로 사용

절대 경로를 사용하여 파일을 참조하려면 다음 중 하나를 사용할 수 있습니다.

- AppSpec 파일의 `files` 섹션의 `destination` 속성에서 절대 경로를 지정합니다. 그런 다음 후크 스크립트에서 동일한 절대 경로를 지정합니다. 자세한 내용은 AppSpec 'files' 섹션(EC2/온프레미스 배포만 해당) 섹션을 참조하세요.
- 후크 스크립트에서 동적 절대 경로를 지정합니다. 자세한 내용은 배포 아카이브 위치를 참조하세요.

배포 아카이브 위치

DownloadBundle 수명 주기 이벤트 중에 CodeDeploy 에이전트는 배포를 위한 개정 버전을 다음 형식의 디렉터리로 추출합니다.

root-directory/*deployment-group-id*/*deployment-id*/deployment-archive

경로의 *root-directory* 부분은 항상 다음 표에 표시된 기본값으로 설정되거나 `:root_dir` 구성 설정으로 제어됩니다. 구성 설정에 대한 자세한 내용은 CodeDeploy 에이전트 구성 참조 단원을 참조하세요.

에이전트 플랫폼	기본 루트 디렉터리
Linux – 모든 RPM 분포	`/opt/codedeploy-agent/deployment-root`
Ubuntu 서버 – 모든 Debian 분포	`/opt/codedeploy-agent/deployment-root`
Windows Server	`%ProgramData%\Amazon\CodeDeploy`

후크 스크립트에서 루트 디렉터리 경로와 DEPLOYMENT_ID 및 DEPLOYMENT_GROUP_ID 환경 변수를 사용하여 현재 배포 아카이브에 액세스할 수 있습니다. 사용할 수 있는 변수에 대한 자세한 내용은 후크의 환경 변수 가용성 섹션을 참조하세요.

예를 들어 Linux에서 개정 버전의 루트에 있는 `data.json` 파일에 액세스하는 방법은 다음과 같습니다.

```
#!/bin/bash

rootDirectory="/opt/codedeploy-agent/deployment-root" # note: this will be different if
 you
                                                      # customize the :root_dir
 configuration
dataFile="$rootDirectory/$DEPLOYMENT_GROUP_ID/$DEPLOYMENT_ID/deployment-archive/
data.json"
data=$(cat dataFile)
```

다른 예로, 다음은 Windows에서 Powershell을 사용하여 개정 버전의 루트에 있는 `data.json` 파일에 액세스하는 방법입니다.

```
$rootDirectory="$env:ProgramData\Amazon\CodeDeploy" # note: this will be different if
 you
                                                    # customize the :root_dir
 configuration
$dataFile="$rootDirectory\$env:DEPLOYMENT_GROUP_ID\$env:DEPLOYMENT_ID\deployment-
archive\data.json"
$data=(Get-Content $dataFile)
```

상대 경로 사용

상대 경로를 사용하여 파일을 참조하려면 CodeDeploy 에이전트의 작업 디렉터리를 알아야 합니다. 파일 경로는 이 디렉터리를 기준으로 합니다.

다음 표는 CodeDeploy 에이전트의 지원되는 각 플랫폼에 대한 작업 디렉터리를 보여줍니다.

에이전트 플랫폼	프로세스 관리 방법	수명 주기 이벤트 스크립트용 작업 디렉터리
Linux – 모든 RPM 분포	systemd(기본 값)	`/`
	init.d – 자세히 알아보기	`/opt/codedeploy-agent`
Ubuntu 서버 – 모든 Debian 분포	모두	`/opt/codedeploy-agent`
Windows Server	해당 사항 없음	`C:\Windows\System32`

후크의 환경 변수 가용성

각 배포 수명 주기 이벤트 중 후크 스크립트는 다음 환경 변수에 액세스할 수 있습니다.

APPLICATION_NAME

현재 배포의 일부인 CodeDeploy의 애플리케이션 이름(예: `WordPress_App`)

DEPLOYMENT_ID

ID CodeDeploy가 현재 배포에 할당됩니다(예: `d-AB1CDEF23`).

DEPLOYMENT_GROUP_NAME

현재 배포의 일부인 CodeDeploy의 배포 그룹 이름(예: `WordPress_DepGroup`)

DEPLOYMENT_GROUP_ID

현재 배포의 일부인 CodeDeploy의 배포 그룹 ID(예: `b1a2189b-dd90-4ef5-8f40-4c1c5EXAMPLE`)

LIFECYCLE_EVENT

현재 배포 수명 주기 이벤트의 이름(예: `AfterInstall`)

이러한 환경 변수는 각 배포 수명 주기 이벤트에 대해 로컬에서 적용됩니다.

배포 번들의 소스에 따라 후크 스크립트에 사용할 수 있는 추가 환경 변수가 있습니다.

Amazon S3의 번들

- BUNDLE_BUCKET

  배포 번들을 다운로드할 수 있는 Amazon S3 버킷의 이름입니다(예: `my-s3-bucket`).

- BUNDLE_KEY

  Amazon S3 버킷에서 다운로드한 번들의 객체 키입니다(예: `WordPress_App.zip`).

- BUNDLE_VERSION

  번들의 객체 버전입니다(예: `3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo`). 이 변수는 Amazon S3 버킷에 객체 버전 관리가 활성화된 경우에만 설정됩니다.

- BUNDLE_ETAG

  번들의 객체 etag입니다(예: `b10a8db164e0754105b7a99be72e3fe5-4`).

GitHub에서 제공하는 번들

- BUNDLE_COMMIT

  Git에서 생성한 번들의 SHA256 커밋 해시입니다(예: `d2a84f4b8b650937ec8f73cd8be2c74add5a911ba64df27458ed8229da804a26`).

다음 스크립트는 DEPLOYMENT_GROUP_NAME의 값이 `Staging`과 동일하면 Apache HTTP 서버의 수신 포트를 80이 아니라 9090으로 변경합니다. `BeforeInstall` 배포 수명 주기 이벤트 중에 이 스크립트를 호출해야 합니다.

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/Listen 80/Listen 9090/g' /etc/httpd/conf/httpd.conf
fi
```

다음 스크립트 예제는 DEPLOYMENT_GROUP_NAME 환경 변수의 값이 Staging과 동일하면 오류 로그에 기록되는 메시지의 세부 수준을 경고에서 디버그로 변경합니다. BeforeInstall 배포 수명 주기 이벤트 중에 이 스크립트를 호출해야 합니다.

```
if [ "$DEPLOYMENT_GROUP_NAME" == "Staging" ]
then
    sed -i -e 's/LogLevel warn/LogLevel debug/g' /etc/httpd/conf/httpd.conf
fi
```

다음 스크립트 예제는 지정된 웹 페이지의 텍스트를 이러한 환경 변수의 값을 표시하는 텍스트로 바꿉니다. AfterInstall 배포 수명 주기 이벤트 중에 이 스크립트를 호출해야 합니다.

```
#!/usr/bin/python

import os

strToSearch="<h2>This application was deployed using CodeDeploy.</h2>"
strToReplace="<h2>This page for "+os.environ['APPLICATION_NAME']+"
 application and "+os.environ['DEPLOYMENT_GROUP_NAME']+" deployment group with
 "+os.environ['DEPLOYMENT_GROUP_ID']+" deployment group ID was generated by a
 "+os.environ['LIFECYCLE_EVENT']+" script during "+os.environ['DEPLOYMENT_ID']+"
 deployment.</h2>"

fp=open("/var/www/html/index.html","r")
buffer=fp.read()
fp.close()

fp=open("/var/www/html/index.html","w")
fp.write(buffer.replace(strToSearch,strToReplace))
fp.close()
```

후크 예제

AfterInstall 수명 주기 이벤트에 대한 두 개의 후크를 지정하는 hooks 항목의 예입니다.

```
hooks:
   AfterInstall:
     - location: Scripts/RunResourceTests.sh
       timeout: 180
     - location: Scripts/PostDeploy.sh
       timeout: 180
```

Scripts/RunResourceTests.sh 스크립트는 배포 프로세스의 AfterInstall 단계 중 실행됩니다. 스크립트 실행 시간이 180초(3분)를 넘어가면 배포에 성공하지 못합니다.

'hooks' 섹션에서 지정한 스크립트 위치는 애플리케이션 수정 번들의 루트를 기준으로 합니다. 위의 예제에서 RunResourceTests.sh 파일은 Scripts 디렉터리에 있습니다. Scripts 디렉터리는 번들의 루트 수준에 있습니다. 자세한 내용은 CodeDeploy의 개정 계획 섹션을 참조하세요.

AppSpec 파일 예

이 주제에서는 AWS Lambda 및 EC2/온프레미스 배포에 대한 AppSpec 파일의 예를 제공합니다.

주제

- Amazon ECS 배포용 AppSpec 파일 예제
- AWS Lambda 배포에 대한 AppSpec 파일 예제
- EC2/온프레미스 배포용 AppSpec 파일 예제

Amazon ECS 배포용 AppSpec 파일 예제

다음은 Amazon ECS 서비스를 배포하기 위해 YAML로 작성된 AppSpec 파일의 예입니다.

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: "arn:aws:ecs:us-east-1:111222333444:task-definition/my-task-
definition-family-name:1"
        LoadBalancerInfo:
          ContainerName: "SampleApplicationName"
          ContainerPort: 80
# Optional properties
        PlatformVersion: "LATEST"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets: ["subnet-1234abcd","subnet-5678abcd"]
            SecurityGroups: ["sg-12345678"]
            AssignPublicIp: "ENABLED"
        CapacityProviderStrategy:
          - Base: 1
            CapacityProvider: "FARGATE_SPOT"
```

```
            Weight: 2
          - Base: 0
            CapacityProvider: "FARGATE"
            Weight: 1
Hooks:
  - BeforeInstall: "LambdaFunctionToValidateBeforeInstall"
  - AfterInstall: "LambdaFunctionToValidateAfterInstall"
  - AfterAllowTestTraffic: "LambdaFunctionToValidateAfterTestTrafficStarts"
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterAllowingProductionTraffic"
```

다음은 위의 예제를 JSON으로 작성한 버전입니다.

```
{
    "version": 0.0,
    "Resources": [
        {
            "TargetService": {
                "Type": "AWS::ECS::Service",
                "Properties": {
                    "TaskDefinition": "arn:aws:ecs:us-east-1:111222333444:task-
definition/my-task-definition-family-name:1",
                    "LoadBalancerInfo": {
                        "ContainerName": "SampleApplicationName",
                        "ContainerPort": 80
                    },
                    "PlatformVersion": "LATEST",
                    "NetworkConfiguration": {
                        "AwsvpcConfiguration": {
                            "Subnets": [
                                "subnet-1234abcd",
                                "subnet-5678abcd"
                            ],
                            "SecurityGroups": [
                                "sg-12345678"
                            ],
                            "AssignPublicIp": "ENABLED"
                        }
                    },
                    "CapacityProviderStrategy": [
                        {
                            "Base" : 1,
                            "CapacityProvider" : "FARGATE_SPOT",
```

```
                            "Weight" : 2
                        },
                        {
                            "Base" : 0,
                            "CapacityProvider" : "FARGATE",
                            "Weight" : 1
                        }
                    ]
                }
            }
        }
    ],
    "Hooks": [
        {
            "BeforeInstall": "LambdaFunctionToValidateBeforeInstall"
        },
        {
            "AfterInstall": "LambdaFunctionToValidateAfterInstall"
        },
        {
            "AfterAllowTestTraffic": "LambdaFunctionToValidateAfterTestTrafficStarts"
        },
        {
            "BeforeAllowTraffic":
 "LambdaFunctionToValidateBeforeAllowingProductionTraffic"
        },
        {
            "AfterAllowTraffic":
 "LambdaFunctionToValidateAfterAllowingProductionTraffic"
        }
    ]
}
```

다음은 배포 중 이벤트의 순서입니다.

1. 업데이트된 Amazon ECS 애플리케이션을 대체 작업 세트에 설치하기 전에 `LambdaFunctionToValidateBeforeInstall`(이)라는 Lambda 함수가 실행됩니다.
2. 업데이트된 Amazon ECS 애플리케이션을 대체 작업 세트에 설치했지만 트래픽을 수신하기 전에 `LambdaFunctionToValidateAfterInstall`(이)라는 Lambda 함수가 실행됩니다.
3. 대체 작업 세트의 Amazon ECS 애플리케이션은 테스트 리스너에서 트래픽을 수신하기 시작한 후, `LambdaFunctionToValidateAfterTestTrafficStarts`(이)라는 Lambda 함수가 실행됩니

다. 이 함수는 배포가 계속되는지 여부를 확인하는 확인 테스트를 실행할 가능성이 있습니다. 배포 그룹에서 테스트 리스너를 지정하지 않는 경우, 이 후크는 무시됩니다.

4. `AfterAllowTestTraffic` 후크의 확인 테스트가 완료되고 프로덕션 트래픽이 업데이트된 Amazon ECS 애플리케이션에 제공되기 전에, `LambdaFunctionToValidateBeforeAllowingProductionTraffic`(이)라는 Lambda 함수가 실행됩니다.
5. 프로덕션 트래픽이 대체 작업 세트의 업데이트된 Amazon ECS 애플리케이션에 제공된 후 `LambdaFunctionToValidateAfterAllowingProductionTraffic`(이)라는 Lambda 함수가 실행됩니다.

후크 중에 실행되는 Lambda 함수는 확인 테스트를 수행하거나 트래픽 지표를 수집할 수 있습니다.

AWS Lambda 배포에 대한 AppSpec 파일 예제

다음은 Lambda 함수 버전을 배포하기 위해 YAML로 작성된 AppSpec 파일의 예입니다.

```
version: 0.0
Resources:
  - myLambdaFunction:
      Type: AWS::Lambda::Function
      Properties:
        Name: "myLambdaFunction"
        Alias: "myLambdaFunctionAlias"
        CurrentVersion: "1"
        TargetVersion: "2"
Hooks:
  - BeforeAllowTraffic: "LambdaFunctionToValidateBeforeTrafficShift"
  - AfterAllowTraffic: "LambdaFunctionToValidateAfterTrafficShift"
```

다음은 위의 예제를 JSON으로 작성한 버전입니다.

```
{
  "version": 0.0,
  "Resources": [{
   "myLambdaFunction": {
    "Type": "AWS::Lambda::Function",
    "Properties": {
     "Name": "myLambdaFunction",
     "Alias": "myLambdaFunctionAlias",
     "CurrentVersion": "1",
```

```
     "TargetVersion": "2"
    }
   }
  }],
  "Hooks": [{
    "BeforeAllowTraffic": "LambdaFunctionToValidateBeforeTrafficShift"
      },
      {
    "AfterAllowTraffic": "LambdaFunctionToValidateAfterTrafficShift"
   }
  ]
 }
```

다음은 배포 중 이벤트의 순서입니다.

1. myLambdaFunction(이)라는 Lambda 함수 버전 1의 트래픽을 버전 2로 전환하기 전에 배포가 트래픽 전환을 시작할 준비가 되었는지 확인하기 위해 LambdaFunctionToValidateBeforeTrafficShift(이)라는 Lambda 함수를 실행합니다.
2. LambdaFunctionToValidateBeforeTrafficShift가 종료 코드 0(성공)을 반환하면 myLambdaFunction 버전 2로 트래픽 이동을 시작합니다. 이 배포에 대한 배포 구성은 트래픽 이동 속도를 지정합니다.
3. myLambdaFunction(이)라는 Lambda 함수 버전 1의 트래픽을 버전 2로 전환하는 작업이 완료되면 배포가 성공적으로 완료되었는지 확인하기 위해 LambdaFunctionToValidateAfterTrafficShift(이)라는 Lambda 함수를 실행합니다.

EC2/온프레미스 배포용 AppSpec 파일 예제

다음은 Amazon Linux, Ubuntu Server 또는 RHEL 인스턴스에 대한 인 플레이스 배포를 위한 AppSpec 파일의 예제입니다.

> **Note**
>
> Windows Server 인스턴스에 배포하는 기능은 runas 요소를 지원하지 않습니다. Windows Server 인스턴스를 배포하고 있다면 이 요소를 AppSpec 파일에 포함하지 마십시오.

```
version: 0.0
os: linux
```

```
files:
  - source: Config/config.txt
    destination: /webapps/Config
  - source: source
    destination: /webapps/myApp
hooks:
  BeforeInstall:
    - location: Scripts/UnzipResourceBundle.sh
    - location: Scripts/UnzipDataBundle.sh
  AfterInstall:
    - location: Scripts/RunResourceTests.sh
      timeout: 180
  ApplicationStart:
    - location: Scripts/RunFunctionalTests.sh
      timeout: 3600
  ValidateService:
    - location: Scripts/MonitorService.sh
      timeout: 3600
      runas: codedeployuser
```

Windows Server 인스턴스의 경우 `os: linux`을(를) `os: windows`(으)로 변경합니다. 또한 `destination` 경로를 정규화해야 합니다(예: `c:\temp\webapps\Config` 및 `c:\temp\webapps\myApp`). `runas` 요소를 포함하면 안 됩니다.

다음은 배포 중 이벤트의 순서입니다.

1. `Scripts/UnzipResourceBundle.sh`에 있는 스크립트를 실행합니다.
2. 이전 스크립트가 종료 코드 0(성공)을 반환한 경우 이 스크립트를 `Scripts/UnzipDataBundle.sh`에서 실행합니다.
3. `Config/config.txt` 경로의 파일을 `/webapps/Config/config.txt` 경로로 복사합니다.
4. `source` 디렉터리의 파일을 `/webapps/myApp` 디렉터리로 모두 복사합니다.
5. `Scripts/RunResourceTests.sh`에 있는 스크립트를 실행합니다. 제한 시간은 180초(3분)입니다.
6. `Scripts/RunFunctionalTests.sh`에 있는 스크립트를 실행합니다. 제한 시간은 3600초(1시간)입니다.
7. `Scripts/MonitorService.sh`에 있는 스크립트를 사용자 `codedeploy`로 실행합니다. 제한 시간은 3600초(1시간)입니다.

AppSpec 파일 간격

다음은 AppSpec 간격 지정의 올바른 형식입니다. 대괄호로 묶인 숫자는 항목 간에 있어야 하는 공백 수를 나타냅니다. 예: [4]은(는) 항목 사이에 네 개의 공백을 삽입하는 것을 의미합니다. AppSpec 파일의 공백 위치 및 개수가 올바르지 않으면 CodeDeploy에는 디버깅하기 어려울 수 있는 오류가 발생합니다.

```
version:[1]version-number
os:[1]operating-system-name
files:
[2]-[1]source:[1]source-files-location
[4]destination:[1]destination-files-location
permissions:
[2]-[1]object:[1]object-specification
[4]pattern:[1]pattern-specification
[4]except:[1]exception-specification
[4]owner:[1]owner-account-name
[4]group:[1]group-name
[4]mode:[1]mode-specification
[4]acls:
[6]-[1]acls-specification
[4]context:
[6]user:[1]user-specification
[6]type:[1]type-specification
[6]range:[1]range-specification
[4]type:
[6]-[1]object-type
hooks:
[2]deployment-lifecycle-event-name:
[4]-[1]location:[1]script-location
[6]timeout:[1]timeout-in-seconds
[6]runas:[1]user-name
```

다음은 간격이 올바르게 지정된 AppSpec 파일의 예입니다.

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
```

```
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/change_permissions.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
    - location: scripts/create_test_db.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

간격 지정에 대한 자세한 내용은 YAML 사양을 참조하세요.

AppSpec 파일 및 파일 위치 확인

파일 구문

AWS제공 AppSpec Assistant 스크립트를 사용하여 AppSpec 파일의 내용을 검증할 수 있습니다. GitHub에서 AppSpec 파일 템플릿과 함께 스크립트를 찾을 수 있습니다.

또한 YAML Lint 또는 Online YAML Parser와 같은 브라우저 기반 도구를 사용하여 YAML 구문을 확인할 수 있습니다.

파일 위치

AppSpec 파일을 애플리케이션 소스 콘텐츠 디렉터리 구조의 루트 디렉터리에 저장했는지 확인하려면 다음 명령 중 하나를 실행합니다.

Linux, macOS 또는 Unix 인스턴스에서:

```
ls path/to/root/directory/appspec.yml
```

이 루트 디렉터리에 AppSpec 파일이 없으면 "해당 파일 또는 디렉터리가 없음" 오류가 표시됩니다.

로컬 Windows 인스턴스에서:

```
dir path\to\root\directory\appspec.yml
```

AppSpec 파일이 루트 디렉터리에 없으면 "File Not Found"(파일을 찾을 수 없음) 오류가 표시됩니다.

CodeDeploy 에이전트 구성 참조

CodeDeploy 에이전트가 설치되면 구성 파일은 인스턴스에 있습니다. 인스턴스와 상호 작용 시 CodeDeploy에서 사용할 디렉터리 경로 및 기타 설정을 지정합니다. 이 파일의 일부 구성 옵션은 변경할 수 있습니다.

Amazon Linux, Ubuntu Server 및 Red Hat Enterprise Linux(RHEL) 인스턴스의 경우 구성 파일의 이름은 `codedeployagent.yml`입니다. 그것은 `/etc/codedeploy-agent/conf` 디렉터리로 이동합니다.

Windows Server 인스턴스의 경우 구성 파일의 이름은 `conf.yml`입니다. 그것은 `C:\ProgramData\Amazon\CodeDeploy` 디렉터리로 이동합니다.

구성 설정은 다음과 같습니다.

:log_aws_wire:	CodeDeploy 에이전트`true`가 Amazon S3에서 와이어 로그를 캡처하고 :log_dir: 설정으로 가리키는 위치의 로그 파일에 쓰려면 로 설정합니다. 버전 2.0.x 이상의 경우 파일 이름은 `codedeploy-agent.aws_wire.log` (0640 권한으로 생성됨)입니다. 버전 1.8.x 이하의 경우 파일 이름은 입니다`codedeploy-agent.wire.log` . **Warning** 와이어 로그를 캡처하는 데 필요한 시간 동안만 :log_aws_wire:를 `true`(으)로 설정해야 합니다. 와이어 로그 파일은 매우 큰 크기로 빠르게 증가할 수 있습니다. 유선 로그 출력에는이 설정이 로 설정된 동안 Amazon S3로 전송되거나 Amazon S3에서 전송되는 파일의 일반 텍스트 콘텐츠를 포함한 민감한 정보가

	포함될 수 있습니다`true`. 유선 로그에는 CodeDeploy 배포와 관련된 활동뿐만 아니라 이 설정이 `true`로 설정된 동안 AWS 계정과 연결된 모든 Amazon S3 활동에 대한 정보가 포함됩니다. 기본 설정은 `false`입니다. 이 설정은 모든 인스턴스 유형에 적용됩니다. 이 구성 설정을 사용하려면 Windows Server 인스턴스에 추가해야 합니다.
:log_dir:	CodeDeploy 에이전트 작업과 관련된 로그 파일이 저장되는 인스턴스의 폴더입니다. 기본 설정은 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스의 경우 `'/var/log/aws/codedeploy-agent'` , Windows Server 인스턴스의 경우 `C:\ProgramData\Amazon\CodeDeploy\log` 입니다.
:pid_dir:	`codedeploy-agent.pid` 이(가) 저장된 폴더입니다. 이 파일에는 CodeDeploy 에이전트의 프로세스 ID(PID)가 들어 있습니다. 기본 설정은 `'/opt/codedeploy-agent/state/.pid'` 입니다. 이 설정은 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에만 적용됩니다.
:program_name:	CodeDeploy 에이전트 프로그램 이름입니다. 기본 설정은 `codedeploy-agent` 입니다. 이 설정은 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에만 적용됩니다.

:root_dir:	인스턴스의 관련 수정 버전, 배포 기록 및 배포 스크립트가 저장된 폴더입니다. 기본 설정은 Amazon Linux, Ubuntu Server 및 RHEL 인스턴스의 경우 `/opt/codedeploy-agent/deployment-root` , Windows Server 인스턴스의 경우 `C:\ProgramData\Amazon\CodeDeploy` 입니다.
:verbose:	CodeDeploy 에이전트가 인스턴스에서 디버그 메시지 로그 파일을 인쇄하려면 `true`(으)로 설정합니다. 기본 설정은 `false`입니다.
:wait_between_runs:	보류 중인 배포에 대한 CodeDeploy의 CodeDeploy의 에이전트 폴링 간격(초)입니다. 버전 2.0.x 이상의 경우 기본 설정은 입니다`30`. 버전 1.8.x 이하의 경우 기본 설정은 입니다`1`.
:on_premises_config_file:	온프레미스 인스턴스에서 (Ubuntu Server 및 RHEL의 경우) `codedeploy.onpremises.yml` (이)라는 구성 파일의 대체 위치에 대한 경로 또는 (Windows Server의 경우) `conf.onpremises.yml` (이)라는 구성 파일의 대체 위치에 대한 경로입니다. 기본적으로 이러한 파일은 Ubuntu Server 및 RHEL의 경우 `/etc/codedeploy-agent/conf/codedeploy.onpremises.yml` , Windows Server의 경우 `C:\ProgramData\Amazon\CodeDeploy \conf.onpremises.yml` 에 저장됩니다. 버전 1.0.1.686 및 이후 버전의 CodeDeploy 에이전트에서 사용할 수 있습니다.

:proxy_uri:	(선택 사항) CodeDeploy 에이전트가 CodeDeploy 작업을 AWS 위해 연결할 HTTP 프록시입니다. `https://user:password@my.proxy:443/path?query` 와(과) 유사한 형식을 사용합니다. 버전 1.0.1.824 및 이후 버전의 CodeDeploy 에이전트에서 사용할 수 있습니다.
:max_revisions:	(선택 사항) CodeDeploy 에이전트에서 보관하도록 하려는 배포 그룹의 애플리케이션 수정 버전의 개수입니다. 지정된 수를 초과하는 수정 버전은 모두 삭제됩니다. 양수 정수를 입력합니다. 값을 지정하지 않으면 CodeDeploy는 현재 배포된 수정 이외에 최근 수정 5개를 보유합니다. 버전 1.0.1.966 이상 버전의 CodeDeploy 에이전트에서 지원됩니다.
:enable_auth_policy:	(선택 사항) IAM 권한 부여를 사용하여 CodeDeploy 에이전트가 사용 중인 IAM 역할 또는 사용자의 액세스 제어 및 제한 권한을 구성하려면 `true`(으)로 설정합니다.. Amazon Virtual Private Cloud와 함께 CodeDeploy 사용에 이 값은 `true`이어야 합니다. 기본 설정은 `false`입니다.
:disable_imds_v1:	이 설정은 CodeDeploy 에이전트 1.7.0 이상에서 사용할 수 있습니다. IMDSv2 오류 발생 시 IMDSv1으로의 대체 기능을 비활성화하려면 `true`로 설정합니다. 기본값은 `false`입니다(폴백 활성화).

:disable_core_dumps:	버전 2.0.x 이상. 코어 덤프를 억제`true`하려면 로 설정합니다. 충돌 조사를 `false` 위해를 로 설정합니다. 기본 설정은 `true`입니다.
:enable_command_port:	버전 2.0.x 이상. 에이전트 상태를 쿼리하고 명령을 로컬로 주입`true`하기 위해 루프백 전용(127.0.0.1) 로컬 관리 인터페이스를 활성화하려면 로 설정합니다. 기본 설정은 `false`입니다.
:enable_archive_reuse:	버전 2.1.0 이상. 인스턴스에 이미 개정이 있는 RESTART 배포에서도 항상 애플리케이션 개정을 다운로드`false`하려면 로 설정합니다. 이 설정이 이면 `true`에이전트는 인스턴스에 이미 있는 복사본에서를 설치하고 해당 복사본이 누락되거나 불완전한 경우 개정을 다운로드합니다. 기본 설정은 `true`입니다. 이 동작에 대한 자세한 내용은 섹션을 참조하세요CodeDeploy 에이전트의 버전 기록.
:reject_symlinks_in_bundle:	버전 2.0.x 이상. symlink 또는 hardlink가 포함된 번들을 거부`true`하려면 로 설정합니다. 기본 설정은 `false`입니다.

:reject_path_traversal_in_bundle:	버전 2.0.x 이상. `../` 경로 순회 또는 절대 경로가 포함된 아카이브 항목과 번들 루트를 이스케이프하는 AppSpec 소스 필드를 거부`true`하려면 로 설정합니다. 기본 설정은 `false`입니다.
:reject_unsafe_permissions_in_bundle:	버전 2.0.x 이상. SUID/SGID를 설정하는 SUID/SGID 파일 또는 AppSpec 권한 모드가 포함된 번들을 거부`true`하려면 로 설정합니다. 기본 설정은 `false`입니다.
:reject_unconfined_selinux_in_bundle:	버전 2.0.x 이상. MAC 비활성화 SELinux 유형의 파일이 포함된 번들을 거부`true`하려면 로 설정합니다. 기본 설정은 `false`입니다.
:reject_symlink_permission_targets:	버전 2.0.x 이상. 대상이 심볼 링크인 AppSpec 파일 `permissions` 항목을 거부`true`하려면 로 설정합니다. 기본적으로 에이전트는 소유권, 모드, ACL 및 SELinux 변경 사항을 적용할 때 링크를 따릅니다. 기본 설정은 `false`입니다.

:ignore_ownership_in_bundle:	버전 2.0.x 이상. 아카이브에 기록된 사용자 ID `tar` 및 그룹 ID를 적용하는 `root` 대신 및 `tgz` 번들에서 추출한 파일의 소유권을 로 재설정`true`하려면 로 설정합니다. `zip` 번들의 파일은이 설정에 `root` 관계없이에서 소유합니다. 기본 설정은 `false`입니다.
:restrict_agent_dir_permissions:	버전 2.0.x 이상. Linux 인스턴스만 해당됩니다. 에이전트 설치 루트 아래의 디렉터리와 파일을 기본적으로 사용되는 세계 읽기 모드 대신 루트 전용 액세스로 제한`true`하려면 로 설정합니다. 기본 설정은 `false`입니다.
:restrict_log_dir_permissions:	버전 2.0.x 이상. Linux 인스턴스만 해당됩니다. 에이전트 로그와 업데이터 로그가 포함된 에이전트 로그 디렉터리를 기본적으로 사용되는 세계 읽기 모드 대신 루트 전용 액세스로 제한`true`하려면 로 설정합니다. 기본 설정은 `false`입니다.
:restrict_hook_env_to_allowlist:	버전 2.0.x 이상. 기본 쉘 변수와 CodeDeploy 배포 변수만 포함하는 최소한의 환경에서 수명 주기 이벤트 후크를 실행`true`하려면 로 설정합니다. 기본적으로 후크는 에이전트 프로세스의 전체 환경을 상속합니다. 기본 설정은 `false`입니다.

:strip_loader_env_in_hooks:	버전 2.0.x 이상. Linux 인스턴스 및 버전 2.1.2 이상의 macOS 인스턴스. 수명 주기 이벤트 후크 환경에서 LD_LIBRARY_PATH , 및 LD_AUDIT 변수를 제거true하려면 LD_PRELOAD 로 설정합니다. macOS에서 설정은 대신 , DYLD_INSERT_LIBRARIES , DYLD_LIBRARY_PATH , DYLD_FRAMEWORK_PATH , DYLD_FALLBACK_LIBRARY_PATH DYLD_FALLBACK_FRAMEWORK_PATH DYLD_VERSIONED_LIBRARY_PATH , 및 DYLD_VERSIONED_FRAMEWORK_PATH 변수를 제거합니다. 기본 설정은 false입니다.
:disable_powershell_profile_in_hooks:	버전 2.0.x 이상. Windows Server 인스턴스만 해당됩니다. -NoProfile 및 -NonInteractive 옵션을 사용하여 .ps1 수명 주기 이벤트 후크를 실행true하려면 로 설정합니다. 기본적으로 이러한 후크는 에이전트가 실행되는 사용자의 PowerShell 프로파일을 로드합니다. 기본 설정은 false입니다.
:archive_max_extraction_size:	버전 2.0.x 이상. 선언된 비압축 번들 크기에 대한 한도를 설정합니다. 추출된 콘텐츠가이 값을 초과하면 배포가 실패합니다. 기본값은 한도 없음(무제한)입니다.
:wait_after_error:	정상 작업을 재개하기 전에 오류가 발생한 후 에이전트가 대기하는 초 수입니다. 기본 설정은 30입니다.

:http_read_timeout:	서비스 API 호출에 대한 초 단위의 HTTP 읽기 제한 시간입니다. 기본 설정은 `80`입니다.
:kill_agent_max_wait_time_seconds:	에이전트가 강제 종료 전에 정상적인 종료를 기다리는 최대 초 수입니다. 버전 2.0.x 이상의 경우 이 값은 24시간(86400초)으로 제한됩니다. 기본 설정은 `7200` (2시간)입니다.
:enable_deployments_log:	집계된 배포 로그()를 작성`true`하려면 로 설정합니다`codedeploy-agent-deployments.log` . 기본 설정은 `true`입니다.
:use_fips_mode:	FIPS 준수 엔드포인트`true`를 사용하려면 로 설정합니다. 미국 리전에만 적용됩니다. 기본 설정은 `false`입니다.
:deploy_control_endpoint:	(선택 사항) CodeDeploy 서비스에 대한 사용자 지정 컨트롤 플레인 엔드포인트 URL입니다. 기본값은 재정의 없음입니다(에이전트가 리전에서 엔드포인트를 확인함).
:s3_endpoint_override:	(선택 사항) 번들 다운로드를 위한 사용자 지정 Amazon S3 엔드포인트 URL입니다. 기본값은 재정의 없음입니다(에이전트가 리전에서 Amazon S3 엔드포인트를 확인함).

:s3_download_max_concurrency:	버전 2.1.2 이상. 에이전트가 Amazon S3에서 애플리케이션 개정을 다운로드하는 데 사용하는 병렬 HTTP 범위 요청 수입니다. 병렬 범위에서 큰 개정을 다운로드1하려면 보다 큰 값으로 설정합니다. 1를 사용하면 에이전트가 단일 `GetObject` 요청으로 개정을 다운로드합니다. 위의 값은 로 64 처리되고 640는 로 처리됩니다1. 기본 설정은 1입니다.
:s3_download_chunk_size:	버전 2.1.2 이상. 가 보다 큰 경우 각 범위 요청의 크기:`s3_download_max_concurrency:` 입니다1. "16MiB" 또는와 같은 바이트 수 또는 크기 문자열을 입력합니다"16MB". 이 크기 이하의 개정은 한 번의 요청으로 다운로드됩니다. 64KiB 미만의 값은 64KiB로 처리됩니다. 기본 설정은 "8MiB"입니다.
:ongoing_deployment_tracking:	배포 추적 상태에 사용되는 하위 디렉터리의 이름입니다. 기본 설정은 `ongoing-deployment` 입니다.

관련 주제

CodeDeploy 에이전트 작업

CodeDeploy 에이전트 작업 관리

CloudFormation CodeDeploy 참조용 템플릿

이 섹션에서는 CodeDeploy 배포와 함께 작동하도록 설계된 CloudFormation 리소스, 변환 및 후크를 소개합니다. CodeDeploy용 CloudFormation 후크에서 관리하는 스택 업데이트를 생성하는 방법에 대한 연습은 섹션을 참조하세요. 를 통해 Amazon ECS 블루/그린 배포 생성 CloudFormation

> **Note**
>
> CloudFormation 후크는에 대한 CloudFormation 구성 요소의 일부이며 CodeDeploy 수명 주기 이벤트 후크 AWS 와 다릅니다.

CodeDeploy에서 사용할 수 있는 다른 방법 외에도 CloudFormation 템플릿을 사용하여 다음 작업을 수행할 수 있습니다.

- 애플리케이션 생성
- 배포 그룹을 만들고 대상 수정 버전을 지정합니다.
- 배포 구성을 만듭니다.
- Amazon EC2 인스턴스를 생성합니다.

CloudFormation 는 템플릿을 사용하여 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스입니다. CloudFormation 템플릿은 형식이 JSON 표준을 준수하는 텍스트 파일입니다. 원하는 모든 AWS 리소스를 설명하는 템플릿을 생성하면 CloudFormation 가 해당 리소스를 프로비저닝하고 구성합니다.

자세한 내용은 AWS CloudFormation이란 무엇입니까? 및 AWS CloudFormation 사용 설명서 AWS CloudFormation 템플릿 작업을 참조하세요.

조직의 CodeDeploy와 호환되는 CloudFormation 템플릿을 사용하려는 경우 관리자로서가 CloudFormation 의존하는 AWS 서비스 CloudFormation 및 작업에 대한 액세스 권한을 부여해야 합니다. 애플리케이션, 배포 그룹 및 배포 구성을 생성할 수 있는 권한을 부여하려면 작업할 사용자의 권한 세트에 CloudFormation다음 정책을 추가합니다.

JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

정책에 대한 자세한 내용은 다음 항목을 참조하세요.

- Amazon EC2 인스턴스를 생성할 사용자의 권한 세트에 연결해야 하는 정책을 보려면 CodeDeploy용 Amazon EC2 인스턴스 생성(CloudFormation 템플릿) 섹션을 참조하세요.
- 권한 세트에 정책을 추가하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 Create a permission set(권한 세트 생성)를 참조하세요.
- 사용자를 CodeDeploy 작업 및 리소스의 한정된 세트로 제한하는 방법을 알아보려면 AWS CodeDeploy에 대한 관리형(미리 정의된) 정책 섹션을 참조하세요.

다음 표에는 CloudFormation 템플릿이 사용자를 대신하여 수행할 수 있는 작업이 나와 있으며 CloudFormation 템플릿에 추가할 수 있는 AWS 리소스 유형 및 속성 유형에 대한 자세한 정보 링크가 포함되어 있습니다.

작업	CloudFormation 참조	참조 유형
CodeDeploy 애플리케이션을 만듭니다.	AWS::CodeDeploy::application	CloudFormation 리소스
애플리케이션 수정 버전을 배포하는 데 사용할 배포 그룹에 대한 세부 정보를 만들고 지정합니다. 1	AWS::CodeDeploy::DeploymentGroup	CloudFormation 리소스
배포 중 CodeDeploy가 사용할 배포 규칙, 배포 성공 조건 및 배포 실패 조건 세트를 생성합니다.	AWS::CodeDeploy::DeploymentConfig	CloudFormation 리소스

작업	CloudFormation 참조	참조 유형
Amazon EC2 인스턴스 생성 2	AWS::EC2::instance	CloudFormation 리소스
변환 및 `AWS::CodeDeploy::BlueGreen` 후크를 사용하여 CloudFormation `AWS::CodeDeployBlueGreen` 스택 업데이트를 관리하고, 리소스를 생성하고, CodeDeploy 블루/그린 배포의 트래픽을 이동합니다.[3]	AWS::CodeDeployBlueGreen	`AWS::CodeDeployBlueGreen` 변형은 CloudFormation 에서 호스팅하는 매크로입니다.
	AWS::CodeDeploy::BlueGreen	`AWS::CodeDeploy::BlueGreen` 후크는의 Hook 리소스로 구성됩니다 CloudFormation. 후크에는 지정된 CodeDeploy 수명 주기 이벤트 후크를 가리켜 CodeDeploy AppSpec 파일을 대신하는 파라미터가 포함됩니다.

[1] 배포 그룹의 일부로 배포할 애플리케이션 수정 버전을 지정하는 경우 프로비저닝 프로세스가 완료되면 즉시 대상 수정 버전이 배포됩니다. 템플릿 구성에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 CodeDeploy DeploymentGroup 배포 버전 S3Location 및 CodeDeploy DeploymentGroup 배포 수정 버전 GitHubLocation을 참조하세요.

2 CodeDeploy가 지원되는 리전에서 Amazon EC2 인스턴스를 생성하는 데 사용할 수 있는 템플릿을 제공합니다. 템플릿 사용 또는 생성에 대한 자세한 내용은 CodeDeploy용 Amazon EC2 인스턴스 생성(CloudFormation 템플릿) 단원을 참조하세요.

[3]이 배포 구성에서는 Amazon ECS 블루/그린 배포만 지원됩니다. CloudFormation을 통한 Amazon ECS 블루/그린 배포의 배포 구성에 대한 자세한 내용은 CloudFormation 블루/그린 배포를 위한 배포 구성(Amazon ECS) 섹션을 참조하세요. 를 통한 Amazon ECS 블루/그린 배포 CloudFormation 및 CodeDeploy에서 배포를 보는 방법에 대한 자세한 내용은 섹션을 참조하세요를 통해 Amazon ECS 블루/그린 배포 생성 CloudFormation.

Amazon Virtual Private Cloud와 함께 CodeDeploy 사용

Amazon Virtual Private Cloud(VPC)를 사용하여AWS리소스를 호스팅하는 경우 VPC와 CodeDeploy 간에 프라이빗 연결을 설정할 수 있습니다. 이 연결을 사용하면 CodeDeploy가 퍼블릭 인터넷을 통하지 않고 VPC의 리소스와 통신하게 할 수 있습니다.

Amazon VPC는 정의한 가상 네트워크에서AWS리소스를 시작하는 데 사용할 수 있는AWS서비스입니다. VPC가 있으면 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등 네트워크 설정을 제어할 수 있습니다. VPC 엔드포인트를 사용하면 VPC와AWS서비스 간의 라우팅이AWS네트워크에서 처리되며 IAM 정책을 사용하여 서비스 리소스에 대한 액세스를 제어할 수 있습니다.

VPC를 CodeDeploy에 연결하려면 CodeDeploy에 대해 인터페이스 VPC 엔드포인트를 정의하세요. 인터페이스 엔드포인트는 지원되는AWS서비스로 향하는 트래픽의 진입점 역할을 하는 프라이빗 IP 주소가 있는 탄력적 네트워크 인터페이스입니다. 이 엔드포인트를 이용하면 인터넷 게이트웨이나 NAT(네트워크 주소 변환) 인스턴스 또는 VPN 연결 없이도 CodeDeploy에 안정적이고 확장 가능하게 연결됩니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC란 무엇입니까를 참조하세요.

인터페이스 VPC 엔드포인트는 프라이빗 IP 주소가 있는 탄력적 네트워크 인터페이스를 사용하여 AWS서비스 간 프라이빗 통신을 지원하는AWS PrivateLinkAWS기술로 구동됩니다. 자세한 내용은 AWS PrivateLink 단원을 참조하십시오.

다음은 Amazon VPC 사용자를 위한 단계들입니다. 자세한 내용은 Amazon VPC 사용 설명서의 시작하기를 참조하세요.

가용성

CodeDeploy에는 두 개의 VPC 엔드포인트가 있습니다. 하나는 CodeDeploy 에이전트 작업용이고 다른 하나는 CodeDeploy API 작업용입니다. 아래 표에는 각 엔드포인트에 지원되는AWS리전이 나와 있습니다.

FIPS 엔드포인트에 대한 자세한 내용은 AWS CodeDeploy엔드포인트 및 할당량을 참조하세요.

리전 이름	리전 코드	에이전트 엔드포인트	API 엔드포인트	FIPS 준수 리전 여부
미국 동부(버지니아 북부)	us-east-1	예	예	예

리전 이름	리전 코드	에이전트 엔드포인트	API 엔드포인트	FIPS 준수 리전 여부
미국 동부(오하이오)	us-east-2	예	예	예
미국 서부(캘리포니아 북부)	us-west-1	예	예	예
미국 서부(오리건)	us-west-2	예	예	예
아프리카(케이프타운)	af-south-1	예	예	
Asia Pacific (Hong Kong)	ap-east-1	예	예	
아시아 태평양(하이데라바드)	ap-south-2	예	예	
아시아 태평양(자카르타)	ap-southeast-3	예	예	
아시아 태평양(멜버른)	ap-southeast-4	예	예	
아시아 태평양(말레이시아)	ap-southeast-5	예	예	
아시아 태평양(뉴질랜드)	ap-southeast-6	예	예	
아시아 태평양(태국)	ap-southeast-7	예	예	
아시아 태평양(타이베이)	ap-east-2	예	예	

리전 이름	리전 코드	에이전트 엔드포인트	API 엔드포인트	FIPS 준수 리전 여부
아시아 태평양(뭄바이)	ap-south-1	예	예	
아시아 태평양(오사카)	ap-northeast-3	예	예	
아시아 태평양(서울)	ap-northeast-2	예	예	
아시아 태평양(싱가포르)	ap-southeast-1	예	예	
아시아 태평양(시드니)	ap-southeast-2	예	예	
아시아 태평양(도쿄)	ap-northeast-1	예	예	
캐나다(중부)	ca-central-1	예	예	
중국(베이징)	cn-north-1	예	아니요	
중국(닝샤)	cn-northwest-1	아니요	아니요	
유럽(프랑크푸르트)	eu-central-1	예	예	
유럽(아일랜드)	eu-west-1	예	예	
유럽(런던)	eu-west-2	예	예	
유럽(밀라노)	eu-south-1	예	예	
유럽(파리)	eu-west-3	예	예	
유럽(스페인)	eu-south-2	예	예	
유럽(스톡홀름)	eu-north-1	예	예	

리전 이름	리전 코드	에이전트 엔드포인트	API 엔드포인트	FIPS 준수 리전 여부
유럽(취리히)	eu-central-2	예	예	
이스라엘(텔아비브)	il-central-1	예	예	
중동(바레인)	me-south-1	예	예	
중동(UAE)	me-central-1	예	예	
멕시코(중부)	mx-central-1	예	예	
남아메리카(상파울루)	sa-east-1	예	예	
AWS GovCloud (US-East)	us-gov-east-1	아니요	아니요	예
AWS GovCloud (US-West)	us-gov-west-1	아니요	아니요	예

CodeDeploy에 대한 VPC 엔드포인트 생성

VPC에서 CodeDeploy 사용을 시작하려면 CodeDeploy에 대한 인터페이스 VPC 엔드포인트를 생성합니다. CodeDeploy는 에이전트 Git 작업과 CodeDeploy API 작업에 대해 별도의 엔드포인트가 필요합니다. 비즈니스 요구 사항에 따라 VPC 엔드포인트를 두 개 이상 생성해야 할 수 있습니다. CodeDeploy용 VPC 엔드포인트를 생성하는 경우, AWS서비스를 선택하고, 서비스 이름에 다음 옵션 중에서 선택합니다.

- com.amazonaws.***region***.codedeploy: CodeDeploy API 작업을 위한 VPC 엔드포인트를 생성하려는 경우 이 옵션을 선택합니다. 예를 들어 사용자가AWS CLI, CodeDeploy API 또는AWS SDKs를 사용하여 , `CreateApplication` `GetDeployment`및와 같은 작업을 위해 CodeDeploy와 상호 작용하는 경우이 옵션을 선택합니다`ListDeploymentGroups`.
- com.amazonaws.***##***.codedeploy-fips: FIPS 엔드포인트에 대한 CodeDeploy API 작업을 위한 VPC 엔드포인트를 생성하려면 이 옵션을 선택합니다.

- com.amazonaws.***region***.codedeploy-commands-secure:CodeDeploy 에이전트 작업을 위한 VPC 엔드포인트를 생성하려는 경우 이 옵션을 선택합니다. 에이전트 구성 파일에서 `:enable_auth_policy:`을(를) `true`(으)로 설정하고 필요한 사용 권한을 연결해야 합니다. 자세한 내용은 CodeDeploy 에이전트 및 IAM 권한 구성 단원을 참조하십시오.

Lambda 또는 ECS 배포를 사용하는 경우 com.amazonaws.***region***.codedeploy에 대한 VPC 엔드포인트만 생성하면 됩니다. Amazon EC2 배포를 사용하는 고객은 com.amazonaws.***region***.codedeploy 및 com.amazonaws.***region***.codedeploy-commands-secure 두 가지 모두에 대해 VPC 엔드포인트가 필요합니다.

CodeDeploy 에이전트 및 IAM 권한 구성

CodeDeploy에서 Amazon VPC 엔드포인트를 사용하려면 EC2 또는 온프레미스 인스턴스에 있는 에이전트 구성 파일에서 `:enable_auth_policy:`의 값을 `true`(으)로 설정해야 합니다. 에이전트 구성 파일에 대한 자세한 내용은 CodeDeploy 에이전트 구성 참조 단원을 참조하세요.

또한 Amazon EC2 인스턴스 프로파일(Amazon EC2 인스턴스를 사용하는 경우) 또는 IAM 사용자 또는 역할(온프레미스 인스턴스를 사용하는 경우)에 다음 IAM 권한을 추가해야 합니다.

```
{
  "Statement": [
    {
      "Action": [
        "codedeploy-commands-secure:GetDeploymentSpecification",
        "codedeploy-commands-secure:PollHostCommand",
        "codedeploy-commands-secure:PutHostCommandAcknowledgement",
        "codedeploy-commands-secure:PutHostCommandComplete"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

CodeDeploy 리소스 키트 참조

CodeDeploy가 사용하는 대부분의 파일은 공개적으로 사용 가능한 AWS 리전별 Amazon S3 버킷에 저장됩니다. 이러한 파일에는 샘플 애플리케이션 파일, 템플릿 및 CodeDeploy 에이전트의 설치 파일이 들어 있습니다. 이러한 파일 모음을 CodeDeploy 리소스 키트라고 합니다.

주제

- 리전별 리소스 키트 버킷 이름
- 리소스 키트 콘텐츠
- 리소스 키트 파일 목록 표시
- 리소스 키트 파일 다운로드

리전별 리소스 키트 버킷 이름

이 표에는 가이드의 일부 절차에 필요한 *bucket-name* 대체 이름이 나와 있습니다. 이는 CodeDeploy 리소스 키트 파일이 포함된 Amazon S3 버킷의 이름입니다.

> **Note**
>
> 아시아 태평양(홍콩) 리전의 Amazon S3 버킷에 액세스하려면 AWS 계정에서 리전을 활성화해야 합니다. 자세한 내용은 AWS 리전 관리를 참조하세요.

리전 이름	*bucket-name* 대체	리전 식별자
미국 동부(버지니아 북부)	aws-codedeploy-us-east-1	us-east-1
미국 동부(오하이오)	aws-codedeploy-us-east-2	us-east-2
미국 서부(캘리포니아 북부)	aws-codedeploy-us-west-1	us-west-1
미국 서부(오리건)	aws-codedeploy-us-west-2	us-west-2
아프리카(케이프타운)	aws-codedeploy-af-south-1	af-south-1
아시아 태평양(홍콩)	aws-codedeploy-ap-east-1	ap-east-1
아시아 태평양(하이데라바드)	aws-codedeploy-ap-south-2	ap-south-2

리전 이름	*bucket-name* 대체	리전 식별자
아시아 태평양(자카르타)	aws-codedeploy-ap-southeast-3	ap-southeast-3
아시아 태평양(멜버른)	aws-codedeploy-ap-southeast-4	ap-southeast-4
아시아 태평양(말레이시아)	aws-codedeploy-ap-southeast-5	ap-southeast-5
아시아 태평양(뉴질랜드)	aws-codedeploy-ap-southeast-6	ap-southeast-6
아시아 태평양(태국)	aws-codedeploy-ap-southeast-7	ap-southeast-7
아시아 태평양(타이베이)	aws-codedeploy-ap-east-2	ap-east-2
아시아 태평양(뭄바이)	aws-codedeploy-ap-south-1	ap-south-1
아시아 태평양(오사카)	aws-codedeploy-ap-northeast-3	ap-northeast-3
아시아 태평양(서울)	aws-codedeploy-ap-northeast-2	ap-northeast-2
아시아 태평양(싱가포르)	aws-codedeploy-ap-southeast-1	ap-southeast-1
아시아 태평양(시드니)	aws-codedeploy-ap-southeast-2	ap-southeast-2
아시아 태평양(도쿄)	aws-codedeploy-ap-northeast-1	ap-northeast-1
캐나다(중부)	aws-codedeploy-ca-central-1	ca-central-1
유럽(프랑크푸르트)	aws-codedeploy-eu-central-1	eu-central-1

리전 이름	*bucket-name* 대체	리전 식별자
유럽(아일랜드)	aws-codedeploy-eu-west-1	eu-west-1
유럽(런던)	aws-codedeploy-eu-west-2	eu-west-2
유럽(밀라노)	aws-codedeploy-eu-south-1	eu-south-1
유럽(파리)	aws-codedeploy-eu-west-3	eu-west-3
유럽(스페인)	aws-codedeploy-eu-south-2	eu-south-2
유럽(스톡홀름)	aws-codedeploy-eu-north-1	eu-north-1
유럽(취리히)	aws-codedeploy-eu-central-2	eu-central-2
이스라엘(텔아비브)	aws-codedeploy-il-central-1	il-central-1
Middle East (Bahrain)	aws-codedeploy-me-south-1	me-south-1
중동(UAE)	aws-codedeploy-me-central-1	me-central-1
멕시코(중부)	aws-codedeploy-mx-central-1	mx-central-1
남아메리카(상파울루)	aws-codedeploy-sa-east-1	sa-east-1
AWS GovCloud (US-East)	aws-codedeploy-us-gov-east-1	us-gov-east-1
AWS GovCloud (US-West)	aws-codedeploy-us-gov-west-1	us-gov-west-1

리소스 키트 콘텐츠

다음 표는 CodeDeploy 리소스 키트의 파일 목록입니다.

파일	설명
`LATEST_VERSION`	Amazon EC2 Systems Manager 같은 업데이트 메커니즘에서 CodeDeploy 에이전트의 최신 버전을 확인하기 위해 사용하는 파일입니다.

파일	설명
VERSION	CodeDeploy 에이전트 버전 1.1.0에서 자동 업데이트 메커니즘이 제거되었으며 이 파일은 더 이상 사용되지 않습니다. 인스턴스에서 실행 중 자체 업데이트를 위해 CodeDeploy 에이전트에서 사용하는 파일
codedeploy-agent.noarch.rpm	Amazon Linux 및 Red Hat Enterprise Linux(RHEL)용 CodeDeploy 에이전트 기본 파일 이름이 동일하지만 버전이 다른 파일이 여러 개 있을 수 있습니다(예: -1.0-0).
codedeploy-agent_all.deb	Ubuntu Server용 CodeDeploy 에이전트 기본 파일 이름이 동일하지만 버전이 다른 파일이 여러 개 있을 수 있습니다(예: _1.0-0).
codedeploy-agent.msi	Windows Server용 CodeDeploy 에이전트 기본 파일 이름이 동일하지만 버전이 다른 파일이 여러 개 있을 수 있습니다(예: -1.0-0).
install	CodeDeploy 에이전트를 더욱 쉽게 설치하기 위해 사용할 수 있는 파일 latestv2/ 접두사는 버전 2.0.x 설치 스크립트를 제공합니다.
CodeDeploy_SampleCF_Template.json	Amazon Linux 또는 Windows Server를 실행하는 1~3개의 Amazon EC2 인스턴스에서 시작하는 데 사용할 수 있는 AWS CloudFormation 템플릿입니다. 기본 파일 이름이 동일하지만 버전이 다른 파일이 여러 개 있을 수 있습니다(예: -1.0.0).

파일	설명
`CodeDeploy_SampleCF_ELB_Integration.json`	Apache 웹 서버에서 실행되는 로드 밸런싱된 샘플 웹 사이트를 생성하는 데 사용할 수 있는 AWS CloudFormation 템플릿입니다. 애플리케이션은 생성한 리전의 모든 가용 영역에 걸쳐 있도록 구성됩니다. 이 템플릿은 세 개의 Amazon EC2 인스턴스와 IAM 인스턴스 프로파일을 생성하여 인스턴스에 Amazon S3, Amazon EC2 Auto Scaling CloudFormation및 Elastic Load Balancing의 리소스에 대한 액세스 권한을 부여합니다. 또한 로드 밸런서 및 CodeDeploy 서비스 역할도 생성합니다.
`SampleApp_ELB_Integration.zip`	Elastic Load Balancing 로드 밸런서에 등록된 Amazon EC2 인스턴스에 배포할 수 있는 샘플 애플리케이션 수정
`SampleApp_Linux.zip`	Amazon Linux를 실행하는 Amazon EC2 인스턴스 또는 Ubuntu Server 또는 RHEL 인스턴스에 배포할 수 있는 샘플 애플리케이션 수정 버전. 기본 파일 이름이 동일하지만 버전이 다른 파일이 여러 개 있을 수 있습니다(예: `-1.0`).
`SampleApp_Windows.zip`	Windows Server 인스턴스에 배포할 수 있는 샘플 애플리케이션 수정 버전. 기본 파일 이름이 동일하지만 버전이 다른 파일이 여러 개 있을 수 있습니다(예: `-1.0`).

리소스 키트 파일 목록 표시

파일 목록을 보려면 해당 리전의 aws s3 ls 명령을 사용합니다.

> **Note**
>
> 각 버킷의 파일은 해당 리전의 리소스와 함께 작동하도록 설계되었습니다.

- `aws s3 ls --recursive s3://aws-codedeploy-us-east-2 --region us-east-2`
- `aws s3 ls --recursive s3://aws-codedeploy-us-east-1 --region us-east-1`
- `aws s3 ls --recursive s3://aws-codedeploy-us-west-1 --region us-west-1`
- `aws s3 ls --recursive s3://aws-codedeploy-us-west-2 --region us-west-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ca-central-1 --region ca-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-1 --region eu-west-1`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-2 --region eu-west-2`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-west-3 --region eu-west-3`
- `aws s3 ls --recursive s3://aws-codedeploy-eu-central-1 --region eu-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-il-central-1 --region il-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-east-1 --region ap-east-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-1 --region ap-northeast-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-northeast-2 --region ap-northeast-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-1 --region ap-southeast-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-2 --region ap-southeast-2`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-4 --region ap-southeast-4`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-5 --region ap-southeast-5`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-6 --region ap-southeast-6`

- `aws s3 ls --recursive s3://aws-codedeploy-ap-southeast-7 --region ap-southeast-7`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-east-2 --region ap-east-2`
- `aws s3 ls --recursive s3://aws-codedeploy-mx-central-1 --region mx-central-1`
- `aws s3 ls --recursive s3://aws-codedeploy-ap-south-1 --region ap-south-1`
- `aws s3 ls --recursive s3://aws-codedeploy-sa-east-1 --region sa-east-1`

리소스 키트 파일 다운로드

파일을 다운로드하려면 해당 리전의 aws s3 cp 명령을 사용합니다.

> **Note**
>
> 끝 부분에 마침표(.)를 삽입해야 합니다. 그러면 파일이 현재 디렉터리로 다운로드됩니다.

예를 들어, 다음 명령은 버킷의 /samples/latest/ 폴더 중 하나에서 SampleApp_Linux.zip이라는 파일 하나를 다운로드합니다.

- `aws s3 cp s3://aws-codedeploy-us-east-2/samples/latest/SampleApp_Linux.zip . --region us-east-2`
- `aws s3 cp s3://aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip . --region us-east-1`
- `aws s3 cp s3://aws-codedeploy-us-west-1/samples/latest/SampleApp_Linux.zip . --region us-west-1`
- `aws s3 cp s3://aws-codedeploy-us-west-2/samples/latest/SampleApp_Linux.zip . --region us-west-2`
- `aws s3 cp s3://aws-codedeploy-ca-central-1/samples/latest/SampleApp_Linux.zip . --region ca-central-1`

- ```
  aws s3 cp s3://aws-codedeploy-eu-west-1/samples/latest/SampleApp_Linux.zip . --region
   eu-west-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-eu-west-2/samples/latest/SampleApp_Linux.zip . --region
   eu-west-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-eu-west-3/samples/latest/SampleApp_Linux.zip . --region
   eu-west-3
  ```
- ```
  aws s3 cp s3://aws-codedeploy-eu-central-1/samples/latest/SampleApp_Linux.zip . --
  region eu-central-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-il-central-1/samples/latest/SampleApp_Linux.zip . --
  region il-central-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-east-1/samples/latest/SampleApp_Linux.zip . --region
   ap-east-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-northeast-1/samples/latest/SampleApp_Linux.zip . --
  region ap-northeast-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-northeast-2/samples/latest/SampleApp_Linux.zip . --
  region ap-northeast-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-1/samples/latest/SampleApp_Linux.zip . --
  region ap-southeast-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-2/samples/latest/SampleApp_Linux.zip . --
  region ap-southeast-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-4/samples/latest/SampleApp_Linux.zip . --
  region ap-southeast-4
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-5/samples/latest/SampleApp_Linux.zip . --
  region ap-southeast-5
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-6/samples/latest/SampleApp_Linux.zip . --
  region ap-southeast-6
  ```
  ```

- ```
  aws s3 cp s3://aws-codedeploy-ap-southeast-7/samples/latest/SampleApp_Linux.zip . --region ap-southeast-7
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-east-2/samples/latest/SampleApp_Linux.zip . --region ap-east-2
  ```
- ```
  aws s3 cp s3://aws-codedeploy-mx-central-1/samples/latest/SampleApp_Linux.zip . --region mx-central-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-ap-south-1/samples/latest/SampleApp_Linux.zip . --region ap-south-1
  ```
- ```
  aws s3 cp s3://aws-codedeploy-sa-east-1/samples/latest/SampleApp_Linux.zip . --region sa-east-1
  ```

모든 파일을 다운로드하려면 리전에 대해 다음 명령 중 하나를 사용합니다.

- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-east-2 . --region us-east-2
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-east-1 . --region us-east-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-west-1 . --region us-west-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-us-west-2 . --region us-west-2
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-ca-central-1 . --region ca-central-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-west-1 . --region eu-west-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-west-2 . --region eu-west-2
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-west-3 . --region eu-west-3
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-eu-central-1 . --region eu-central-1
  ```
- ```
  aws s3 cp --recursive s3://aws-codedeploy-il-central-1 . --region il-central-1
  ```
  ```

- `aws s3 cp --recursive s3://aws-codedeploy-ap-east-1 . --region ap-east-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-1 . --region ap-northeast-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-northeast-2 . --region ap-northeast-2`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-1 . --region ap-southeast-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-2 . --region ap-southeast-2`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-4 . --region ap-southeast-4`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-5 . --region ap-southeast-5`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-6 . --region ap-southeast-6`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-southeast-7 . --region ap-southeast-7`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-east-2 . --region ap-east-2`
- `aws s3 cp --recursive s3://aws-codedeploy-mx-central-1 . --region mx-central-1`
- `aws s3 cp --recursive s3://aws-codedeploy-ap-south-1 . --region ap-south-1`
- `aws s3 cp --recursive s3://aws-codedeploy-sa-east-1 . --region sa-east-1`

CodeDeploy 할당량

다음 표에서는 CodeDeploy의 할당량에 대해 설명합니다.

> **Note**
>
> EC2/온프레미스 현재 위치 배포 실행 시간 제한은 다양합니다. 2023년 6월 전에 생성된 사용자 지정 배포 구성의 경우 한도는 8시간입니다. 2023년 6월 이후에 생성된 사용자 지정 배포 구성의 경우 한도는 12시간입니다. 사전 정의된 배포 구성의 경우 한도는 12시간입니다.

이름	기본값	조정 가능	설명
AWS Lambda 배포 실행 시간	지원되는 각 리전: 50	아니요	AWS Lambda 배포를 실행할 수 있는 최대 시간(첫 번째 트래픽 전환과 마지막 트래픽 전환 사이의 최대 시간에 48시간 + 가능한 두 수명 주기 후크 각각에 대해 1시간)
리전별 계정당 연결된 애플리케이션 수	지원되는 각 리전: 1,000	예	단일 리전의 AWS 계정과 연결된 최대 애플리케이션 수
배포 그룹별 연결된 경보 수	지원되는 각 지역: 50	예	배포 그룹과 연결된 경보 최대 수
배포 그룹의 오토 스케일링 수	지원되는 각 리전: 10	예	Amazon EC2 Auto Scaling 그룹의 최대 수
계정당 동시 배포 수	지원되는 각 리전: 1,300	예	AWS 계정과 연결된 최대 동시 배포 수입니다. Amazon EC2 Auto Scaling 그룹에 있는 확장된 Amazon EC2 인스턴스로의 개별 배포는 EC2 인스턴스가 연결된 각 애플리케이션에 대한 단일 동시 배포로 계산됩니다.
배포 그룹별 동시 배포 수	지원되는 각 리전: 1	아니요	배포 그룹에 대한 최대 동시 배포 수. 같은 배포 그룹으로 동일한 애플리케이션이 동시에 배포되지 않

이름	기본값	조정 가능	설명
			도록 방지하기 위한 제한입니다.
계정별 사용자 지정 배포 구성 수	지원되는 각 리전: 50	아니요	AWS 계정과 연결된 최대 사용자 지정 배포 구성 수
단일 애플리케이션과 연결된 배포 그룹 수	지원되는 각 리전: 1,000	예	단일 애플리케이션과 연결된 배포 그룹 최대 수
EC2/온프레미스 블루/그린 배포 실행(시간)	지원되는 각 리전: 109	아니요	EC2/온프레미스 블루/그린 배포를 실행할 수 있는 최대 시간(위의 두 제한 각각의 경우 48시간 + 13개 수명 주기 이벤트 각각에 대해 1시간)
EC2/온프레미스 인플레이스 배포 실행(시간)	지원되는 각 리전: 12	아니요	EC2/온프레미스 인 플레이스(in-place) 배포를 실행할 수 있는 최대 시간
배포 그룹 내 이벤트 알림 트리거 수	지원되는 각 리전: 10	예	배포 그룹 내 이벤트 알림 트리거의 최대 수
계정당 GitHub 연결 토큰 수	지원되는 각 리전: 25	아니요	단일 AWS 계정의 최대 GitHub 연결 토큰 수
EC2/온프레미스 블루/그린 배포 시, 배포 완료부터 기존 인스턴스 종료까지의 시간	지원되는 각 리전: 48	아니요	EC2/온프레미스 블루/그린 배포 시, 배포 완료부터 기존 인스턴스 종료까지의 최대 시간

이름	기본값	조정 가능	설명
EC2/온프레미스 블루/그린 배포 시, 개정 배포부터 대체 인스턴스로의 트래픽 이동까지의 시간	지원되는 각 리전: 48	아니요	EC2/온프레미스 블루/그린 배포 시, 개정 배포부터 대체 인스턴스로의 트래픽 이동까지의 최대 시간
배포당 인스턴스 수	us-east-1: 2,000 us-east-2: 2,000 us-west-2: 2,000 eu-central-1: 2,000 각각의 지원되는 다른 리전: 1,000	예	단일 배포 내 인스턴스의 최대 수
배포 성공 후 기존 배포에서 인스턴스를 종료하기 전까지 블루/그린 배포가 대기할 수 있는 시간(분)	지원되는 각 리전: 2,800	아니요	배포 성공 후 원본 배포에서 인스턴스를 종료하기 전까지 블루/그린 배포가 대기할 수 있는 최대 시간(분)
AWS Lambda canary 또는 선형 배포 중 첫 번째 트래픽 이동과 마지막 트래픽 이동 사이의 분	지원되는 각 리전: 2,880	아니요	AWS Lambda canary 또는 선형 배포 중 첫 번째 트래픽 이동과 마지막 트래픽 이동 사이의 최대 분

이름	기본값	조정 가능	설명
수명 주기 이벤트가 시작되지 않을 시 배포 실패까지 걸리는 시간(분)	지원되는 각 리전: 5	아니요	(1) 콘솔 또는 AWS CLI create-deployment 명령을 사용하여 배포를 트리거하거나 (2) 이전 수명 주기 이벤트가 완료된 후 수명 주기 이벤트가 시작되지 않으면 배포가 실패할 때까지 최대 분 수입니다.
Amazon ECS 서비스와 연결할 수 있는 배포 그룹 수	지원되는 각 리전: 1	아니요	Amazon ECS 서비스와 연결할 수 있는 최대 배포 그룹 수
BatchGetOnPremisesInstances API 작업에 전달될 수 있는 인스턴스 수	지원되는 각 리전: 100	아니요	BatchGetOnPremises Instances API 작업에 전달될 수 있는 최대 인스턴스 수
진행 중이고 동시 배포에 의해 사용되는 계정당 인스턴스 개수	us-east-1: 3,000 각각의 지원되는 다른 리전: 1,000	예	진행 중이고 한 계정과 연결된 동시 배포에 의해 사용될 수 있는 최대 인스턴스 개수
Amazon ECS 배포 중에 트래픽 경로의 리스너 수	지원되는 각 리전: 1	아니요	Amazon ECS 배포 중에 트래픽 경로의 최대 리스너 수
배포 수명 주기 이벤트가 완료되지 않은 경우 실패할 때까지 걸리는 시간(초)	지원되는 각 리전: 3,600초	아니요	배포 수명 주기 이벤트가 완료되지 않은 경우 실패할 때까지 걸리는 최대 시간(초)

이름	기본값	조정 가능	설명
배포 그룹 이름의 크기	지원되는 각 리전: 100	아니요	배포 그룹 이름에 사용되는 최대 문자 수
태그 키 크기	지원되는 각 리전: 128	아니요	태그 키에 사용되는 최대 문자 수
태그 값 크기	지원되는 각 리전: 256	아니요	태그 값에 사용되는 최대 문자 수
배포 그룹의 태그 수	지원되는 각 리전: 10	아니요	배포 그룹 내 태그의 최대 수
AWS Lambda 배포 중에 한 증분으로 이동할 수 있는 트래픽	지원되는 각 리전: 99	아니요	AWS Lambda 배포 중에 한 증분으로 이동할 수 있는 트래픽의 최대 비율

CodeDeploy 문제 해결

이 단원의 주제는 CodeDeploy 사용 시 발생할 수 있는 문제 및 오류를 해결하는 데 유용하게 활용할 수 있습니다.

> **Note**
>
> 배포 프로세스 중 생성되는 로그 파일을 검토하면 다양한 배포 실패의 원인을 파악할 수 있습니다. 간소화를 위해 인스턴스별로 로그 파일을 보는 대신 Amazon CloudWatch Logs를 사용하여 중앙에서 로그 파일을 모니터링하는 것이 좋습니다. 자세한 내용은 Monitoring Deployments with Amazon CloudWatch Tools 단원을 참조하세요.

주제

- 이 시나리오에서는 플릿에 상태에서 정상 인스턴스가 표시되지만 위의 오류도 표시됩니다.
- EC2/온프레미스 배포 문제 해결
- Amazon ECS 배포 문제 해결
- AWS Lambda 배포 문제 해결
- 배포 그룹 관련 문제 해결
- 인스턴스 문제 해결
- GitHub 토큰 문제 해결
- Amazon EC2 Auto Scaling 문제 해결
- 에 대한 오류 코드 AWS CodeDeploy

이 시나리오에서는 플릿에 상태에서 정상 인스턴스가 표시되지만 위의 오류도 표시됩니다.

주제

- 일반적인 문제 해결 체크리스트
- CodeDeploy 배포 리소스는 일부 AWS 리전에서만 지원됩니다.
- 이 설명서의 절차가 CodeDeploy 콘솔과 일치하지 않음
- 필요한 IAM 역할을 사용할 수 없음
- 일부 텍스트 편집기를 사용하여 AppSpec 파일 및 쉘 스크립트를 작성하면 배포에 실패할 수 있음

- macOS에서 Finder를 사용하여 애플리케이션 수정을 번들링하면 배포에 실패할 수 있음

일반적인 문제 해결 체크리스트

다음 체크리스트를 사용하여 실패한 배포 문제를 해결할 수 있습니다.

1. 배포에 실패했는지 확인하려면 CodeDeploy 배포 세부 정보 보기 및 View Instance Details 단원을 확인하세요. 원인을 알 수 없는 경우에는 이 체크리스트의 항목을 살펴보십시오.
2. 다음과 같이 인스턴스가 올바르게 구성되었는지 확인합니다.
   - 지정된 EC2 키 페어로 인스턴스가 시작되었습니까? 자세한 내용은 Amazon EC2 사용 설명서의 EC2 키 페어를 참조하세요.
   - 올바른 IAM 인스턴스 프로파일이 인스턴스에 연결되어 있습니까? 자세한 내용은 CodeDeploy 작업을 위한 Amazon EC2 인스턴스 구성 및 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 단원을 참조하세요.
   - 인스턴스에 태그가 지정되어 있습니까? 자세한 내용은 Amazon EC2 사용 설명서의 콘솔의 태그 작업을 참조하세요.
   - 인스턴스에서 CodeDeploy 에이전트가 설치 및 업데이트되었고 실행 중인가요? 자세한 내용은 CodeDeploy 에이전트 작업 관리 단원을 참조하십시오. 설치된 에이전트의 버전을 확인하려면 CodeDeploy 에이전트의 버전 확인 섹션을 참조하세요.
3. 다음과 같이 애플리케이션 및 배포 그룹 설정을 확인합니다.
   - 애플리케이션 설정을 확인하려면 CodeDeploy로 애플리케이션 세부 정보 보기 단원을 참조하세요.
   - 배포 그룹 설정을 확인하려면 CodeDeploy에서 배포 그룹 세부 정보 보기 단원을 참조하세요.
4. 다음과 같이 애플리케이션 수정이 올바르게 구성되었는지 확인합니다.
   - AppSpec 파일의 형식을 확인합니다. 자세한 내용은 CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 및 CodeDeploy AppSpec 파일 참조 단원을 참조하세요.
   - Amazon S3 버킷 또는 GitHub 리포지토리를 점검하여 애플리케이션이 예상된 위치에 있는지 확인합니다.
   - CodeDeploy 애플리케이션 수정의 세부 정보를 검토하여 애플리케이션 수정이 올바르게 등록되었는지 확인합니다. 자세한 내용은 CodeDeploy를 사용하여 애플리케이션 개정 세부 정보 보기 단원을 참조하세요.
   - Amazon S3에서 배포하는 경우 Amazon S3 버킷을 점검하여 CodeDeploy에 애플리케이션 개정 버전을 다운로드할 수 있는 권한이 부여되었는지 확인합니다. 버킷 정책에 대한 자세한 내용은 배포 사전 조건 단원을 참조하세요.

- GitHub에서 배포하는 경우 GitHub 리포지토리를 점검하여 CodeDeploy에 애플리케이션 수정을 다운로드할 수 있는 권한이 부여되었는지 확인합니다. 자세한 내용은 CodeDeploy에서 배포 만들기 및 CodeDeploy에서 애플리케이션으로 GitHub 인증 단원을 참조하세요.

5. 서비스 역할이 올바르게 구성되었는지 확인합니다. 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하세요.
6. CodeDeploy 시작하기의 단계를 수행했는지 확인하고 다음을 수행합니다.
   - 사용자에게 적절한 권한을 프로비저닝합니다.
   - AWS CLI를 설치 또는 업그레이드한 후 구성합니다.
   - IAM 인스턴스 프로파일 및 서비스 역할을 생성합니다.

   자세한 내용은 에 대한 자격 증명 및 액세스 관리 AWS CodeDeploy 단원을 참조하십시오.
7. AWS CLI 버전 1.6.1 이상을 사용하고 있는지 확인합니다. 설치된 버전을 확인하려면 aws --version을 호출합니다.

실패한 배포 관련 문제를 여전히 해결할 수 없는 경우 이 항목의 다른 문제를 검토합니다.

CodeDeploy 배포 리소스는 일부 AWS 리전에서만 지원됩니다.

AWS CLI 또는 CodeDeploy 콘솔에서 애플리케이션, 배포 그룹, 인스턴스 또는 기타 배포 리소스가 표시되지 않거나 액세스할 수 없는 경우의 AWS 리전 및 엔드포인트에 나열된 리전 중 하나를 참조해야 합니다AWS 일반 참조.

CodeDeploy 배포에 사용되는 EC2 인스턴스 및 Amazon EC2 Auto Scaling 그룹은 이러한 AWS 리전 중 하나에서 시작하고 생성해야 합니다.

를 사용하는 경우에서 aws configure 명령을 AWS CLI실행합니다 AWS CLI. 그런 다음 기본 AWS 리전을 보고 설정할 수 있습니다.

CodeDeploy 콘솔을 사용하는 경우 탐색 모음의 리전 선택기에서 지원되는 AWS 리전 중 하나를 선택합니다.

> **Important**
>
> 중국(베이징) 리전 또는 중국(닝샤) 리전에서 서비스를 이용하려면 이들 리전에 대한 계정 및 자격 증명이 있어야 합니다. 다른 AWS 리전의 계정 및 자격 증명은 베이징 및 닝샤 리전에서는 작동하지 않으며 그 반대의 경우도 마찬가지입니다.
> CodeDeploy 리소스 키트 버킷 이름 및 CodeDeploy 에이전트 설치 절차와 같은 중국 리전의 일부 리소스 관련 정보는 이 CodeDeploy 사용 설명서 에디션에 포함되어 있지 않습니다.

자세한 내용:

- 중국(베이징) 리전 AWS 에서 시작하기의 CodeDeploy
- 중국 리전의 CodeDeploy 사용 설명서(영어 버전 | 중국어 버전)

이 설명서의 절차가 CodeDeploy 콘솔과 일치하지 않음

이 설명서의 절차는 새 콘솔 디자인을 반영하여 작성되었습니다. 이전 버전의 콘솔을 사용하더라도 이 설명서의 여러 가지 개념과 기본 절차가 계속해서 적용됩니다. 새 콘솔에서 도움말에 액세스하려면 정보 아이콘을 선택하세요.

필요한 IAM 역할을 사용할 수 없음

AWS CloudFormation 스택의 일부로 생성된 IAM 인스턴스 프로파일 또는 서비스 역할에 의존하는 경우 스택을 삭제하면 모든 IAM 역할도 삭제됩니다. 이로 인해 IAM 콘솔에 IAM 역할이 더 이상 표시되지 않고 CodeDeploy가 예상대로 더 이상 작동하지 않을 수 있습니다. 이 문제를 해결하려면 삭제된 IAM 역할을 수동으로 다시 만들어야 합니다.

일부 텍스트 편집기를 사용하여 AppSpec 파일 및 쉘 스크립트를 작성하면 배포에 실패할 수 있음

일부 텍스트 편집기에서는 파일에는 인쇄되지 않는 비준수 문자가 포함됩니다. Amazon Linux, Ubuntu Server 또는 RHEL 인스턴스에서 실행하기 위해 텍스트 편집기를 사용하여 AppSpec 파일 또는 쉘 스크립트를 만들거나 수정하면 이러한 파일을 사용하는 모든 배포에 실패할 수 있습니다. CodeDeploy에서 배포 중 이러한 파일을 사용하면 이러한 문자로 인해 해결하기 어려운 AppSpec 파일 검증 실패 및 스크립트 실행 오류가 발생할 수 있습니다.

CodeDeploy 콘솔의 배포 이벤트 세부 정보 페이지에서 로그 보기(View logs)를 선택합니다. (또는를 사용하여 get-deployment-instance 명령을 호출 AWS CLI 합니다.) `invalid character`, `command not found`, `file not found`와 같은 오류를 찾습니다.

이 문제를 해결하려면 다음과 같이 수행하는 것이 좋습니다.

- AppSpec 파일 및 쉘 스크립트 파일에 캐리지 리턴(`^M` 문자) 등 인쇄되지 않는 문자를 포함하는 텍스트 편집기는 사용하지 마십시오.
- AppSpec 파일 및 쉘 스크립트 파일에 캐리지 리턴 등과 같이 인쇄되지 않는 문자를 표시하는 텍스트 편집기를 사용합니다. 그러면 삽입될 수 있는 해당 문자를 모두 찾아서 제거할 수 있습니다. 이러한

유형의 텍스트 편집기에 대한 예를 찾아보려면 인터넷에서 캐리지 리턴을 표시하는 텍스트 편집기를 검색해 보십시오.

- Amazon Linux, Ubuntu Server 또는 RHEL 인스턴스에서 실행되는 쉘 스크립트 파일을 생성하려면 Amazon Linux, Ubuntu Server 또는 RHEL 인스턴스에서 실행되는 텍스트 편집기를 사용합니다. 이러한 유형의 텍스트 편집기에 대한 예를 찾아보려면 인터넷에서 Linux 셸 스크립트 편집기를 검색해 보십시오.
- Windows 또는 macOS에서 텍스트 편집기를 사용하여 Amazon Linux, Ubuntu Server 또는 RHEL 인스턴스에서 실행하기 위한 셸 스크립트 파일을 작성해야 하는 경우 Windows 또는 macOS 형식 텍스트를 Unix 형식으로 변환하는 프로그램 또는 유틸리티를 사용합니다. 이러한 프로그램 및 유틸리티의 예를 찾아보려면 인터넷에서 DOS에서 UNIX로 변환 또는 Mac에서 UNIX로 변환을 검색해 보십시오. 대상 운영 체제에서 변환된 셸 스크립트 파일을 테스트해 보십시오.

macOS에서 Finder를 사용하여 애플리케이션 수정을 번들링하면 배포에 실패할 수 있음

Mac에서 Finder 그래픽 사용자 인터페이스(GUI) 애플리케이션을 사용하여 AppSpec 파일 및 관련 파일과 스크립트를 애플리케이션 수정 아카이브(.zip) 파일로 번들링(압축)하면 배포에 실패할 수 있습니다. 이는 Finder가 .zip 파일에 중간 `__MACOSX` 폴더를 만들어 이 폴더에 구성 요소 파일을 배치하는 것이 원인일 수 있습니다. CodeDeploy가 구성 요소 파일을 찾을 수 없으므로 배포에 실패합니다.

이 문제를 해결하려면 AWS CLI 를 사용하여 구성 요소 파일을 예상 구조로 압축하는 push 명령을 호출하는 것이 좋습니다. 또는 GUI 대신 터미널을 사용하여 구성 요소 파일을 압축할 수 있습니다. 터미널은 중간 `__MACOSX` 폴더를 만들지 않습니다.

EC2/온프레미스 배포 문제 해결

주제

- CodeDeploy 플러그 인 CommandPoller 자격 증명 누락 오류
- "Validation of PKCS7 signed message failed" 메시지와 함께 배포에 실패합니다.
- 동일한 파일을 같은 인스턴스 위치로 배포 또는 다시 배포하려고 하면 실패하고 오류 "The deployment failed because a specified file already exists at this location"이 표시됨
- 파일 경로가 길면 “No such file or directory(해당 파일 또는 디렉터리가 없음)” 오류가 발생함
- 오래 실행되는 프로세스로 인해 배포에 실패할 수 있음
- 배포 로그에 오류를 보고하지 않은 실패한 AllowTraffic 수명 주기 이벤트 문제 해결

- 실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결
- 실패하고 UnknownError: not opened for reading이 표시되는 DownloadBundle 배포 수명 주기 이벤트 문제 해결
- 모든 수명 주기 이벤트를 건너뛰었을 때 문제 해결
- Windows PowerShell 스크립트에서 기본적으로 Windows PowerShell 64비트 버전을 사용하지 못함

> **Note**
>
> 배포 프로세스 중 생성되는 로그 파일을 검토하면 다양한 배포 실패의 원인을 파악할 수 있습니다. 간소화를 위해 인스턴스별로 로그 파일을 보는 대신 Amazon CloudWatch Logs를 사용하여 중앙에서 로그 파일을 모니터링하는 것이 좋습니다. 자세한 내용은 CloudWatch Logs 콘솔에서 CodeDeploy 로그 보기를 참조하세요.

> **Tip**
>
> EC2/온프레미스 배포와 관련된 많은 문제 해결 작업을 자동화하는 런북은 AWS Systems Manager 자동화 런북 참조에서 AWSSupport-TroubleshootCodeDeploy를 참조하세요.

CodeDeploy 플러그 인 CommandPoller 자격 증명 누락 오류

`InstanceAgent::Plugins::CodeDeployPlugin::CommandPoller: Missing credentials - please check if this instance was started with an IAM instance profile`과 비슷한 오류 메시지가 수신되면 다음 중 한 가지가 원인일 수 있습니다.

- 배포하려는 인스턴스에 IAM 인스턴스 프로파일이 연결되어 있지 않습니다.
- IAM 인스턴스 프로파일에 권한이 올바르게 구성되어 있지 않습니다.

IAM 인스턴스 프로파일은 CodeDeploy 에이전트에게 CodeDeploy와 통신하고, Amazon S3에서 개정 버전을 다운로드할 수 있는 권한을 부여합니다. EC2 인스턴스에 대한 자세한 내용은 에 대한 자격 증명 및 액세스 관리 AWS CodeDeploy 단원을 참조하세요. 온프레미스 인스턴스는 Working with On-Premises Instances 단원을 참조하세요.

"Validation of PKCS7 signed message failed" 메시지와 함께 배포에 실패합니다.

이 오류 메시지는 인스턴스가 SHA-1 해시 알고리즘만 지원하는 CodeDeploy 에이전트 버전을 실행하는 중임을 나타냅니다. SHA-2 해시 알고리즘에 대한 지원은 2015년 11월에 출시된 CodeDeploy 에이전트 버전 1.0.1.854에서 도입되었습니다. 2016년 10월 17일부터 CodeDeploy 에이전트 버전 1.0.1.854 이전이 설치되어 있는 경우에는 배포에 실패합니다. 자세한 내용은 SSL 인증서를 위해 SHA256 해시 알고리즘으로 전환하는AWS, 알림: 버전 1.0.1.85 이전 CodeDeploy 호스트 에이전트 사용 중지 및 CodeDeploy 에이전트 업데이트 섹션을 참조하세요.

동일한 파일을 같은 인스턴스 위치로 배포 또는 다시 배포하려고 하면 실패하고 오류 "The deployment failed because a specified file already exists at this location"이 표시됨

CodeDeploy에서 파일을 인스턴스에 배포하려고 하지만, 지정된 대상 위치에 같은 이름의 파일이 이미 있는 경우 해당 인스턴스에 대한 배포에 실패할 수 있습니다. "The deployment failed because a specified file already exists at this location: *location-name*" 오류 메시지가 표시될 수 있습니다. 이는 각 배포 중 CodeDeploy가 먼저 정리 로그 파일에 나열된 파일을 이전 배포에서 모두 삭제하기 때문입니다. 대상 설치 폴더에이 정리 파일에 나열되지 않은 파일이 있는 경우 CodeDeploy 에이전트는 기본적으로 이를 오류로 해석하고 배포에 실패합니다.

> **Note**
>
> Amazon Linux, RHEL 및 Ubuntu Server 인스턴스에서 정리 파일은 `/opt/codedeploy-agent/deployment-root/deployment-instructions/`에 있습니다. Windows Server 인스턴스에서의 위치는 `C:\ProgramData\Amazon\CodeDeploy\deployment-instructions\`입니다.

이러한 오류를 피하려면 기본 동작을 배포 실패 이외의 옵션으로 지정하는 것이 가장 쉬운 방법입니다. 각 배포에 대해 배포에 실패하거나, 정리 파일에 나열되지 않은 파일을 덮어 쓰거나, 인스턴스에 이미 있는 파일을 보관하도록 선택할 수 있습니다.

예를 들어, 마지막 배포 후 인스턴스에 수동으로 파일을 배치했으나 다음 애플리케이션 수정에 이름이 동일한 파일을 추가한 경우 덮어쓰기 옵션이 유용합니다.

애플리케이션 수정 패키지에 추가하지 않고 다음 배포의 일부로 포함하려는 파일에 대해 보관 옵션을 선택할 수 있습니다. 또한 애플리케이션 파일이 프로덕션 환경에 이미 있는데 처음으로 CodeDeploy를

사용하여 배포하려고 하는 경우에는 보관 옵션이 유용합니다. 자세한 내용은 EC2/온프레미스 컴퓨팅 플랫폼의 배포 생성(콘솔) 및 기존 컨텐츠의 롤백 동작 단원을 참조하세요.

`The deployment failed because a specified file already exists at this location` 배포 문제 해결

CodeDeploy가 대상 배포 위치에서 감지한 콘텐츠를 덮어 쓰거나 보관하는 옵션을 지정하지 않은 경우(또는 프로그래밍 방식 명령에서 기존 콘텐츠를 처리하기 위한 배포 옵션을 지정하지 않은 경우)에는 오류를 해결하도록 선택할 수 있습니다.

다음 정보는 콘텐츠를 보관 또는 덮어쓰도록 선택하지 않은 경우에만 적용됩니다.

동일한 이름 및 위치를 사용해 파일을 다시 배포하려는 경우 이전에 사용한 것과 동일한 기본 배포 그룹 ID로 애플리케이션 이름 및 배포 그룹을 지정하면 다시 배포에 성공할 가능성이 더욱 높습니다. CodeDeploy는 기본 배포 그룹 ID를 사용하여 재배포 전에 제거할 파일을 식별합니다.

다음과 같은 이유로 새 파일을 배포 또는 인스턴스의 같은 위치로 동일한 파일 다시 배포에 실패할 수 있습니다.

- 동일한 수정을 같은 인스턴스로 다시 배포하기 위해 다른 애플리케이션 이름을 지정했습니다. 배포 그룹 이름이 동일하다고 하더라도 다른 애플리케이션 이름을 사용하면 다른 기본 배포 그룹 ID가 사용되기 때문에 다시 배포에 실패합니다.
- 애플리케이션의 배포 그룹을 삭제한 다음 다시 만든 후 해당 배포 그룹으로 동일한 수정을 다시 배포하려고 했습니다. 배포 그룹 이름이 동일하다고 하더라도 CodeDeploy에서 다른 기본 배포 그룹 ID를 참조하기 때문에 다시 배포에 실패합니다.
- CodeDeploy에서 애플리케이션 및 배포 그룹을 삭제한 다음 삭제한 것과 동일한 이름을 사용해 새 애플리케이션 및 배포 그룹을 만들었습니다. 그런 다음 이전 배포 그룹에 배포한 수정을 동일한 이름을 가진 새 배포 그룹에 다시 배포하려고 했습니다. 애플리케이션 및 배포 그룹 이름이 동일하다고 하더라도 CodeDeploy는 삭제한 배포 그룹의 ID를 참조하기 때문에 다시 배포에 실패합니다.
- 수정을 배포 그룹 하나에 배포한 다음 동일한 수정을 같은 인스턴스의 다른 배포 그룹에 배포했습니다. CodeDeploy는 다른 기본 배포 그룹 ID를 참조하기 때문에 두 번째 배포에 실패합니다.
- 수정을 배포 그룹 하나에 배포한 다음 다른 수정을 같은 인스턴스의 다른 배포 그룹에 배포했습니다. 두 번째 배포 그룹에서 배포하려고 하는 파일 중 이름과 위치가 같은 파일이 하나 이상 있습니다. CodeDeploy가 두 번째 배포를 시작하기 전에 기존 파일을 제거하지 않기 때문에 두 번째 배포에 실패합니다. 두 배포 모두 서로 다른 배포 그룹 IDs.
- CodeDeploy에서 수정을 배포했으나 이름과 위치가 동일한 파일이 하나 이상 있습니다. 기본적으로 CodeDeploy에서 배포를 시작하기 전에 기존 파일을 제거하지 않기 때문에 배포에 실패합니다.

이러한 상황을 해결하려면 다음 중 하나를 수행하세요.

- 이전에 배포된 위치 및 인스턴스에서 파일을 제거한 다음 배포를 다시 시도합니다.
- ApplicationStop 또는 BeforeInstall 배포 수명 주기 이벤트 시 수정의 AppSpec 파일에서 수정에서 설치하려고 하는 파일과 일치하는 파일을 모든 위치에서 삭제하는 사용자 지정 스크립트를 지정합니다.
- 이전 배포의 일부가 아닌 위치 또는 인스턴스로 파일을 배포 또는 다시 배포합니다.
- 애플리케이션 또는 배포 그룹을 삭제하기 전에 인스턴스로 파일을 복사하지 않도록 지정하는 AppSpec 파일이 포함된 수정을 배포합니다. 배포에 대해 삭제하려는 것과 동일한 기본 애플리케이션 및 배포 그룹 ID를 사용하는 애플리케이션 이름 및 배포 그룹 이름을 지정합니다. (get-groument-grout-name 명령을 사용하여 배포 그룹 ID를 검색할 수 있습니다.) CodeDeploy는 기본 배포 그룹 ID와 AppSpec 파일을 사용하여 이전의 성공한 배포에서 설치한 파일을 모두 제거합니다.

파일 경로가 길면 “No such file or directory(해당 파일 또는 디렉터리가 없음)” 오류가 발생함

Windows 인스턴스에 배포할 때 appspec.yml 파일의 파일 섹션에 파일 경로가 260자를 초과하는 경우 다음과 비슷한 오류가 발생하며 배포가 실패하는 것을 볼 수 있습니다.

```
No such file or directory @ dir_s_mkdir - C:\your-long-file-path
```

이 오류는 Microsoft 설명서에 설명된 대로 Windows에서 기본적으로 260자를 초과하는 파일 경로를 허용하지 않기 때문에 발생합니다.

CodeDeploy 에이전트 버전 1.4.0 이상에서는 에이전트 설치 프로세스에 따라 두 가지 방법으로 긴 파일 경로를 활성화할 수 있습니다.

CodeDeploy 에이전트가 아직 설치되지 않은 경우:

1. CodeDeploy 에이전트를 설치하려는 시스템에서 다음 명령을 사용하여 `LongPathsEnabled` Windows 레지스트리 키를 활성화합니다.

   ```
   New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
            -Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
   ```

2. CodeDeploy 에이전트를 설치합니다. 자세한 내용은 CodeDeploy 에이전트 설치 단원을 참조하십시오.

CodeDeploy 에이전트가 이미 설치된 경우:

1. CodeDeploy 에이전트 시스템에서 다음 명령을 사용하여 LongPathsEnabled Windows 레지스트리 키를 활성화합니다.

```
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem"
-Name "LongPathsEnabled" -Value 1 -PropertyType DWORD -Force
```

2. 레지스트리 키 변경 사항을 적용하려면 CodeDeploy 에이전트를 다시 시작합니다. 에이전트를 다시 시작하려면 다음 명령을 사용합니다.

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

오래 실행되는 프로세스로 인해 배포에 실패할 수 있음

Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에 배포하는 경우, 오래 실행되는 프로세스를 시작하는 배포 스크립트가 있으면 CodeDeploy에서는 배포 수명 주기 이벤트를 오랜 시간 기다린 뒤 배포에 실패합니다. 해당 이벤트에서 실행이 예상되는 포그라운드 및 백그라운드 프로세스보다 오래 실행되는 프로세스가 있으면 이 프로세스가 예상대로 실행 중이더라도 CodeDeploy는 배포를 중지하고 배포에 실패합니다.

예를 들어, 애플리케이션 수정의 루트에는 `after-install.sh` 및 `sleep.sh`라는 두 파일이 있습니다. 이 AppSpec 파일에는 다음과 같은 지침이 포함되어 있습니다.

```
version: 0.0
os: linux
files:
  - source: ./sleep.sh
    destination: /tmp
hooks:
  AfterInstall:
    - location: after-install.sh
      timeout: 60
```

`after-install.sh` 파일이 AfterInstall 애플리케이션 수명 주기 이벤트 중 실행됩니다. 이 파일의 내용은 다음과 같습니다.

```
#!/bin/bash
/tmp/sleep.sh
```

sleep.sh 파일에는 프로그램 실행을 3분(180초) 동안 일시 중지하여 오래 실행되는 프로세스를 시뮬레이션하는 다음과 같은 내용이 포함되어 있습니다.

```
#!/bin/bash
sleep 180
```

after-install.sh가 sleep.sh를 호출할 경우 sleep.sh가 시작되어 3분(180초) 동안 실행됩니다. 이는 CodeDeploy에서 sleep.sh(및 관계에 따라 after-install.sh)의 실행이 중지될 것으로 예상하는 시간보다 2분(120초)이 더 지난 시간입니다. 1분(60초)의 제한 시간이 지나면 CodeDeploy는 sleep.sh가 예상대로 계속해서 실행되더라도 AfterInstall 애플리케이션 수명 주기 이벤트에서 배포를 중지하고 배포에 실패합니다. 다음 오류가 표시됩니다.

`Script at specified location: after-install.sh failed to complete in 60 seconds`.

after-install.sh에 앰퍼샌드(&)를 추가하기만 해서는 sleep.sh가 백그라운드에서 실행되도록 할 수 없습니다.

```
#!/bin/bash
# Do not do this.
/tmp/sleep.sh &
```

앰퍼샌드를 추가하면 배포가 최대 1시간의 기본 배포 수명 주기 이벤트 제한 시간 동안 보류 중인 상태로 남아 있은 후 CodeDeploy가 이전처럼 AfterInstall 애플리케이션 수명 주기 이벤트에서 배포를 중지하고 배포에 실패합니다.

after-install.sh에서 다음과 같이 프로세스 실행 시작 후 CodeDeploy가 계속해서 실행되도록 하는 sleep.sh를 호출합니다.

```
#!/bin/bash
/tmp/sleep.sh > /dev/null 2> /dev/null < /dev/null &
```

이전 호출에서 sleep.sh는 백그라운드에서 실행하기 시작하려는 프로세스의 이름으로, stdout, stderr 및 stdin을 /dev/null로 리디렉션합니다.

배포 로그에 오류를 보고하지 않은 실패한 AllowTraffic 수명 주기 이벤트 문제 해결

AllowTraffic 수명 주기 이벤트 중 블루/그린 배포에 실패하지만 배포 로그에서 실패 원인을 확인할 수 없는 경우가 있습니다.

이 실패는 일반적으로 배포 그룹의 트래픽을 관리하는 데 사용되는 Classic Load Balancer, Application Load Balancer 또는 네트워크 로드 밸런서에 대해 Elastic Load Balancing에서 상태 확인이 잘못 구성되었기 때문에 발생합니다.

이 문제를 해결하려면 로드 밸런서의 상태 확인 구성에서 모든 오류를 검토해 수정합니다.

Classic Load Balancer의 경우 Classic Load Balancers 사용 설명서의 상태 확인 구성 및 Elastic Load Balancing API 참조 버전 2012-06-01의 ConfigureHealthCheck을 참조하세요.

Application Load Balancers의 경우 Application Load Balancer 사용 설명서의 대상 그룹에 대한 상태 확인을 참조하세요.

네트워크 로드 밸런서의 경우 Network Load Balancer 사용 설명서의 대상 그룹에 대한 상태 확인을 참조하세요.

실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결

배포 중 CodeDeploy 에이전트는 이전에 성공한 배포의 AppSpec 파일에서 ApplicationStop, BeforeBlockTraffic 및 AfterBlockTraffic에 대해 지정된 스크립트를 실행합니다. 기타 모든 스크립트는 현재 배포의 AppSpec 파일에서 실행됩니다. 이러한 스크립트 중 하나가 오류를 포함하고 있고 성공적으로 실행하지 않으면 배포에 실패할 수 있습니다.

이러한 실패의 가능한 원인은 다음과 같습니다.

- CodeDeploy 에이전트는 올바른 위치에서 *deployment-group-id*_last_successful_install 파일을 찾지만, *deployment-group-id*_last_successful_install 파일에 지정된 위치가 존재하지 않습니다.

  Amazon Linux, Ubuntu Server 및 RHEL 인스턴스에서 이 파일은 /opt/codedeploy-agent/deployment-root/deployment-instructions에 있어야 합니다.

  Windows Server 인스턴스에서 이 파일의 위치는 C:\ProgramData\Amazon\CodeDeploy\deployment-instructions 폴더여야 합니다.
- *deployment-group-id*_last_successful_install 파일에 지정된 위치에서 AppSpec 파일이 잘못되었거나 스크립트가 성공적으로 실행되지 않습니다.
- 스크립트에 해결할 수 없는 오류가 포함되어 있어 성공적으로 실행할 수 없습니다.

CodeDeploy 콘솔을 사용하여 이러한 이벤트 중 배포에 실패할 수 있는 이유를 조사합니다. 배포의 세부 정보 페이지에서 [View events]를 선택합니다. 인스턴스 세부 정보 페이지의 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 행에서 로그 보기를 선택합니다. 또는 AWS CLI 를 사용하여 get-deployment-instance 명령을 호출합니다.

이전에 성공한 배포에 포함된, 성공적으로 실행할 수 없는 스크립트가 실패의 원인인 경우 배포를 만들고 ApplicationStop, BeforeBlockTraffic 및 AfterBlockTraffic 실패를 무시하도록 지정합니다. 이렇게 하는 방법은 두 가지입니다.

- CodeDeploy 콘솔을 사용하여 배포를 만듭니다. [Create deployment] 페이지의 [ApplicationStop lifecycle event failure]에서 [Don't fail the deployment to an instance if this lifecycle event on the instance fails]을 선택합니다.
- AWS CLI 를 사용하여 create-deployment 명령을 호출하고 `--ignore-application-stop-failures` 옵션을 포함합니다.

그러면 애플리케이션 수정을 다시 배포하는 경우 이러한 3가지 수명 주기 이벤트 중 하나가 실패하더라도 배포는 계속 진행됩니다. 새 수정에 이러한 수명 주기 이벤트에 대한 수정된 스크립트가 포함되어 있으면 이러한 수정 사항을 적용하지 않아도 향후 배포에 성공할 수 있습니다.

실패하고 UnknownError: not opened for reading이 표시되는 DownloadBundle 배포 수명 주기 이벤트 문제 해결

Amazon S3에서 애플리케이션 수정 버전을 배포하려고 하는데 DownloadBundle 배포 수명 주기 이벤트 중 배포에 실패하고 `UnknownError: not opened for reading` 오류가 표시됩니다.

- 내부 Amazon S3 서비스 오류가 있습니다. 애플리케이션 수정을 다시 배포합니다.
- EC2 인스턴스의 IAM 인스턴스 프로파일에 Amazon S3에서 애플리케이션 개정 버전에 액세스할 수 있는 권한이 없습니다. Amazon S3 버킷 정책에 대한 자세한 내용은 Amazon S3에 CodeDeploy의 개정 푸시(EC2 온프레미스 배포 전용) 및 배포 사전 조건 섹션을 참조하세요.
- 에 배포하는 인스턴스는 한 AWS 리전(예: 미국 서부(오레곤))과 연결되지만 애플리케이션 개정이 포함된 Amazon S3 버킷은 다른 AWS 리전(예: 미국 동부(버지니아 북부))과 연결됩니다. 애플리케이션 개정이 인스턴스와 동일한 AWS 리전과 연결된 Amazon S3 버킷에 있는지 확인합니다.

배포의 이벤트 세부 정보 페이지에 있는 Download bundle 행에서 로그 보기를 선택합니다. 또는 AWS CLI 를 사용하여 get-deployment-instance 명령을 호출합니다. 오류가 발생하면 출력에 오류 코드 `UnknownError` 및 오류 메시지 `not opened for reading`과 함께 오류가 표시되어야 합니다.

이 오류가 발생한 이유를 확인하려면:

1. 하나 이상의 인스턴스에서 유선 로깅을 활성화한 후 애플리케이션 수정을 다시 배포합니다.
2. 유선 로깅 파일을 검사하여 오류를 찾습니다. 이 문제에 대한 일반적인 오류 메시지에는 "액세스 거부됨(access denied)"이라는 문구가 포함됩니다.
3. 로그 파일을 검사한 후에는 유선 로깅을 비활성화하여, 이후 인스턴스에서 출력에 일반 텍스트로 표시될 수 있는 민감한 정보의 양과 로그 파일의 크기를 줄이는 것이 좋습니다.

유선 로깅 파일을 찾아 유선 로깅을 활성화 및 비활성화하는 방법에 대한 자세한 내용은 CodeDeploy 에이전트 구성 참조의 `:log_aws_wire:` 단원을 참조하세요.

모든 수명 주기 이벤트를 건너뛰었을 때 문제 해결

EC2 또는 온프레미스 배포 수명 주기 이벤트를 모두 건너뛸 경우 "`The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing problems. (Error code: HEALTH_CONSTRAINTS)`"와 비슷한 오류 메시지가 수신될 수 있습니다. 이때 몇 가지 가능한 원인과 해결책은 다음과 같습니다.

- CodeDeploy 에이전트가 인스턴스에 설치되어 있지 않거나 실행되고 있지 않을 수 있습니다. CodeDeploy 에이전트의 실행 여부를 확인하려면:
  - Amazon Linux RHEL 또는 Ubuntu 서버일 경우 다음과 같이 실행합니다.

    ```
    systemctl status codedeploy-agent
    ```

  - Windows일 경우 다음과 같이 실행합니다.

    ```
    powershell.exe -Command Get-Service -Name CodeDeployagent
    ```

  CodeDeploy 에이전트가 설치되어 있지 않거나, 혹은 실행되지 않으면 CodeDeploy 에이전트가 실행 중인지 확인 단원을 참조하세요.

  인스턴스가 포트 443을 사용해서 CodeDeploy 또는 Amazon S3 퍼블릭 엔드포인트에 도달하지 못할 수 있습니다. 다음 중 하나를 시도하세요.

  - 퍼블릭 IP 주소를 인스턴스에 할당한 후 라우팅 테이블을 사용해 인터넷 액세스를 허용합니다. 이때 인스턴스에 연결된 보안 그룹이 포트 443을 통한 아웃바운드 액세스를 허용해야 합니다 (HTTPS). 자세한 내용은 CodeDeploy 에이전트용 통신 프로토콜 및 포트 단원을 참조하십시오.
  - 인스턴스가 프라이빗 서브넷에 프로비저닝되어 있을 경우에는 라우팅 테이블에서 인터넷 게이트웨이가 아닌 NAT 게이트웨이를 사용합니다. 자세한 내용은 NAT 게이트웨이 단원을 참조하세요.
- CodeDeploy 서비스 역할은 권한이 필요하지 않을 수도 있습니다. CodeDeploy 서비스 역할을 구성하려면 2단계: CodeDeploy에 대한 서비스 역할 생성 단원을 참조하세요.
- HTTP 프록시를 사용할 때는 이 프록시를 CodeDeploy 에이전트 구성 파일의 `:proxy_uri:` 설정에서 지정해야 합니다. 자세한 내용은 CodeDeploy 에이전트 구성 참조 단원을 참조하십시오.
- 배포 인스턴스의 날짜 및 시간 서명이 배포 요청의 날짜 및 시간 서명과 일치하지 않을 수도 있습니다. CodeDeploy 에이전트 로그 파일에서 "`Cannot reach InstanceService: Aws::CodeDeployCommand::Errors::InvalidSignatureException - Signature expired`"와 유사한 오류 메시지가 있는지 찾습니다. 오류 메시지가 있다면 "InvalidSignatureException – 서명 만료: [time]이 이제 [time]보다 이전" 배포 오류 문제 해결의 단계를 따르십시오. 자세한 내용은 CodeDeploy EC2/온프레미스 배포에 대한 로그 데이터 보기 단원을 참조하십시오.
- 인스턴스 실행 중 메모리 또는 하드 디스크 공간 부족으로 인해 CodeDeploy 에이전트가 실행을 멈췄을 수도 있습니다. 이때는 CodeDeploy 에이전트 구성에서 `max_revisions` 설정을 업데이트하여 인스턴스에 보관된 배포 수를 줄여 봅니다. EC2 인스턴스에서 배포 수를 줄인 후에도 문제가 지속되면 용량이 더 큰 인스턴스를 사용하는 것이 좋습니다. 예를 들어 인스턴스 유형이 `t2.small`이라면 `t2.medium`으로 바꿔 사용하세요. 자세한 내용은 CodeDeploy 에이전트에 의해 설치된 파일, CodeDeploy 에이전트 구성 참조 및 인스턴스 유형 섹션을 참조하세요.
- 배포하려는 인스턴스에 IAM 인스턴스 프로파일이 연결되어 있지 않거나 IAM 인스턴스 프로파일이 필요한 권한 없이 연결되어 있을 수 있습니다.
  - IAM 인스턴스 프로파일이 인스턴스에 연결되어 있지 않다면 필요한 권한과 함께 인스턴스 프로파일을 생성하여 연결합니다.
  - IAM 인스턴스 프로파일이 이미 인스턴스에 연결되어 있다면 필요한 권한이 있는지 확인합니다.

  연결된 인스턴스 프로파일이 필요한 권한과 함께 구성되어 있는지 확인한 후 인스턴스를 다시 시작합니다. 자세한 내용은 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 섹션과 Amazon EC2 사용 설명서에서 Amazon EC2의 IAM 역할을 참조하세요.

Windows PowerShell 스크립트에서 기본적으로 Windows PowerShell 64비트 버전을 사용하지 못함

배포의 일부로 실행 중인 Windows PowerShell 스크립트가 64비트 기능을 사용하는 경우(예를 들어 32비트보다 더 많은 메모리를 사용하기 때문에 애플리케이션에서 64비트 버전에서만 제공하는 라이브러리만 허용 또는 호출함) 이 스크립트가 충돌하거나 그렇지 않으면 예상대로 실행되지 않을 수 있습니다. 기본적으로 CodeDeploy가 Windows PowerShell 32비트 버전을 사용하여 애플리케이션 수정의 일부인 Windows PowerShell 스크립트를 실행하는 것이 원인입니다.

Windows PowerShell 64비트 버전을 사용해 실행해야 하는 스크립트의 시작 부분에 다음과 같은 코드를 추가합니다.

```
# Are you running in 32-bit mode?
#   (\SysWOW64\ = 32-bit mode)

if ($PSHOME -like "*SysWOW64*")
{
  Write-Warning "Restarting this script under 64-bit Windows PowerShell."

  # Restart this script under 64-bit Windows PowerShell.
  #   (\SysNative\ redirects to \System32\ for 64-bit mode)

  & (Join-Path ($PSHOME -replace "SysWOW64", "SysNative") powershell.exe) -File `
    (Join-Path $PSScriptRoot $MyInvocation.MyCommand) @args

  # Exit 32-bit script.

  Exit $LastExitCode
}

# Was restart successful?
Write-Warning "Hello from $PSHOME"
Write-Warning "  (\SysWOW64\ = 32-bit mode, \System32\ = 64-bit mode)"
Write-Warning "Original arguments (if any): $args"

# Your 64-bit script code follows here...
# ...
```

이 코드의 파일 경로 정보가 합리적이지 않은 것처럼 보일 수 있지만 32비트 Windows PowerShell은 다음과 같은 경로를 사용합니다.

```
c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
```

64비트 Windows PowerShell은 다음과 같은 경로를 사용합니다.

```
c:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Amazon ECS 배포 문제 해결

주제

- 대체 작업 세트를 기다리는 동안 시간 초과 발생
- 알림이 계속되기를 기다리는 동안 시간 초과 발생
- IAM 역할에 충분한 권한이 없음
- 상태 콜백을 기다리는 동안 배포 시간 초과
- 하나 이상의 수명 주기 이벤트 검증 함수가 실패하여 배포 실패
- 다음 오류로 인해 ELB를 업데이트할 수 없음: 기본 작업 세트 대상 그룹은 리스너 뒤에 있어야 함
- Auto Scaling을 사용할 때 때때로 배포가 실패함
- ALB만 점진적인 트래픽 라우팅을 지원하므로 배포 그룹을 생성/업데이트할 때는 대신 AllAtOnce 트래픽 라우팅을 대신 사용해야 함
- 배포에 성공했는데도 대체 작업 세트가 Elastic Load Balancing 상태 확인에 실패하고 애플리케이션이 다운됨
- 배포 그룹에 여러 로드 밸런서를 연결할 수 있나요?
- 로드 밸런서 없이 CodeDeploy 블루/그린 배포를 수행할 수 있나요?
- 배포 중에 새 정보로 Amazon ECS 서비스를 업데이트하려면 어떻게 해야 하나요?

대체 작업 세트를 기다리는 동안 시간 초과 발생

문제: CodeDeploy를 사용하여 Amazon ECS 애플리케이션을 배포하는 동안 다음 오류 메시지가 표시됩니다.

```
The deployment timed out while waiting for the replacement task set to
become healthy. This time out period is 60 minutes.
```

가능한 원인: 작업 정의 파일 또는 기타 배포 관련 파일에 오류가 있는 경우 이 오류가 발생할 수 있습니다. 예를 들어 작업 정의 파일의 `image` 필드에 오타가 있는 경우 Amazon ECS가 잘못된 컨테이너 이미지를 가져오려고 시도하고 계속 실패하여 이 오류가 발생합니다.

가능한 해결 방법 및 다음 단계:

- 작업 정의 파일과 기타 파일의 오타 및 구성 문제를 해결합니다.
- 관련 Amazon ECS 서비스 이벤트를 확인하고 대체 작업이 정상적으로 진행되지 않는 이유를 알아봅니다. Amazon ECS 이벤트에 대한 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 Amazon ECS 이벤트를 참조하세요.
- Amazon Elastic Container Service 개발자 안내서의 Amazon ECS 문제 해결 섹션에서 이벤트의 메시지와 관련된 오류를 확인합니다.

알림이 계속되기를 기다리는 동안 시간 초과 발생

문제: CodeDeploy를 사용하여 Amazon ECS 애플리케이션을 배포하는 동안 다음 오류 메시지가 표시됩니다.

`The deployment timed out while waiting for a notification to continue. This time out period is` *`n`* `minutes.`

가능한 원인: 배포 그룹을 생성할 때 트래픽을 언제 다시 라우팅할지 지정에 대기 시간을 지정했지만 대기 시간이 만료되기 전에 배포가 완료되지 못한 경우 이 오류가 발생할 수 있습니다.

가능한 해결 방법 및 다음 단계:

- 배포 그룹에서 트래픽을 언제 다시 라우팅할지 지정을 더 많은 시간으로 설정하고 다시 배포합니다. 자세한 내용은 Amazon ECS 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 섹션을 참조하세요.
- 배포 그룹에서 트래픽을 언제 다시 라우팅할지 지정을 즉시 트래픽 다시 라우팅으로 변경하고 다시 배포합니다. 자세한 내용은 Amazon ECS 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 섹션을 참조하세요.
- `--deployment-wait-type` 옵션을 로 설정하여 재배포한 다음 aws deploy continue-deployment AWS CLI 명령을 실행합니다`READY_WAIT`. 트래픽을 언제 다시 라우팅할지 지정에 지정된 시간이 만료되기 전에 이 명령을 실행해야 합니다.

IAM 역할에 충분한 권한이 없음

문제: CodeDeploy를 사용하여 Amazon ECS 애플리케이션을 배포하는 동안 다음 오류 메시지가 표시됩니다.

`The IAM role` *`role-arn`* `does not give you permission to perform operations in the following AWS service: AWSLambda.`

가능한 원인: AppSpec 파일의 Hooks 섹션에서 Lambda 함수를 지정했지만 Lambda 서비스에 CodeDeploy 권한을 부여하지 않은 경우 이 오류가 발생할 수 있습니다.

가능한 해결 방법: CodeDeploy 서비스 역할에 `lambda:InvokeFunction` 권한을 추가합니다. 이 권한을 추가하려면 다음 AWS관리형 정책 중 하나를 **AWSCodeDeployRoleForECS** 또는 **AWSCodeDeployRoleForECSLimited** 역할에 추가합니다. 이러한 정책 및 CodeDeploy 서비스 역할에 정책을 추가하는 방법에 대한 자세한 내용은 2단계: CodeDeploy에 대한 서비스 역할 생성 섹션을 참조하세요.

상태 콜백을 기다리는 동안 배포 시간 초과

문제: CodeDeploy를 사용하여 Amazon ECS 애플리케이션을 배포하는 동안 다음 오류 메시지가 표시됩니다.

```
The deployment timed out while waiting for a status callback. CodeDeploy
expects a status callback within one hour after a deployment hook is
invoked.
```

가능한 원인: AppSpec 파일의 Hooks 섹션에서 Lambda 함수를 지정했지만 Lambda 함수가 CodeDeploy에 `Succeeded` 또는 `Failed` 상태를 반환하는 데 필요한 `PutLifecycleEventHookExecutionStatus` API를 호출하지 못한 경우 이 오류가 발생할 수 있습니다.

가능한 해결 방법 및 다음 단계:

- AppSpec 파일에서 지정한 Lambda 함수가 사용하는 Lambda 실행 역할에 `codedeploy:putlifecycleEventHookExecutionStatus` 권한을 추가합니다. 이 권한은 Lambda 함수가 CodeDeploy에 `Succeeded` 또는 `Failed` 상태를 반환할 수 있도록 허용합니다. Lambda 실행 역할에 대한 자세한 내용은 AWS Lambda 사용 설명서의 Lambda 실행 역할을 참조하세요.
- Lambda 함수 코드와 실행 로그를 검토해 Lambda 함수가 CodeDeploy의 `PutLifecycleEventHookExecutionStatus` API를 호출하여 수명 주기 검증 테스트 `Succeeded` 또는 `Failed`에 대해 CodeDeploy에 알리고 있는지 확인합니다. `putlifecycleEventHookExecutionStatus` API에 대한 자세한 내용은 AWS CodeDeploy API 참조의 PutLifecycleEventHookExecutionStatus를 참조하세요. Lambda 실행 로그에 대한 자세한 내용은 AWS Lambda에 대한 Amazon CloudWatch Logs 액세스를 참조하세요.

하나 이상의 수명 주기 이벤트 검증 함수가 실패하여 배포 실패

문제: CodeDeploy를 사용하여 Amazon ECS 애플리케이션을 배포하는 동안 다음 오류 메시지가 표시됩니다.

```
The deployment failed because one or more of the lifecycle event validation functions failed.
```

가능한 원인: AppSpec 파일의 Hooks 섹션에서 Lambda 함수를 지정했지만 Lambda 함수가 PutLifecycleEventHookExecutionStatus를 호출할 때 CodeDeploy에 Failed를 반환한 경우 이 오류가 발생할 수 있습니다. 이 실패는 CodeDeploy에 수명 주기 검증 테스트가 실패했음을 나타냅니다.

가능한 다음 단계: Lambda 실행 로그를 검토하여 검증 테스트 코드가 실패한 이유를 확인합니다. Lambda 실행 로그에 대한 자세한 내용은 AWS Lambda에 대한 Amazon CloudWatch Logs 액세스를 참조하세요.

다음 오류로 인해 ELB를 업데이트할 수 없음: 기본 작업 세트 대상 그룹은 리스너 뒤에 있어야 함

문제: CodeDeploy를 사용하여 Amazon ECS 애플리케이션을 배포하는 동안 다음 오류 메시지가 표시됩니다.

```
The ELB could not be updated due to the following error: Primary taskset target group must be behind listener
```

가능한 원인: 선택적 테스트 리스너를 구성했는데 잘못된 대상 그룹으로 구성된 경우 이 오류가 발생할 수 있습니다. CodeDeploy의 테스트 리스너에 대한 자세한 내용은 Amazon ECS 배포를 시작하기 전 및 Amazon ECS 배포 중에 발생하는 일 섹션을 참조하세요. 작업 세트에 대한 자세한 내용은 Amazon Elastic Container Service API 참조의 TaskSet 및 AWS CLI 명령 참조의 Amazon ECS 섹션에 있는 describe-task-set를 참조하세요.

가능한 해결 방법: Elastic Load Balancing의 프로덕션 리스너와 테스트 리스너가 모두 현재 워크로드를 제공하는 대상 그룹을 가리키고 있는지 확인합니다. 다음 세 곳에서 확인할 수 있습니다.

- Amazon EC2에서 로드 밸런서의 리스너 및 규칙 설정. 자세한 내용은 Application Load Balancer 사용 설명서의 Application Load Balancer를 위한 리스너 또는 Network Load Balancer 사용 설명서의 Network Load Balancer를 위한 리스너를 참조하세요.

- Amazon ECS에서 클러스터의 서비스 네트워킹 구성 아래. 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 Application Load Balancer 및 Network Load Balancer 고려 사항을 참조하세요.
- CodeDeploy의 배포 그룹 설정. 자세한 내용은 Amazon ECS 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 섹션을 참조하세요.

Auto Scaling을 사용할 때 때때로 배포가 실패함

문제: CodeDeploy와 함께 Auto Scaling을 사용하고 있는데 배포가 실패하는 경우가 있습니다. 이 문제의 증상에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 서비스 자동 크기 조정 및 블루/그린 배포 유형을 사용하도록 구성된 서비스의 경우 배포 중에 자동 크기 조정이 차단되지 않지만 일부 상황에서는 배포가 실패할 수 있습니다라는 주제를 참조하세요.

가능한 원인: CodeDeploy와 Auto Scaling 프로세스가 충돌하는 경우 이 문제가 발생할 수 있습니다.

수정 방법: `RegisterScalableTarget` API(또는 해당 `register-scalable-target` AWS CLI 명령)를 사용하여 CodeDeploy 배포 중에 Auto Scaling 프로세스를 일시 중지하고 재개합니다. 자세한 내용을 알아보려면 Application Auto Scaling 사용 설명서의 Application Auto Scaling의 조정 일시 중지 및 재개를 참조하세요.

> **Note**
>
> CodeDeploy는 `RegisterScaleableTarget`를 직접 호출할 수 없습니다. 이 API를 사용하려면 Amazon Simple Notification Service(또는 Amazon CloudWatch)에 알림 또는 이벤트를 전송하도록 CodeDeploy를 구성해야 합니다. 그런 다음 Lambda 함수를 호출하도록 Amazon SNS(또는 CloudWatch)를 구성하고, `RegisterScalableTarget` API를 호출하도록 Lambda 함수를 구성해야 합니다. Auto Scaling 작업을 일시 중지하려면 `SuspendedState` 파라미터를 `true`로 설정하고, 재개하려면 `false`로 설정하여 `RegisterScalableTarget` API를 호출해야 합니다.
>
> CodeDeploy가 전송하는 알림 또는 이벤트는 배포가 시작될 때(Auto Scaling 일시 중지 작업을 트리거하기 위해) 또는 배포가 성공, 실패 또는 중지될 때(Auto Scaling 재개 작업을 트리거하기 위해) 발생해야 합니다.
>
> Amazon SNS 알림 또는 CloudWatch 이벤트를 생성하도록 CodeDeploy를 구성하는 방법에 대한 자세한 내용은 Amazon CloudWatch Events를 사용하여 배포 모니터링 및 Monitoring Deployments with Amazon SNS Event Notifications 섹션을 참조하세요.

ALB만 점진적인 트래픽 라우팅을 지원하므로 배포 그룹을 생성/업데이트할 때는 대신 AllAtOnce 트래픽 라우팅을 대신 사용해야 함

문제: CodeDeploy에서 배포 그룹을 생성하거나 업데이트하는 동안 다음 오류 메시지가 표시됩니다.

```
Only ALB supports gradual traffic routing, use AllAtOnce Traffic routing
instead when you create/update Deployment group.
```

가능한 원인: Network Load Balancer를 사용 중이고 `CodeDeployDefault.ECSAllAtOnce` 이외의 미리 정의된 배포 구성을 사용하려고 하면 이 오류가 발생할 수 있습니다.

수정 방법:

- 미리 정의된 배포 구성을 `CodeDeployDefault.ECSAllAtOnce`로 변경합니다. 이는 Network Load Balancer에서 지원하는 유일한 미리 정의된 배포 구성입니다.

  미리 정의된 배포 구성에 대한 자세한 내용은 Amazon ECS 컴퓨팅 플랫폼에 대해 미리 정의된 배포 구성 섹션을 참조하세요.
- 로드 밸런서를 Application Load Balancer로 변경합니다. Application Load Balancer는 미리 정의된 모든 배포 구성을 지원합니다. Application Load Balancer 생성에 대한 자세한 내용은 CodeDeploy Amazon ECS 배포를 위한 로드 밸런서, 대상 그룹 및 리스너 설정 섹션을 참조하세요.

배포에 성공했는데도 대체 작업 세트가 Elastic Load Balancing 상태 확인에 실패하고 애플리케이션이 다운됨

문제: CodeDeploy에 배포가 성공했다고 표시되지만 대체 작업 세트가 Elastic Load Balancing의 상태 확인에 실패하고 애플리케이션이 다운되었습니다.

가능한 원인: 이 문제는 CodeDeploy 일괄 배포를 수행했고 대체(그린) 작업 세트에 Elastic Load Balancing 상태 확인 실패를 유발하는 잘못된 코드가 포함된 경우 발생할 수 있습니다. all-at-once 배포 구성을 사용하면 트래픽이 대체 작업 세트로 이동된 후(즉, CodeDeploy의 `AllowTraffic` 수명 주기 이벤트가 발생한 후) 로드 밸런서의 상태 확인이 대체 작업 세트에서 실행되기 시작합니다. 따라서 트래픽이 이동한 후 대체 작업 세트에서 상태 확인이 실패하지만 이전에는 실패하지 않습니다. CodeDeploy에서 생성하는 수명 주기 이벤트에 대한 자세한 내용은 Amazon ECS 배포 중에 발생하는 일 섹션을 참조하세요.

수정 방법:

- 배포 구성을 일괄에서 Canary 또는 Linear로 변경합니다. 카나리아 또는 선형 구성에서 로드 밸런서의 상태 확인은 CodeDeploy가 대체 환경에 애플리케이션을 설치하는 동안과 트래픽이 이동되기 전에(즉, `Install` 수명 주기 이벤트 중 및 `AllowTraffic` 이벤트 전) 대체 작업 세트에서 실행되기 시작합니다. 애플리케이션 설치 중에 트래픽이 이동하기 전에 검사를 실행하도록 허용하면 잘못된 애플리케이션 코드가 탐지되어 애플리케이션이 공개되기 전에 배포가 실패하게 됩니다.

  Canary 또는 Linear 배포를 구성하는 방법에 대한 자세한 내용은 CodeDeploy에서 배포 그룹 설정 변경 섹션을 참조하세요.

  Amazon ECS 배포 중에 실행되는 CodeDeploy 수명 주기 이벤트에 대한 자세한 내용은 Amazon ECS 배포 중에 발생하는 일 섹션을 참조하세요.

  > **Note**
  >
  > Canary 및 Linear 배포 구성은 Application Load Balancer에서만 지원됩니다.

- 일괄 배포 구성을 유지하려면 테스트 리스너를 설정하고 `BeforeAllowTraffic` 수명 주기 후크를 사용하여 대체 작업 세트의 상태를 확인합니다. 자세한 내용은 Amazon ECS 배포를 위한 수명 주기 이벤트 후크 목록 섹션을 참조하세요.

배포 그룹에 여러 로드 밸런서를 연결할 수 있나요?

아니요. 여러 Application Load Balancer 또는 Network Load Balancer를 사용하려면 CodeDeploy 블루/그린 배포 대신 Amazon ECS 롤링 업데이트를 사용하세요. 롤링 업데이트에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 롤링 업데이트를 참조하세요. Amazon ECS에 여러 로드 밸런서를 사용하는 방법에 대한 자세한 내용을 알아보려면 Amazon Elastic Container Service 개발자 안내서의 서비스에 여러 대상 그룹 등록을 참조하세요.

로드 밸런서 없이 CodeDeploy 블루/그린 배포를 수행할 수 있나요?

아니요. 로드 밸런서 없이 CodeDeploy 블루/그린 배포를 수행할 수 없습니다. 로드 밸런서를 사용할 수 없는 경우 Amazon ECS의 롤링 업데이트 기능을 대신 사용하세요. Amazon ECS 롤링 업데이트에 대한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 롤링 업데이트를 참조하세요.

배포 중에 새 정보로 Amazon ECS 서비스를 업데이트하려면 어떻게 해야 하나요?

배포를 수행하는 동안 CodeDeploy가 Amazon ECS 서비스를 새 파라미터로 업데이트하도록 하려면 AppSpec 파일의 `resources` 섹션에서 해당 파라미터를 지정하세요. CodeDeploy는 작업 정의 파일 및 컨테이너 이름 파라미터와 같은 몇 가지 Amazon ECS 파라미터만 지원합니다. CodeDeploy에서 업데이트할 수 있는 Amazon ECS 파라미터의 전체 목록은 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션 섹션을 참조하세요.

> **Note**
>
> CodeDeploy에서 지원하지 않는 파라미터로 Amazon ECS 서비스를 업데이트해야 하는 경우 다음 작업을 완료합니다.
>
> 1. 업데이트하려는 파라미터를 사용하여 Amazon ECS의 `UpdateService` API를 호출합니다. 업데이트할 수 있는 파라미터의 전체 목록은 Amazon Elastic Container Service API 참조의 UpdateService를 참조하세요.
> 2. 작업에 변경 내용을 적용하려면 새 Amazon ECS 블루/그린 배포를 생성합니다. 자세한 내용은 Amazon ECS 컴퓨팅 플랫폼에 대한 배포 생성(콘솔) 섹션을 참조하세요.

AWS Lambda 배포 문제 해결

주제

- AWS Lambda 롤백이 구성되지 않은 Lambda 배포를 수동으로 중지한 후 배포가 실패함

AWS Lambda 롤백이 구성되지 않은 Lambda 배포를 수동으로 중지한 후 배포가 실패함

경우에 따라 배포에 지정된 Lambda 함수 별칭이 두 가지 서로 다른 함수 버전을 참조할 수 있습니다. 이 경우 Lambda 함수를 배포하기 위한 다음 시도가 실패합니다. Lambda 배포에 구성된 롤백이 없고 수동으로 중지한 경우 배포가 이러한 상태로 될 수 있습니다. 계속하려면 AWS Lambda 콘솔을 사용하여 함수가 두 버전 간에 트래픽을 이동하도록 구성되지 않았는지 확인합니다.

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/lambda/ AWS Lambda 콘솔을 엽니다.

2. 왼쪽 창에서 함수를 선택합니다.
3. CodeDeploy 배포에 있는 Lambda 함수의 이름을 선택합니다.
4. Aliases(별칭)에서 CodeDeploy 배포에 사용된 별칭을 선택한 다음 Edit(편집)을 선택합니다.
5. Weighted alias(가중치 기반 별칭)에서 **none**을 선택합니다. 이렇게 하면 트래픽의 백분율이나 가중치가 여러 버전으로 이동하도록 별칭이 구성되지 않습니다. 버전에서 선택한 버전을 메모해 두십시오.
6. 저장을 선택합니다.
7. CodeDeploy 콘솔을 열고 5단계의 드롭다운 메뉴에 표시된 버전의 배포를 시도합니다.

배포 그룹 관련 문제 해결

인스턴스에 배포 그룹의 일부로 태그를 지정해도 애플리케이션이 새 인스턴스로 자동으로 배포되지 않음

CodeDeploy는 새로 태그가 지정된 인스턴스로 애플리케이션을 자동으로 배포하지 않습니다. 배포 그룹에서 새 배포를 만들어야 합니다.

CodeDeploy를 사용하여 Amazon EC2 Auto Scaling 그룹의 새 EC2 인스턴스에 자동으로 배포할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling과 CodeDeploy 통합 단원을 참조하십시오.

인스턴스 문제 해결

주제

- 태그를 올바르게 설정해야 함
- AWS CodeDeploy 인스턴스에 에이전트를 설치하고 실행해야 합니다.
- 배포 중 인스턴스가 종료되면 최대 1시간 동안 배포에 실패하지 않음
- 인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석
- 실수로 삭제된 경우 새 CodeDeploy 로그 파일 만들기
- "InvalidSignatureException – 서명 만료: [time]이 이제 [time]보다 이전" 배포 오류 문제 해결

태그를 올바르게 설정해야 함

list-deployment-instances 명령을 사용하여 배포에 사용되는 인스턴스에 태그가 올바르게 지정되었는지 확인합니다. 출력에서 EC2 인스턴스가 누락된 경우 EC2 콘솔을 사용하여 인스턴스에 대해 태그가

설정되었는지 확인합니다. 자세한 내용은 Amazon EC2 사용 설명서의 콘솔의 태그 작업을 참조하세요.

> **Note**
>
> 인스턴스에 태그를 지정한 다음 즉시 CodeDeploy를 사용하여 해당 인스턴스에 애플리케이션을 배포하면 이 인스턴스가 배포에 포함되지 않을 수 있습니다. 이는 CodeDeploy가 태그를 읽으려면 몇 분 가량 걸릴 수 있기 때문입니다. 따라서 인스턴스에 태그를 지정하고 5분 이상 기다린 다음 해당 인스턴스에 배포하는 것이 좋습니다.

AWS CodeDeploy 인스턴스에 에이전트를 설치하고 실행해야 합니다.

인스턴스에서 CodeDeploy 에이전트가 설치되어 실행 중인지 확인하려면 CodeDeploy 에이전트가 실행 중인지 확인 단원을 참조하세요.

CodeDeploy 에이전트를 설치, 제거 또는 다시 설치하려면 CodeDeploy 에이전트 설치 단원을 참조하세요.

배포 중 인스턴스가 종료되면 최대 1시간 동안 배포에 실패하지 않음

CodeDeploy는 각 배포 수명 주기 이벤트가 완료될 때까지 실행되도록 1시간의 기간을 제공합니다. 이 기간은 길게 실행되는 스크립트에 충분한 시간입니다.

수명 주기 이벤트 진행 중 스크립트가 완료될 때까지 실행되지 않는 경우(예: 인스턴스 또는 CodeDeploy 에이전트가 종료된 경우)에는 최대 1시간 가량 지난 후 배포 상태가 Failed로 표시될 수 있습니다. 스크립트에 지정된 제한 시간이 1시간보다 짧은 경우에도 마찬가지입니다. 이는 인스턴스가 종료되면 CodeDeploy 에이전트도 종료되어 스크립트를 더 이상 처리하지 못하기 때문입니다.

그러나 인스턴스가 수명 주기 이벤트 사이에 종료되거나 첫 번째 수명 주기 이벤트 단계 시작 전에 종료되면 불과 5분 뒤에 시간 초과가 발생합니다.

인스턴스에 대한 배포 실패를 조사하기 위해 로그 파일 분석

배포의 인스턴스 상태가 Succeeded 이외의 다른 상태이면 배포 로그 파일을 검토해 문제를 식별할 수 있습니다. 배포 로그 데이터에 액세스하는 방법은 CodeDeploy EC2/온프레미스 배포에 대한 로그 데이터 보기 단원을 참조하세요.

실수로 삭제된 경우 새 CodeDeploy 로그 파일 만들기

인스턴스에서 배포 로그 파일을 실수로 삭제한 경우 CodeDeploy는 대체 로그 파일을 만들지 않습니다. 새 로그 파일을 만들려면 인스턴스에 로그인한 후 다음 명령을 실행합니다.

Amazon Linux, Ubuntu Server 또는 RHEL 인스턴스의 경우, 다음 명령을 아래의 순서대로 한 번에 하나씩 실행합니다.

```
systemctl stop codedeploy-agent
```

```
systemctl start codedeploy-agent
```

Windows Server 인스턴스:

```
powershell.exe -Command Restart-Service -Name codedeployagent
```

"InvalidSignatureException – 서명 만료: [time]이 이제 [time]보다 이전" 배포 오류 문제 해결

CodeDeploy는 작업을 수행하기 위해 정확한 시간 참조가 필요합니다. 인스턴스의 날짜와 시간이 잘못 설정되어 있으면 배포 요청의 서명 날짜와 일치하지 않아 CodeDeploy가 해당 요청을 거부합니다.

잘못된 시간 설정과 관련된 배포 실패를 피하려면 다음 항목을 참조하세요.

- Linux 인스턴스의 시간 설정
- Windows 인스턴스에 대한 시간 설정

GitHub 토큰 문제 해결

유효하지 않은 GitHub OAuth 토큰

2017년 6월 이후에 만든 CodeDeploy 애플리케이션은 각 AWS 리전에 GitHub OAuth 토큰을 사용합니다. 특정 AWS 리전에 연결된 토큰을 사용하면 GitHub 리포지토리에 액세스할 수 있는 CodeDeploy 애플리케이션을 더 잘 제어할 수 있습니다.

GitHub 토큰 오류가 발생하면 현재 유효하지 않은 이전 토큰일 수 있습니다.

유효하지 않은 GitHub OAuth 토큰 문제를 해결하려면

1. 다음 방법 중 하나를 사용하여 이전 토큰을 제거합니다.
   - API를 사용하여 이전 토큰을 삭제하려면 DeleteGitHubAccountToken을 사용합니다.
   - AWS Command Line Interface를 사용하여 이전 토큰을 제거하려면
     a. 토큰이 있는 컴퓨터로 이동합니다.
     b. 이 컴퓨터에 AWS CLI 가 설치되어 있는지 확인합니다. 설치 지침은 AWS Command Line Interface 사용 설명서에서 AWS CLI설치, 업데이트 및 제거를 참조하세요.
     c. 토큰이 있는 컴퓨터에서 다음 명령을 입력합니다.

        ```
        aws delete-git-hub-account-token
        ```

        명령 구문에 대한 세부 정보는 delete-git-hub-account-token을 참조하세요.
2. 새 OAuth 토큰을 추가합니다. 자세한 내용은 GitHub와 CodeDeploy 통합 단원을 참조하십시오.

GitHub OAuth 토큰 최대 수 초과

CodeDeploy 배포를 생성할 때 허용되는 GitHub 토큰의 최대 수는 10개입니다. GitHub OAuth 토큰에 대한 오류가 발생하면 토큰 수가 10개 이하인지 확인하세요. 토큰이 10개가 넘으면 가장 오래된 토큰이 무효해집니다. 예를 들어 토큰이 11개이면 제일 처음에 만든 토큰이 무효해집니다. 토큰이 12개이면 처음에 만든 토큰 2개가 무효해집니다. CodeDeploy API를 사용하여 오래 된 토큰을 삭제하는 방법은 DeleteGitHubAccountToken을 참조하세요.

Amazon EC2 Auto Scaling 문제 해결

주제

- 일반적인 Amazon EC2 Auto Scaling 문제 해결
- “CodeDeployRole은 AmazonAutoScaling” 오류 AWS 서비스에서 작업을 수행할 수 있는 권한을 부여하지 않습니다.
- 개정 버전을 배포하기 전에 Amazon EC2 Auto Scaling 그룹의 인스턴스가 계속해서 프로비저닝 및 종료됨
- Amazon EC2 Auto Scaling 인스턴스를 종료 또는 재부팅하면 배포에 실패할 수 있음
- 단일 Amazon EC2 Auto Scaling 그룹으로 여러 배포 그룹 연결 피하기
- Amazon EC2 Auto Scaling 그룹의 EC2 인스턴스를 시작하지 못하고 "하트비트 제한 시간” 오류 발생

- 일치하지 않는 Amazon EC2 Auto Scaling 수명 주기 후크로 인해 Amazon EC2 Auto Scaling 그룹에 대한 자동 배포가 중지되거나 실패할 수 있습니다.
- "배포 그룹에 대한 인스턴스를 찾을 수 없어서 배포에 실패했습니다." 오류

일반적인 Amazon EC2 Auto Scaling 문제 해결

Amazon EC2 Auto Scaling 그룹에서 EC2 인스턴스로의 배포에 실패할 수 있는 원인은 다음과 같습니다.

- Amazon EC2 Auto Scaling은 EC2 인스턴스를 계속해서 시작하고 종료합니다. CodeDeploy가 애플리케이션 수정을 자동으로 배포할 수 없는 경우 Amazon EC2 Auto Scaling은 EC2 인스턴스를 계속해서 시작하고 종료합니다.

  CodeDeploy 배포 그룹에서 Amazon EC2 Auto Scaling 그룹을 연결 해제하거나 원하는 수의 인스턴스가 현재 인스턴스 개수와 일치하도록(따라서 Amazon EC2 Auto Scaling에서 EC2 인스턴스를 더 이상 시작할 수 없도록) Amazon EC2 Auto Scaling 그룹의 구성을 변경합니다. 자세한 내용은 CodeDeploy에서 배포 그룹 설정 변경 또는 Amazon EC2 Auto Scaling의 수동 확장을 참조하세요.
- CodeDeploy 에이전트가 응답하지 않습니다. EC2 인스턴스가 시작된 직후 실행되는 초기화 스크립트(예: cloud-init 스크립트)가 한 시간 이상 실행되는 경우 CodeDeploy 에이전트가 설치되지 않았을 수 있습니다. CodeDeploy는 CodeDeploy 에이전트가 보류 중인 배포에 대한 응답에 대해 1시간의 제한 시간을 둡니다. 이 문제를 해결하려면 초기화 스크립트를 CodeDeploy 애플리케이션 수정버전으로 합니다.
- Amazon EC2 Auto Scaling 그룹의 EC2 인스턴스가 배포 중에 재부팅됩니다. EC2 인스턴스가 배포 중 재부팅되거나 배포 명령 처리 중 CodeDeploy 에이전트가 종료되면 배포에 실패할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 인스턴스를 종료 또는 재부팅하면 배포에 실패할 수 있음 단원을 참조하십시오.
- 여러 애플리케이션 개정 버전이 Amazon EC2 Auto Scaling 그룹의 동일한 EC2 인스턴스에 동시에 배포됩니다. 배포 중 하나에 실행하는 데 몇 분 이상 걸리는 스크립트가 있는 경우 Amazon EC2 Auto Scaling 그룹의 동일한 EC2 인스턴스로 여러 애플리케이션 개정 버전 배포에 실패할 수 있습니다. 여러 애플리케이션 개정 버전을 Amazon EC2 Auto Scaling 그룹의 동일한 EC2 인스턴스에 배포하지 마세요.
- Amazon EC2 Auto Scaling 그룹의 일부로 실행된 새 EC2 인스턴스에 대한 배포에 실패합니다. 이 시나리오에서 배포 스크립트를 실행하면 Amazon EC2 Auto Scaling 그룹에서 EC2 인스턴스가 시작되지 않을 수 있습니다. (Amazon EC2 Auto Scaling 그룹의 다른 EC2 인스턴스는 정상적으로 실행되는 것처럼 보일 수 있습니다.) 이 문제를 해결하려면 먼저 다른 모든 스크립트가 완벽한지 확인해야 합니다.

- CodeDeploy 에이전트는 AMI에 포함되지 않습니다. 명령을 사용하여 새 인스턴스를 시작하는 동안 CodeDeploy 에이전트를 cfn-init 설치하는 경우 CloudFormation 템플릿의 `cfn-init` 섹션 끝에 에이전트 설치 스크립트를 배치합니다.
- CodeDeploy 에이전트가 AMI에 포함됨: 인스턴스 생성 시 에이전트의 상태가 `Stopped`가 되도록 AMI를 구성한 다음 `cfn-init` 스크립트 라이브러리의 최종 단계로 에이전트를 시작하는 스크립트를 포함합니다.

"CodeDeployRole은 AmazonAutoScaling" 오류 AWS 서비스에서 작업을 수행할 수 있는 권한을 부여하지 않습니다.

시작 템플릿으로 생성된 Auto Scaling 그룹을 사용하는 배포는 다음 권한을 필요로 합니다. 이는 `AWSCodeDeployRole` AWS 관리형 정책에서 부여한 권한에 추가됩니다.

- `EC2:RunInstances`
- `EC2:CreateTags`
- `iam:PassRole`

이러한 권한이 누락된 경우이 오류가 발생할 수 있습니다. 자세한 내용은 튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포, Auto Scaling 그룹에 대한 시작 템플릿 생성 및 Amazon EC2 Auto Scaling 사용 설명서의 권한을 참조하세요.

개정 버전을 배포하기 전에 Amazon EC2 Auto Scaling 그룹의 인스턴스가 계속해서 프로비저닝 및 종료됨

오류로 인해 Amazon EC2 Auto Scaling 그룹의 새로 프로비저닝된 인스턴스로 성공적인 배포에 실패할 수 있습니다. 그 결과, 정상 인스턴스와 성공한 인스턴스가 없습니다. 배포를 실행하거나 성공적으로 완료할 수 없기 때문에 인스턴스가 생성 직후에 종료됩니다. 그런 다음 Amazon EC2 Auto Scaling 그룹 구성으로 인해 최소 정상 호스트 요구 사항을 충족하기 위한 또 다른 인스턴스 배치가 프로비저닝됩니다. 이 배치 역시 종료되고 이러한 주기가 계속 반복됩니다.

가능한 원인은 다음과 같습니다.

- Amazon EC2 Auto Scaling 그룹 상태 확인에 실패했습니다.
- 애플리케이션 수정의 오류

이 문제를 해결하려면 다음 단계를 수행하세요.

1. Amazon EC2 Auto Scaling 그룹에 속하지 않는 EC2 인스턴스를 수동으로 생성합니다. 고유한 EC2 인스턴스 태그로 인스턴스에 태그를 지정합니다.
2. 새 인스턴스를 영향을 받는 배포 그룹에 추가합니다.
3. 배포 그룹에 오류가 없는 새 애플리케이션 수정을 배포합니다.

Amazon EC2 Auto Scaling 그룹의 이후 인스턴스에 애플리케이션 수정 버전을 배포하라는 메시지를 Amazon EC2 Auto Scaling 그룹에 표시합니다.

> **Note**
>
> 배포가 성공했는지 확인한 후 AWS 계정에 지속적인 요금이 부과되지 않도록 생성한 인스턴스를 삭제합니다.

Amazon EC2 Auto Scaling 인스턴스를 종료 또는 재부팅하면 배포에 실패할 수 있음

EC2 인스턴스가 Amazon EC2 Auto Scaling을 통해 시작된 다음에 인스턴스가 종료 또는 재부팅된 경우, 다음과 같은 이유로 해당 인스턴스에 대한 배포가 실패할 수 있습니다.

- 진행 중인 배포에서 축소 이벤트 또는 기타 모든 종료 이벤트로 인해 인스턴스가 Amazon EC2 Auto Scaling 그룹에서 분리된 다음 종료됩니다. 배포를 완료할 수 없기 때문에 배포에 실패합니다.
- 인스턴스가 재부팅되지만 시작되는 데에는 5분 이상 걸립니다. CodeDeploy는 이것을 시간 초과로 처리합니다. 이 서비스에서 인스턴스에 대한 현재 및 이후 배포에 모두 실패합니다.

이 문제를 해결하려면:

- 일반적으로, 인스턴스가 종료 또는 재부팅되기 전에 모든 배포가 완료되어야 합니다. 일반적으로, 인스턴스가 시작 또는 재부팅된 후 모든 배포가 시작됩니다.
- Amazon EC2 Auto Scaling 구성에 대해 Windows Server 기본 Amazon Machine Image(AMI)를 지정하고 EC2Config 서비스를 사용해 인스턴스의 컴퓨터 이름을 설정하는 경우 배포에 실패할 수 있습니다. 이 문제를 해결하려면 Windows Server 기본 AMI에서 EC2 서비스 속성(Ec2 Service Properties)의 일반(General) 탭에서 컴퓨터 이름 설정(Set Computer Name)을 삭제합니다. 이 확인란을 선택 해제하면 해당 Windows Server 기본 AMI로 시작된 모든 새 WindowsServer Amazon EC2 Auto Scaling 인스턴스에 대해 이러한 동작이 비활성화됩니다. 이 동작이 비활성화된

WindowsServer Amazon EC2 Auto Scaling 인스턴스의 경우에는 이 확인란의 선택을 취소할 필요가 없습니다. 인스턴스가 재부팅된 후 실패한 배포를 인스턴스에 다시 부팅하기만 하면 됩니다.

단일 Amazon EC2 Auto Scaling 그룹으로 여러 배포 그룹 연결 피하기

모범 사례에 따라, 각 Amazon EC2 Auto Scaling 그룹에 배포 그룹을 하나만 연결하는 것이 좋습니다.

이는 Amazon EC2 Auto Scaling이 여러 배포 그룹과 연결된 후크를 보유한 인스턴스를 확장하는 경우 모든 후크에 대한 알림을 한 번에 보내기 때문입니다. 이로 인해 각 인스턴스에 대한 여러 배포가 동시에 시작됩니다. 여러 배포가 명령을 CodeDeploy 에이전트에 동시에 전달하면, 수명 주기 이벤트와 배포 시작 또는 이전 수명 주기 이벤트 종료 간의 5분 시간제한에 도달할 수 있습니다. 이렇게 되면 배포 프로세스가 예상대로 실행 중이더라도 배포에 실패하게 됩니다.

> **Note**
>
> 수명 주기 이벤트의 스크립트에 대한 기본 제한 시간은 30분입니다. AppSpec 파일에서 제한 시간을 다른 값으로 변경할 수 있습니다. 자세한 내용은 EC2 온프레미스 배포용 AppSpec 파일 추가 단원을 참조하십시오.

동시에 두 개 이상의 배포 시도가 실행되는 경우 배포가 수행되는 순서를 제어할 수 없습니다.

마지막으로 인스턴스에 대한 배포에 실패하면 Amazon EC2 Auto Scaling은 즉시 해당 인스턴스를 종료합니다. 첫 번째 인스턴스가 종료되면 나머지 실행 중인 배포가 실패로 중단되기 시작합니다. CodeDeploy는 CodeDeploy 에이전트가 대기 중인 배포에 대해 응답하는데 1시간의 제한 시간을 두고 있기 때문에 각 인스턴스에서 제한 시간 초과가 발생하는 데 최대 60분이 걸릴 수 있습니다.

Amazon EC2 Auto Scaling에 대한 자세한 내용은 세부 정보: CodeDeploy 및 Amazon EC2 Auto Scaling 통합을 참조하세요.

Amazon EC2 Auto Scaling 그룹의 EC2 인스턴스를 시작하지 못하고 "하트비트 제한 시간" 오류 발생

Amazon EC2 Auto Scaling 그룹이 새 EC2 인스턴스를 시작하지 못하고 다음과 유사한 메시지가 생성될 수 있습니다.

```
Launching a new EC2 instance <instance-Id>. Status Reason: Instance failed
to complete user's Lifecycle Action: Lifecycle Action with token<token-Id>
was abandoned: Heartbeat Timeout.
```

이 메시지는 일반적으로 다음 중 하나를 나타냅니다.

- AWS 계정과 연결된 최대 동시 배포 수에 도달했습니다. 배포 한도에 대한 자세한 내용은 CodeDeploy 할당량 단원을 참조하세요.
- Auto Scaling 그룹이 너무 많은 EC2 인스턴스를 너무 빨리 시작하려고 했습니다. 각 새 인스턴스에 대한 RecordLifecycleActionHeartbeat 또는 CompleteLifecycleAction API 호출이 제한되었습니다.
- CodeDeploy의 애플리케이션과 연결된 배포 그룹이 업데이트 또는 삭제되기 전에 이러한 애플리케이션이 삭제되었습니다.

  애플리케이션 또는 배포 그룹을 삭제하면 CodeDeploy는 해당 애플리케이션 또는 배포 그룹과 연결된 Amazon EC2 Auto Scaling 후크를 모두 정리하려고 하지만 일부 후크가 남아 있을 수 있습니다. 명령을 실행하여 배포 그룹을 삭제하는 경우 남아있는 후크가 출력으로 반환됩니다. 하지만 명령을 실행하여 애플리케이션을 삭제할 경우에는 남아있는 후크가 출력 화면에 표시되지 않습니다.

  따라서 애플리케이션을 삭제하기 전에 애플리케이션과 연결된 배포 그룹을 모두 삭제하는 것이 모범 사례입니다. 명령으로 출력된 결과를 보고 수동으로 삭제해야 하는 수명 주기 후크를 식별할 수 있습니다.

"하트비트 제한 시간" 오류 메시지가 표시되면 남은 수명 주기 후크가 원인인지 확인하고 다음을 수행하여 문제를 해결할 수 있습니다.

1. 다음 중 하나를 수행하세요.

   - delete-deployment-group 명령을 호출하여 하트비트 제한 시간 초과를 일으키는 Auto Scaling 그룹과 연결된 배포 그룹을 삭제합니다.
   - null이 아닌 빈 Auto Scaling 그룹 이름 목록을 사용하여 update-deployment-group 명령을 호출해 모든 CodeDeploy-관리형 Auto Scaling 수명 주기 후크를 분리합니다.

     예를 들어 다음 AWS CLI 명령을 입력합니다.

     ```
     aws deploy update-deployment-group --application-name my-example-app
     --current-deployment-group-name my-deployment-group --auto-scaling-
     groups
     ```

     또 다른 예로 Java와 함께 CodeDeploy API를 사용하는 경우 `UpdateDeploymentGroup`을 호출하고 `autoScalingGroups`를 `new ArrayList<String>()`으로 설정합니다. 이것은 `autoScalingGroups`를 빈 목록으로 설정하고 기존 목록을 제거합니다. 기본값으로 `null`을

사용하지 마세요. 이렇게 하면 `autoScalingGroups`가 원하는 대로 유지되지 않기 때문입니다.

호출의 출력을 검사합니다. 출력에 Amazon EC2 Auto Scaling 수명 주기 후크 목록이 포함된 `hooksNotCleanedUp` 구조가 있는 경우, 남아 있는 수명 주기 후크가 있습니다.

2. describe-lifecycle-hooks 명령을 호출하여, 시작에 실패하는 EC2 인스턴스에 연결된 Amazon EC2 Auto Scaling 그룹의 이름을 지정합니다. 출력에서 다음 중 하나를 찾습니다.
   - 1단계에서 식별한 `hooksNotCleanedUp` 구조에 해당하는 Amazon EC2 Auto Scaling 수명 주기 후크 이름.
   - Auto Scaling 그룹과 연결된 배포 그룹의 이름을 포함하는 Amazon EC2 Auto Scaling 수명 주기 후크 이름.
   - CodeDeploy 배포에 대한 하트비트 제한 시간을 발생시켰을 수 있는 Amazon EC2 Auto Scaling 수명 주기 후크 이름.
3. 후크가 2단계에서 나열된 범주 중 하나에 속하는 경우 delete-lifecycle-hook 명령을 사용하여 삭제합니다. 호출에서 Amazon EC2 Auto Scaling 그룹 및 수명 주기 후크를 지정합니다.

> **Important**
>
> 2단계에서 설명한 대로 문제를 일으키는 후크만 삭제합니다. 실행 가능한 후크를 삭제하면 배포에 실패하거나 CodeDeploy가 확장된 EC2 인스턴스에 애플리케이션 개정 버전을 배포하지 못할 수 있습니다.

4. 원하는 Auto Scaling 그룹 이름으로 update-deployment-group 또는 create-deployment-group 명령을 호출합니다. CodeDeploy는 새로운 UUID로 Auto Scaling 후크를 다시 설치합니다.

> **Note**
>
> CodeDeploy 배포 그룹에서 Auto Scaling 그룹을 분리하면 Auto Scaling 그룹에 대한 진행 중인 배포에 실패할 수 있으며 Auto Scaling 그룹에 의해 확장된 새 EC2 인스턴스는 CodeDeploy에서 애플리케이션 개정 버전을 받지 못합니다. CodeDeploy에서 Auto Scaling을 다시 사용하려면 Auto Scaling 그룹을 배포 그룹에 다시 연결하고 새 `CreateDeployment`를 호출하여 플릿 전체 배포를 시작할 수 있습니다.

일치하지 않는 Amazon EC2 Auto Scaling 수명 주기 후크로 인해 Amazon EC2 Auto Scaling 그룹에 대한 자동 배포가 중지되거나 실패할 수 있습니다.

Amazon EC2 Auto Scaling 및 CodeDeploy는 Amazon EC2 Auto Scaling 그룹에서 시작된 수명 주기 후크를 사용하여 어떤 애플리케이션 개정 버전을 어떤 EC2 인스턴스에 배포해야 할지 결정합니다. Amazon EC2 Auto Scaling 및 CodeDeploy에서 수명 주기 후크 및 이러한 후크에 대한 정보가 정확하게 일치하지 않는 경우 자동 배포가 중지 또는 실패할 수 있습니다.

Amazon EC2 Auto Scaling 그룹에 대한 배포에 실패하면 Amazon EC2 Auto Scaling 및 CodeDeploy의 수명 주기 후크 이름이 일치하는지 확인하세요. 그렇지 않은 경우 다음 AWS CLI 명령 호출을 사용합니다.

먼저, Amazon EC2 Auto Scaling 그룹 및 배포 그룹 둘 다에 대한 수명 주기 후크 이름 목록을 가져옵니다.

1. describe-lifecycle-hooks 명령을 호출하여, CodeDeploy의 배포 그룹에 연결된 Amazon EC2 Auto Scaling 그룹의 이름을 지정합니다. 출력의 `LifecycleHooks` 목록에서 각 `LifecycleHookName` 값을 기록합니다.
2. get-deployment-group 명령을 호출하여, Amazon EC2 Auto Scaling 그룹에 연결된 배포 그룹의 이름을 지정합니다. 출력의 `autoScalingGroups` 목록에서 Amazon EC2 Auto Scaling 그룹 이름에 일치하는 각 항목의 이름 값을 찾아서 해당되는 `hook` 값을 기록합니다.

이제 두 수명 주기 후크 이름 세트를 비교합니다. 문자 대 문자로 정확하게 일치하는 경우에는 후크 이름이 문제가 아닙니다. 이 단원의 다른 부분에서 설명한 기타 Amazon EC2 Auto Scaling 문제 해결 단계를 시도해 볼 수 있습니다.

그러나 두 세트의 수명 주기 후크 이름이 문자 대 문자로 정확하게 일치하지 않는 경우에는 다음을 수행하세요.

1. get-deployment-group 명령 출력에 없는 수명 주기 후크 이름이 describe-lifecycle-hooks 명령 출력에 있으면 다음과 같이 합니다.

   a. describe-lifecycle-hooks 명령 출력의 각 수명 주기 후크 이름에 대해 delete-lifecycle-hook 명령을 호출합니다.

   b. update-deployment-group 명령을 호출하여, 원래 Amazon EC2 Auto Scaling 그룹의 이름을 지정합니다. CodeDeploy는 Amazon EC2 Auto Scaling 그룹에 새로운 대체 수명 주기 후크를 생성하고 수명 주기 후크를 배포 그룹과 연결합니다. 새 인스턴스가 Amazon EC2 Auto Scaling 그룹에 추가되면 자동 배포가 다시 시작되어야 합니다.

2. describe-lifecycle-hooks 명령 출력에 없는 수명 주기 후크 이름이 get-deployment-group 명령 출력에 있으면 다음과 같이 합니다.

   a. update-deployment-group 명령을 호출하되, 원래 Amazon EC2 Auto Scaling 그룹의 이름은 지정하지 마세요.

   b. update-deployment-group 명령을 다시 호출하여, 이번에는 원래 Amazon EC2 Auto Scaling 그룹의 이름을 지정합니다. CodeDeploy는 Amazon EC2 Auto Scaling 그룹에 누락된 수명 주기 후크를 다시 생성합니다. 새 인스턴스가 Amazon EC2 Auto Scaling 그룹에 추가되면 자동 배포가 다시 시작되어야 합니다.

두 세트의 수명 주기 후크 이름이 문자 대 문자로 정확하게 일치된 후에는 애플리케이션 수정을 다시 배포해 봅니다. 하지만 Amazon EC2 Auto Scaling 그룹에 추가된 새 인스턴스에만 배포합니다. Amazon EC2 Auto Scaling 그룹에 이미 있는 인스턴스에는 자동으로 배포되지 않습니다.

"배포 그룹에 대한 인스턴스를 찾을 수 없어서 배포에 실패했습니다." 오류

다음 CodeDeploy 오류가 표시되면 이 섹션을 읽어 보세요.

```
The deployment failed because no instances were found for your deployment group. Check your deployment group settings to make sure the tags for your EC2 instances or Auto Scaling groups correctly identify the instances you want to deploy to, and then try again.
```

가능한 원인은 다음과 같습니다.

1. 배포 그룹 설정에는 EC2 인스턴스, 온프레미스 인스턴스 또는 올바르지 않은 Auto Scaling 그룹에 대한 태그가 포함됩니다. 이 문제를 해결하려면 태그가 올바른지 확인하고 애플리케이션을 다시 배포합니다.
2. 배포가 시작된 후 플릿이 확장되었습니다. 이 시나리오에서는 플릿에 `InService` 상태에서 정상 인스턴스가 표시되지만 위의 오류도 표시됩니다. 이 문제를 해결하려면 애플리케이션을 다시 배포합니다.
3. Auto Scaling 그룹에 `InService` 상태의 인스턴스가 없습니다. 이 시나리오에서 플릿 전체 배포를 수행하려고 할 경우 CodeDeploy에 하나 이상의 인스턴스가 `InService` 상태여야 하므로 위의 오류 메시지와 함께 배포에 실패합니다. `InService` 상태의 인스턴스가 없는 데는 여러 가지 이유가 있을 수 있습니다. 몇 가지 이유는 다음과 같습니다.
   - Auto Scaling 그룹 크기를 `0`으로 예약(또는 수동으로 구성)했습니다.

- Auto Scaling이 잘못된 EC2 인스턴스(예: EC2 인스턴스에 하드웨어 오류가 있음)를 감지하고 모두 취소하여 `InService` 상태의 인스턴스가 하나도 남지 않았습니다.
- `0`에서 `1`로 확장 이벤트 중 CodeDeploy가 가장 최근 배포 후에 비정상 상태가 된, 이전에 성공한 개정 버전(마지막으로 성공한 개정 버전)을 배포했습니다. 이로 인해 확장된 인스턴스로의 배포가 실패하고 그 결과 Auto Scaling이 인스턴스를 취소하고 `InService` 상태의 인스턴스가 남지 않았습니다.

  `InService` 상태의 인스턴스가 없는 경우, To troubleshoot the error if there are no instances in the InService state 절차에 설명된 대로 문제를 해결합니다.

InService 상태의 인스턴스가 없는 경우 오류를 해결하려면

1. Amazon EC2 콘솔에서 원하는 용량(Desired Capacity) 설정을 확인합니다. 값이 0이면 양수로 설정합니다. 인스턴스가 `InService` 상태가 되어 배포가 성공할 때까지 기다립니다. 문제를 해결했으면 이 문제 해결 절차의 나머지 단계를 건너뛸 수 있습니다. 원하는 용량(Desired Capacity) 설정에 대한 자세한 내용은 Auto Scaling 그룹에 용량 제한 설정의 Amazon EC2 Auto Scaling 사용 설명서를 참조하세요.
2. Auto Scaling이 원하는 용량을 충족하기 위해 새 EC2 인스턴스를 계속 시작하려고 시도하지만 확장을 수행할 수 없는 경우, 일반적으로 Auto Scaling 수명 주기 후크가 실패하기 때문입니다. 이 문제는 다음과 같이 해결합니다.

   a. 어떤 Auto Scaling 수명 주기 후크 이벤트가 실패하는지 확인하려면 Amazon EC2 Auto Scaling 사용 설명서의 Auto Scaling 그룹에 대한 크기 조정 활동 확인을 참조하세요.

   b. 실패한 후크의 이름이 `CodeDeploy-managed-automatic-launch-deployment-hook-`*`DEPLOYMENT_GROUP_NAME`*인 경우, CodeDeploy로 이동하여 배포 그룹을 찾은 다음 Auto Scaling에서 시작된 실패한 배포를 찾습니다. 그런 다음 배포에 실패한 이유를 조사합니다.

   c. 배포가 실패한 이유(예: CloudWatch 경보가 발생한 경우)를 이해하고 개정을 변경하지 않고 문제를 해결할 수 있다면 지금 바로 그렇게 합니다.

   d. 조사 후 CodeDeploy의 마지막으로 성공한 개정이 더 이상 정상이 아닌 것으로 확인되고 Auto Scaling 그룹에 정상 인스턴스가 없는 경우 배포 교착 상태 시나리오에 있는 것입니다. 이 문제를 해결하려면 Auto Scaling 그룹에서 CodeDeploy의 수명 주기 후크를 일시적으로 제거한 다음 후크를 다시 설치하고 새 (양호한) 개정을 다시 배포하여 잘못된 CodeDeploy 개정을 수정해야 합니다. 지침은 다음 섹션을 참조하세요.

      - To fix the deployment deadlock issue (CLI)

- To fix the deployment deadlock issue (console)

배포 교착 상태 문제를 해결하려면(CLI)

1. (선택 사항)이 문제를 해결하는 동안 예기치 않은 배포가 발생하지 않도록 CodeDeploy 오류를 유발하는 CI/CD 파이프라인을 차단합니다.
2. 현재 Auto Scaling DesiredCapacity 설정에 유의합니다.

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name ASG_NAME
   ```

   이 절차의 끝부분에서 이 숫자로 다시 크기를 조정해야 할 수도 있습니다.
3. Auto Scaling DesiredCapacity 설정을 1로 설정합니다. 원하는 용량이 1보다 큰 경우 시작할 수 있는 선택 사항입니다 이 설정을 1로 줄이면 인스턴스가 나중에 프로비저닝하고 배포하는 데 걸리는 시간이 단축되어 문제 해결 속도가 빨라집니다. Auto Scaling에서 원하는 용량이 원래대로 0으로 설정된 경우, 1로 늘려야 합니다. 이 변경은 필수입니다.

   aws autoscaling set-desired-capacity --auto-scaling-group-name *ASG_NAME* --desired-capacity 1

   > **Note**
   >
   > 이 절차의 나머지 단계에서는 DesiredCapacity를 1로 설정했다고 가정합니다.

   이때 Auto Scaling은 하나의 인스턴스로 크기를 조정하려고 시도합니다. 그런 다음 CodeDeploy가 추가한 후크가 여전히 존재하기 때문에 CodeDeploy가 배포를 시도하고, 이 배포가 실패하고, Auto Scaling이 인스턴스를 취소하고, Auto Scaling이 원하는 용량에 도달하기 위해 인스턴스를 다시 시작하려고 시도하고, 다시 실패합니다. 즉, 취소와 재실행이 반복되는 상태인 것입니다.
4. 배포 그룹에서 Auto Scaling 그룹의 등록을 취소합니다.

   > **Warning**
   >
   > 다음 명령은 소프트웨어가 없는 새 EC2 인스턴스를 시작합니다. 명령을 실행하기 전에 소프트웨어를 실행하지 않는 Auto Scaling `InService` 인스턴스가 허용되는지 확인합니다. 예를 들어, 인스턴스에 연결된 로드 밸런서가 소프트웨어 없이 이 호스트로 트래픽을 전송하지 않는지 확인합니다.

> **Important**
>
> 아래 표시된 CodeDeploy 명령을 사용하여 후크를 제거합니다. CodeDeploy에서 제거를 인식하지 못하므로 Auto Scaling 서비스를 통해 후크를 제거하지 마세요.

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups
```

이 명령을 실행하면 다음과 같은 문제가 발생합니다.

a. CodeDeploy가 배포 그룹에서 Auto Scaling 그룹의 등록을 취소합니다.

b. CodeDeploy가 Auto Scaling 그룹의 Auto Scaling 수명 주기 후크를 제거합니다.

c. 배포가 실패한 원인인 후크가 더 이상 존재하지 않으므로 Auto Scaling은 기존 EC2 인스턴스를 취소하고 즉시 새 인스턴스를 시작하여 원하는 용량으로 크기를 조정합니다. 새 인스턴스 곧 `InService` 상태로 변경됩니다. 새 인스턴스에는 소프트웨어가 포함되어 있지 않습니다.

5. EC2 인스턴스가 `InService` 상태가 될 때까지 기다립니다. 이를 확인하려면 다음 명령을 사용합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
ASG_NAME --query AutoScalingGroups[0].Instances[*].LifecycleState
```

6. EC2 인스턴스에 후크를 다시 추가합니다.

> **Important**
>
> 아래 표시된 CodeDeploy 명령을 사용하여 후크를 추가합니다. CodeDeploy에서 추가가 인식되지 않으므로 Auto Scaling 서비스를 사용하여 후크를 추가하지 마세요.

```
aws deploy update-deployment-group --application-name APPLICATION_NAME
--current-deployment-group-name DEPLOYMENT_GROUP_NAME --auto-scaling-
groups ASG_NAME
```

이 명령을 실행하면 다음과 같은 문제가 발생합니다.

a. CodeDeploy EC2 인스턴스에 Auto Scaling 수명 주기 후크를 다시 설치합니다.

b. CodeDeploy Auto Scaling 그룹을 배포 그룹에 다시 등록합니다.

7. 정상 상태이고 사용하려는 Amazon S3 또는 GitHub 개정판을 사용하여 플릿 전체 배포를 생성합니다.

   예를 들어 개정이 `httpd_app.zip` 객체 키를 사용하여 `my-revision-bucket`을 호출하는 Amazon S3 버킷의 .zip 파일인 경우 다음 명령을 입력합니다.

   ```
   aws deploy create-deployment --application-name APPLICATION_NAME
   --deployment-group-name DEPLOYMENT_GROUP_NAME --
   revision "revisionType=S3,s3Location={bucket=my-revision-
   bucket,bundleType=zip,key=httpd_app.zip}"
   ```

   이제 `InService` 인스턴스 1개가 Auto Scaling 그룹에 있으므로 배포는 정상 작동하며 더 이상 배포 그룹에 대한 인스턴스를 찾을 수 없어서 배포에 실패했습니다 오류가 표시되지 않습니다.

8. 다음은 이전에 Auto Scaling 그룹을 크기 조정한 경우, 배포가 성공한 후 원래 용량으로 다시 조정하는 방법입니다.

   ```
   aws autoscaling set-desired-capacity --auto-scaling-group-name ASG_NAME
   --desired-capacity ORIGINAL_CAPACITY
   ```

배포 교착 상태 문제를 해결하려면 (콘솔)

1. (선택 사항)이 문제를 해결하는 동안 예기치 않은 배포가 발생하지 않도록 CodeDeploy 오류를 유발하는 CI/CD 파이프라인을 차단합니다.
2. Amazon EC2 콘솔로 이동하여 Auto Scaling 원하는 용량(Desired capacity) 설정을 확인합니다. 이 절차의 끝부분에서 이 숫자로 다시 크기를 조정해야 할 수도 있습니다. 이 설정을 찾는 방법에 대한 자세한 내용은 Auto Scaling 그룹의 용량 제한 설정을 참조하세요.
3. 원하는 EC2 인스턴스 수를 1로 설정합니다.

   원하는 용량이 처음부터 1보다 컸다면 이 설정은 선택 사항입니다. 이 설정을 1로 줄이면 인스턴스가 나중에 프로비저닝하고 배포하는 데 걸리는 시간이 단축되어 문제 해결 속도가 빨라집니다. Auto Scaling에 대해 원하는 용량(Desired capacity)이 원래 0으로 설정되어 있었다면 1로 늘려야 합니다. 이 변경은 필수입니다.

> **Note**
>
> 이 절차의 나머지 단계에서는 원하는 용량(Desired capacity)을 1로 설정했다고 가정합니다.

a. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.

b. 해당 리전을 선택합니다.

c. 문제가 있는 Auto Scaling 그룹으로 이동합니다.

d. 일반 세부 정보(General details)에서 편집(Edit)을 선택합니다.

e. 원하는 용량(Desired capacity)을 **1**로 설정합니다.

f. 업데이트를 선택합니다.

4. 배포 그룹에서 Auto Scaling 그룹의 등록을 취소합니다.

> **Warning**
>
> 다음 하위 단계에서는 소프트웨어가 없는 새 EC2 인스턴스를 시작합니다. 명령을 실행하기 전에 소프트웨어를 실행하지 않는 Auto Scaling `InService` 인스턴스가 허용되는지 확인합니다. 예를 들어, 인스턴스에 연결된 로드 밸런서가 소프트웨어 없이 이 호스트로 트래픽을 전송하지 않는지 확인합니다.

a. https://console.aws.amazon.com/codedeploy/에서 CodeDeploy 콘솔을 엽니다.

b. 해당 리전을 선택합니다.

c. 탐색 창에서 애플리케이션을 선택합니다.

d. CodeDeploy 애플리케이션 이름을 선택합니다.

e. CodeDeploy 배포 그룹의 이름을 선택합니다.

f. 편집을 선택합니다.

g. 환경 구성(Environment configuration)에서 Amazon EC2 Auto Scaling 그룹(Amazon EC2 Auto Scaling groups)을 선택 해제합니다.

> **Note**
>
> 환경 구성이 정의되어 있지 않으면 콘솔에서 구성을 저장할 수 없습니다. 확인을 우회하려면 호스트로 확인되지 않는 `EC2` 또는 `On-premises` 태그를 임시로 추가하세요. 태그를 추가하려면 Amazon EC2 인스턴스(Amazon EC2 instances)또는온프레미스 인스턴스(On-premises instance)를 선택하고 **`EC2`** 또는 **`On-premises`**의 태그 Key(키)를 추가하세요. 태그 값(Value)은 비워 둘 수 있습니다.

h. 변경 사항 저장을 선택합니다.

이러한 하위 단계를 완료한 후 다음과 같은 상황이 발생합니다.

i. CodeDeploy가 배포 그룹에서 Auto Scaling 그룹의 등록을 취소합니다.

ii. CodeDeploy가 Auto Scaling 그룹의 Auto Scaling 수명 주기 후크를 제거합니다.

iii. 배포가 실패한 원인인 후크가 더 이상 존재하지 않으므로 Auto Scaling은 기존 EC2 인스턴스를 취소하고 즉시 새 인스턴스를 시작하여 원하는 용량으로 크기를 조정합니다. 새 인스턴스 곧 `InService` 상태로 변경됩니다. 새 인스턴스에는 소프트웨어가 포함되어 있지 않습니다.

5. EC2 인스턴스가 `InService` 상태가 될 때까지 기다립니다. 상태를 확인하는 방법은 다음과 같습니다.

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

b. 탐색 창에서 Auto Scaling 그룹을 선택합니다.

c. 사용자의 Auto Scaling 그룹을 선택합니다.

d. 콘텐츠 창에서 인스턴스 관리(Instance Management) 탭을 선택합니다.

e. 인스턴스(Instances)에서 인스턴스 옆에 있는 수명 주기(Lifecycle) 열이 InService라고 표시되는지 확인합니다.

6. Auto Scaling 그룹을 등록 취소했을 때 사용한 방법으로 CodeDeploy 배포 그룹에 다시 등록합니다.

a. https://console.aws.amazon.com/codedeploy/에서 CodeDeploy 콘솔을 엽니다.

b. 해당 리전을 선택합니다.

c. 탐색 창에서 애플리케이션을 선택합니다.

d. CodeDeploy 애플리케이션 이름을 선택합니다.

e. CodeDeploy 배포 그룹의 이름을 선택합니다.

f. 편집을 선택합니다.

g. 환경 구성(Environment configuration)에서 Amazon EC2 Auto Scaling 그룹(Amazon EC2 Auto Scaling groups)을 선택하고 목록에서 사용자의 Auto Scaling 그룹을 선택합니다.

h. Amazon EC2 인스턴스(Amazon EC2 instances) 또는 온프레미스 인스턴스(On-premises instances)에서 추가한 태그를 찾아 제거합니다.

i. Amazon EC2 인스턴스(Amazon EC2 instances) 또는 온프레미스 인스턴스(On-premises instances) 옆에 있는 확인란을 선택 취소합니다.

j. 변경 사항 저장을 선택합니다.

이 구성으로 Auto Scaling 그룹에 수명 주기 후크를 다시 설치합니다.

7. 정상 상태이고 사용하려는 Amazon S3 또는 GitHub 개정판을 사용하여 플릿 전체 배포를 생성합니다.

예를 들어 개정이 `httpd_app.zip` 객체 키를 사용하여 `my-revision-bucket`을 호출하는 Amazon S3 버킷의 .zip 파일인 경우 다음을 수행합니다.

a. CodeDeploy 콘솔의 배포 그룹(Deployment Group) 페이지에서 배포 생성(Create deployment)을 선택합니다.

b. Revision type(수정 유형)에서 My application is stored in Amazon S3(내 애플리케이션은 Amazon S3에 저장됨)를 선택합니다.

c. 개정 위치(Revision location)는 `s3://my-revision-bucket/httpd_app.zip`을 선택합니다.

d. 개정 파일 형식(Revision file type)은 `.zip`을 선택합니다.

e. 배포 만들기를 선택합니다.

이제 `InService` 인스턴스 1개가 Auto Scaling 그룹에 있으므로 배포가 정상 작동하며 더 이상 “배포 그룹에 대한 인스턴스를 찾을 수 없어서 배포에 실패했습니다.” 오류가 표시되지 않습니다.

8. 다음은 이전에 Auto Scaling 그룹을 크기 조정한 경우, 배포가 성공한 후 원래 용량으로 다시 조정하는 방법입니다.

a. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.

b. 해당 리전을 선택합니다.

c. Auto Scaling 그룹으로 이동합니다.

d. 일반 세부 정보(General details)에서 편집(Edit)을 선택합니다.
e. 원하는 용량(Desired capacity)을 원래 값으로 다시 설정합니다.
f. 업데이트를 선택합니다.

에 대한 오류 코드 AWS CodeDeploy

이 단원에서는 CodeDeploy 오류에 대한 참조 정보를 제공합니다.

오류 코드	설명
AGENT_ISSUE	CodeDeploy 에이전트의 문제로 인해 배포에 실패했습니다. 에이전트가 설치되어 이 배포 그룹의 모든 인스턴스에서 실행 중인지 확인하세요. 자세히 알아보기: • CodeDeploy 에이전트가 실행 중인지 확인 • CodeDeploy 에이전트 설치 • CodeDeploy 에이전트 작업
AUTO_SCALING_IAM_ROLE_PERMISSIONS	배포 그룹과 연결된 서비스 역할에 AWS 서비스에서 작업을 수행하는 데 필요한 권한이 없습니다. 자세히 알아보기: • 2단계: CodeDeploy에 대한 서비스 역할 생성 • 역할을 생성하여 AWS 서비스에 대한 권한 위임
HEALTH_CONSTRAINTS	너무 많은 개별 인스턴스가 배포에 실패했거나, 배포에 사용할 수 있는 정상 인스턴스가 너무 적거나, 배포 그룹의 일부 인스턴스에 문제가 발생하여 전체 배포가 실패했습니다. 자세히 알아보기: • Instance Health

오류 코드	설명
	• 인스턴스 문제 해결 • EC2/온프레미스 배포 문제 해결
HEALTH_CONSTRAINTS_INVALID	배포 구성에서 정의한 정상 인스턴스의 최소 수를 사용할 수 없으므로 배포를 시작할 수 없습니다. 배포 구성을 업데이트하여 필요한 정상 인스턴스 수를 줄이거나 이 배포 그룹의 인스턴스 수를 늘릴 수 있습니다. 자세히 알아보기: • Instance Health • CodeDeploy용 인스턴스 작업
IAM_ROLE_MISSING	배포 그룹에 대해 지정된 서비스 역할 이름을 가진 서비스 역할이 없으므로 배포에 실패했습니다. 올바른 서비스 역할 이름을 사용 중인지 확인하세요. 자세히 알아보기: • 2단계: CodeDeploy에 대한 서비스 역할 생성 • CodeDeploy에서 배포 그룹 설정 변경
IAM_ROLE_PERMISSIONS	CodeDeploy에는 역할을 수임하는 데 필요한 권한이 없거나 사용 중인 IAM 역할은 AWS 서비스에서 작업을 수행할 수 있는 권한을 부여하지 않습니다. 자세히 알아보기: • 1단계: 설정 • 2단계: CodeDeploy에 대한 서비스 역할 생성 • 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기

오류 코드	설명
NO_INSTANCES	이는 다음 중 하나로 인해 발생할 수 있습니다. • EC2/온프레미스 blue/green 배포의 경우 Amazon EC2 태그를 사용하면 제대로 구성되지 않을 수 있습니다. CodeDeploy 배포 그룹에서 이들이 blue 인스턴스와 green 인스턴스에 포함되어 있는지 확인합니다. Amazon EC2 콘솔을 사용하여 인스턴스에 태그가 올바르게 지정되었는지 확인할 수 있습니다. • Amazon EC2 Auto Scaling 그룹을 사용하는 경우 Auto Scaling 그룹의 용량이 충분하지 않을 수 있습니다. Auto Scaling 그룹에 배포를 위한 충분한 용량이 있는지 확인합니다. Amazon EC2 콘솔을 사용해 정상 인스턴스 수를 확인하여 Amazon EC2 Auto Scaling 그룹의 용량을 확인할 수 있습니다. • EC2/온프레미스 blue/green 배포의 경우 blue 및 green 플릿의 크기가 같지 않을 수 있습니다. 두 플릿의 크기가 동일하도록 합니다. 자세히 알아보기: • Tagging Instances for Deployments • 튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포 • Blue/Green 배포를 위한 애플리케이션 생성(콘솔)

오류 코드	설명
OVER_MAX_INSTANCES	계정에 허용되는 것보다 더 많은 인스턴스가 배포 대상으로 지정되었기 때문에 배포에 실패했습니다. 이 배포의 대상 인스턴스 수를 줄이려면 이 배포 그룹에 대한 태그 설정을 업데이트하거나 대상 인스턴스 중 일부를 삭제합니다. 또는 AWS Support 에 문의하여 한도 증가를 요청할 수 있습니다. 자세히 알아보기: • CodeDeploy에서 배포 그룹 설정 변경 • CodeDeploy 할당량 • 한도 증가 요청
THROTTLED	IAM 역할에서 허용하는 것보다 많은 요청이 이루어졌기 때문에 배포 AWS CodeDeploy 에 실패했습니다. 요청 수를 줄이세요. 자세히 알아보기: • 쿼리 API 요청 속도
UNABLE_TO_SEND_ASG	배포 그룹이 Amazon EC2 Auto Scaling 그룹으로 올바르게 구성되지 않아 배포에 실패했습니다. CodeDeploy 콘솔에서 배포 그룹에서 Amazon EC2 Auto Scaling 그룹을 삭제한 다음 다시 추가합니다. 자세히 알아보기: • 세부 정보: CodeDeploy 및 Auto Scaling 통합

관련 주제

CodeDeploy 문제 해결

CodeDeploy 리소스

아래에 CodeDeploy를 이용할 때 참조할 수 있는 관련 리소스가 나와 있습니다.

참조 가이드 및 지원 리소스

- AWS CodeDeploy API 참조 - 공통 파라미터 및 오류 코드를 포함하여 CodeDeploy 작업 및 데이터 유형에 대한 설명, 구문 및 사용 예제입니다.
- CodeDeploy 기술 FAQ — CodeDeploy에 대해 고객들이 가장 궁금해 하는 내용
- AWS 지원 센터 - AWS Support 사례를 생성하고 관리하기 위한 허브입니다. 또한 포럼, 기술 FAQ, 서비스 상태 및 AWS Trusted Advisor등의 기타 자료에 대한 링크가 있습니다.
- AWS 지원 플랜 - AWS Support 플랜에 대한 정보를 제공하는 기본 웹 페이지입니다.
- 문의처 - AWS 결제, 계정, 이벤트, 침해 및 기타 문제에 대한 문의를 위한 중앙 연락 창구입니다.
- AWS 사이트 약관 - 저작권 및 상표, 계정, 라이선스 및 사이트 액세스, 기타 주제에 대한 자세한 정보입니다.

샘플

- GitHub의 CodeDeploy 샘플 — CodeDeploy에 대한 샘플 및 템플릿 시나리오.
- CodeDeploy Jenkins 플러그인 — CodeDeploy의 Jenkins 플러그인.
- CodeDeploy 에이전트 — CodeDeploy 에이전트의 오픈 소스 버전.

블로그

- AWS DevOps 블로그 - 개발자, 시스템 관리자 및 아키텍트를 위한 인사이트입니다.

AWS 소프트웨어 개발 키트 및 도구

다음 AWS SDKs 및 도구는 CodeDeploy를 사용한 솔루션 개발을 지원합니다.

- AWS SDK for .NET
- AWS SDK for Java
- AWS SDK for JavaScript

- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby
- AWS Toolkit for Eclipse - 파트 1, 2, 3.
- AWS Tools for Windows PowerShell - PowerShell 환경에서의 기능을 노출하는 Windows AWS SDK for .NET PowerShell cmdlet 세트입니다.
- AWS Tools for PowerShell의 CodeDeploy cmdlets — PowerShell 환경에서 CodeDeploy의 기능을 제공하는 Windows PowerShell cmdlet 세트.
- AWS Command Line Interface - AWS 서비스에 액세스하기 위한 균일한 명령줄 구문입니다. AWS CLI 는 단일 설정 프로세스를 사용하여 지원되는 모든 서비스에 대한 액세스를 활성화합니다.
- AWS 개발자 도구 - CodeDeploy 및를 사용하여 혁신적인 애플리케이션을 구축하는 데 도움이 되는 설명서, 코드 샘플, 릴리스 정보 및 기타 정보를 제공하는 개발자 도구 및 리소스에 대한 링크입니다 AWS.

문서 기록

다음 표에는 CodeDeploy 사용 설명서의 최신 릴리스 이후 새로운 기능 및 개선된 기능을 지원하기 위해 이 사용 설명서에 적용된 주요 변경 사항이 나와 있습니다.

- API 버전: 2014-10-06

변경 사항	설명	날짜
macOS를 지원하는 CodeDeploy 에이전트 v2.1.2 릴리스	AWS CodeDeploy 에이전트가 버전 2.1.2로 업데이트되어 EC2 Mac 인스턴스의 macOS에 대한 지원이 추가되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록 및 macOS용 CodeDeploy 에이전트 설치를 참조하세요.	2026년 10월 13일
CodeDeploy 에이전트 v2.1.1 릴리스	AWS CodeDeploy 에이전트가 버전 2.1.1로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2026년 10월 2일
CodeDeploy 에이전트 v2.1.0 릴리스	AWS CodeDeploy 에이전트가 버전 2.1.0으로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2026년 9월 7일
CodeDeploy 에이전트 v2.0.1 릴리스	AWS CodeDeploy 에이전트가 버전 2.0.1로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2026년 8월 24일

CodeDeploy 에이전트 v2.0.0 릴리스	AWS CodeDeploy 에이전트가 버전 2.0.0으로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2026년 7월 15일
CodeDeploy 에이전트 v1.8.1 릴리스	AWS CodeDeploy 에이전트가 버전 1.8.1로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2026년 2월 3일
CodeDeploy 에이전트 v1.8.0 릴리스	AWS CodeDeploy 에이전트가 버전 1.8.0으로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2025년 7월 31일
CodeDeploy에서 기존 AWS 관리형 정책 업데이트	`AWSCodeDeployDeployerAccess` 이(가) 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책 업데이트를 참조하세요.	2024년 12월 16일
CodeDeploy에서 기존 AWS 관리형 정책 업데이트	`AWSCodeDeployReadOnlyAccess` 이(가) 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책 업데이트를 참조하세요.	2024년 12월 16일
CodeDeploy에서 기존 AWS 관리형 정책 업데이트	`AWSCodeDeployFullAccess` 이(가) 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책 업데이트를 참조하세요.	2024년 12월 16일

CodeDeploy 에이전트 v1.7.1 릴리스	AWS CodeDeploy 에이전트가 버전 1.7.1로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2024년 11월 14일
Amazon S3 버킷 이름 업데이트	이 가이드의 Amazon S3 버킷 예제가에서 예약한 이름을 사용하도록 업데이트되었습니다 AWS.	2024년 6월 17일
대체 텍스트(alt 텍스트) 추가	이 가이드의 모든 이미지에 대체 텍스트를 포함하도록 업데이트했습니다. 대체 텍스트는 스크린 리더가 읽어주며, 시각 장애가 있는 사용자에게 설명서 접근성을 높여줍니다.	2024년 5월 22일
CodeDeploy 에이전트 v1.7.0 릴리스	AWS CodeDeploy 에이전트가 버전 1.7.0으로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2024년 3월 6일

명령 변경	`sudo service codedeploy-agent` *`status|start|stop`* 명령은 모범 사례인 CodeDeploy 에이전트 프로세스 관리에 `systemd`를 사용하지 않으므로 더 이상 권장되지 않습니다. `systemd`가 사용되도록 하려면 다음 예제에서와 같이 `systemctl` 명령을 사용하십시오(예: `sytemctl start codedeploy-agent`). 다음 항목은 `systemctl` 명령으로 업데이트되었습니다. Amazon Linux 또는 RHEL용 CodeDeploy 에이전트 설치, Ubuntu Server용 CodeDeploy 에이전트 설치, 모든 수명 주기 이벤트 건너뛴 오류 문제 해결, 실수로 삭제된 경우 CodeDeploy 로그 파일을 새로 만들기.	2024년 1월 12일
추가된 주제	CodeDeploy 에이전트 프로세스 관리 및 수명 주기 이벤트 스크립트의 파일 참조 주제가 추가되었습니다.	2024년 1월 12일
이제 CodeDeploy에서 영역 구성 지원	영역별 구성 정보로 CodeDeploy로 배포 구성 생성 주제를 업데이트했습니다.	2023년 12월 7일

이제 CodeDeploy에서 종료 배포 지원	Auto Scaling 스케일 인 이벤트 중 종료 배포 활성화 주제에 종료 배포 기능에 대한 설명이 추가되었습니다. 또한 EC2/온프레미스 배포를 위한 AppSpec '후크' 섹션, 현재 위치 배포를 위한 배포 그룹 생성(콘솔), EC2/온프레미스 블루/그린 배포를 위한 배포 그룹 생성(콘솔) 주제가 이 기능을 설명하도록 업데이트되었습니다.	2023년 12월 7일
JSON 형식 수정	AppSpec '리소스' 섹션(Amazon ECS 및 AWS Lambda 배포만 해당) 주제에서 JSON 코드 샘플의 형식을 수정했습니다.	2023년 12월 3일
문제 해결 주제 추가됨	Amazon ECS 배포 문제 해결 주제가 추가되었습니다.	2023년 10월 24일
AppSpec 파일 이름이 업데이트됨	EC2/온프레미스 배포를 위해 AppSpec 파일의 이름을 `appspec.yml` 로 지정해야 함을 나타내도록 CodeDeploy AppSpec 파일 참조를 업데이트했습니다.	2023년 10월 5일
이제 CodeDeploy에서 여러 로드 밸런서 지원	인플레이스 배포를 위한 배포 그룹 생성(콘솔), EC2/온프레미스 블루/그린 배포를 위한 배포 그룹 생성(콘솔), CodeDeploy Amazon EC2 배포를 위한 Elastic Load Balancing의 로드 밸런서 설정(여러 로드 밸런서 지원을 나타냄) 항목을 업데이트했습니다.	2023년 9월 26일

VPC의 리전 항목 업데이트	추가 리전 지원을 나타내도록 Amazon Virtual Private Cloud에서 CodeDeploy 사용 항목의 표를 업데이트했습니다. 특히 아시아 태평양(하이데라바드), 아시아 태평양(멜버른), 유럽(밀라노), 유럽(스페인), 유럽(취리히) 리전을 업데이트하여 에이전트 엔드포인트에 대한 지원을 나타냈습니다.	2023년 9월 22일
리소스 키트의 리전 항목 업데이트	AWS 리전별 리소스 키트 버킷 이름 섹션에 아시아 태평양(오사카), 아시아 태평양(하이데라바드), 캐나다(중부), 유럽(스페인), 유럽(취리히), 중동(UAE) 리전이 추가되었습니다. 또한 이러한 리전과 누락된 다른 리전을 포함하여 IAM 정책을 업데이트했습니다.	2023년 9월 22일
에이전트 설치 및 업데이트 항목 단축	Windows Server용 CodeDeploy 에이전트 설치 및 Windows Server용 CodeDeploy 에이전트 업데이트 항목을 간략하게 줄였습니다. 중복되는 Amazon S3 버킷 URL 및 Amazon S3 복사 명령을 제거했습니다.	2023년 9월 21일
아시아 태평양(자카르타) 리전 추가	리전별 리소스 키트 버킷 이름에 아시아 태평양(자카르타)를 추가했습니다.	2023년 9월 21일

CodeDeploy에서 기존 AWS 관리형 정책 업데이트	AWSCodeDeployRole 관리형 정책이 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책에 대한AWS 업데이트를 참조하세요.	2023년 8월 16일
한도 추가	CodeDeploy 제한 주제에 제한을 추가했습니다. 제한은 배포 그룹과 연결된 경보 최대 수입니다.	2023년 8월 15일
로드 밸런서와 관련된 단계 수정	EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔)의 지침을 수정했습니다. 이제 로드 밸런서 단계가 선택 사항으로 표시됩니다.	2023년 8월 3일
Amazon ECS 항목의 문구 명확하게 변경	튜토리얼: Amazon ECS에 애플리케이션 배포에서 문구를 명확하게 바꿨습니다. 이제 문구가 애플리케이션을 배포한다는 것을 나타냅니다. 이전에는 문구가 Amazon ECS 서비스를 배포한다는 것을 나타냈습니다.	2023년 8월 3일
이제 이스라엘(텔아비브) 리전에서 CodeDeploy 사용 가능	이제 이스라엘(텔아비브) 리전(il-central-1)에서 CodeDeploy를 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2023년 7월 31일

주제 업데이트	문제 해결 작업을 자동으로 수행하기 위해 런북을 사용하는 것과 관련한 팁을 추가하여 EC2/온프레미스 배포 문제 해결 주제를 업데이트했습니다.	2023년 7월 7일
주제 업데이트	작업 정의 ARN에 관한 정보를 추가하여 Amazon ECS 배포를 위한 AppSpec 'resources' 섹션 주제를 업데이트했습니다.	2023년 7월 7일
주제 업데이트	문제 해결 정보로 1단계: 온프레미스 인스턴스 AWS CLI 에 설치 및 구성 주제를 업데이트했습니다.	2023년 7월 7일
주제 업데이트	CloudFormation을 통한 Amazon ECS 블루/그린 배포에 대한 정보를 추가하여 교차 서비스 혼동된 대리자 예방 주제를 업데이트했습니다.	2023년 7월 6일
주제 업데이트	CloudFormation을 통한 Amazon ECS 블루/그린 배포에 대한 정보를 추가하여 교차 서비스 혼동된 대리자 예방 주제를 업데이트했습니다.	2023년 7월 6일
주제 업데이트	EC2/온프레미스 컴퓨팅 플랫폼에 대한 미리 정의된 배포 구성 주제를 업데이트했습니다. Auto Scaling 그룹을 사용한 `CodeDeployDefault.HalfAtATime` 사전 정의된 배포 구성의 동작에 참고 사항을 추가했습니다.	2023년 6월 29일

주제 업데이트	전송 계층 보안(TLS) 프로토콜의 새로운 최소 및 권장 버전을 나타내도록 인프라 보안 in AWS CodeDeploy 주제를 업데이트했습니다.	2023년 6월 28일
한도 업데이트	다음 한도가 변경되었습니다. 'EC2/온프레미스 인 플레이스(in-place) 배포를 실행할 수 있는 최대 시간'. 자세한 내용은 한도를 참조하세요.	2023년 6월 27일
주제 업데이트	3단계: CodeDeploy 사용자 권한 제한 주제가 세부 지침을 포함하여 업데이트되었습니다.	2023년 5월 31일
CodeDeploy에서 기존 AWS 관리형 정책 업데이트	AWSCodeDeployFullAccess 관리형 정책이 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책에 대한AWS 업데이트를 참조하세요.	2023년 5월 16일
CodeDeploy 에이전트 v1.6.0 릴리스	AWS CodeDeploy 에이전트가 버전 1.6.0으로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2023년 3월 30일
CodeDeploy 에이전트 v1.5.0 릴리스	AWS CodeDeploy 에이전트가 버전 1.5.0으로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2023년 3월 3일
Amazon ECS 컴퓨팅 플랫폼 업데이트	Amazon ECS 컴퓨팅 플랫폼에서의 배포가 이제 아시아 태평양(자카르타) 리전에서 지원됩니다.	2023년 2월 8일

CodeDeploy에서 기존 AWS 관리형 정책 업데이트	AWSCodeDeployRole 관리형 정책이 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책에 대한AWS 업데이트를 참조하세요.	2023년 2월 3일
주제 업데이트	Amazon Virtual Private Cloud에서 CodeDeploy 사용 주제가 새 리전과 변경된 AWS 리전으로 업데이트되었습니다.	2023년 2월 2일
주제 업데이트	이제 아시아 태평양(멜버른) 리전(ap-southeast-4)에서 CodeDeploy를 사용할 수 있습니다. CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 이러한 새 리전의 가용성을 반영하도록 업데이트되었습니다.	2023년 1월 26일
보안 모범 사례 업데이트	CodeDeploy 시작하기 섹션 및 기타 몇 가지 섹션이 AWS 보안 모범 사례에 맞게 업데이트되었습니다.	2023년 1월 23일
CodeDeploy 에이전트 v1.4.1 릴리스	AWS CodeDeploy 에이전트가 버전 1.4.1로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2022년 12월 6일

문제 해결 주제 추가됨	Windows용 CodeDeploy 에이전트에서 사용되는 긴 파일 경로로 인한 오류를 해결하는 방법에 대한 주제가 추가되었습니다. 자세한 내용은 파일 경로가 길면 “No such file or directory(해당 파일 또는 디렉터리가 없음)” 오류가 발생함을 참조하세요.	2022년 12월 6일
한도가 변경됨	' AWS 계정과 연결된 사용자 지정 배포 구성의 최대 수' 제한이 변경되었습니다. 한도는 이제 200개입니다. 한도에 대한 자세한 내용은 한도 주제를 참조하세요.	2022년 9월 7일
CodeDeploy 에이전트 v1.4.0 릴리스	AWS CodeDeploy 에이전트가 버전 1.4.0으로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2022년 8월 31일
몇 가지 한도가 수정됨	' AWS 계정과 연결된 최대 동시 배포 수'가 이제 1000개로 제한되었습니다. '단일 배포 내 인스턴스의 최대 수'가 이제 1,000입니다. '진행 중이고 한 계정과 연결된 동시 배포에 의해 사용될 수 있는 최대 인스턴스 개수'가 이제 1,000입니다. 이제 ' AWS 계정과 연결된 최대 사용자 지정 배포 구성 수'가 100개입니다. 한도에 대한 자세한 내용은 한도 주제를 참조하세요.	2022년 8월 8일

각 리전에서 지원되는 CodeDeploy 엔드포인트를 보여주는 테이블 추가	자세한 내용은 Amazon Virtual Private Cloud에서 CodeDeploy 사용을 참조하세요.	2022년 4월 20일
Amazon ECS 블루/그린 배포를 위한 새로운 제한 추가	Amazon ECS 블루/그린 배포 시, 개정 배포부터 대체 환경으로의 트래픽 이동까지의 최대 시간은 이제 120시간입니다. 자세한 내용은 제한 주제의 배포 단원을 참조하세요.	2022년 4월 12일
혼동된 대리자 문제를 방지하는 방법에 대한 주제 추가	자세한 내용은 AWS CodeDeploy용AWS Identity and Access Management를 참조하십시오.	2022년 3월 14일
CodeDeploy에서 기존 AWS 관리형 정책 업데이트	`AmazonEC2RoleforAWSCodeDeployLimited` 역할이 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책 업데이트를 참조하세요.	2021년 11월 22일
CodeDeploy에서 기존 AWS 관리형 정책 업데이트	`AWS CodeDeployRole` 이(가) 업데이트되었습니다. 자세한 내용은 AWS 관리형 정책 업데이트를 참조하세요.	2021년 5월 18일
CodeDeploy 에이전트 v1.3.2 릴리스	AWS CodeDeploy 에이전트가 버전 1.3.2로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2021년 5월 6일
CodeDeploy에서 오래된 Amazon EC2 인스턴스 업데이트 지원	CodeDeploy는 이제 오래된 Amazon EC2 인스턴스 업데이트를 자동으로 지원합니다. 자세한 내용은 배포 그룹에 대한 고급 옵션 구성을 참조하세요.	2021년 2월 23일

CodeDeploy 에이전트 v1.3.1 릴리스	AWS CodeDeploy 에이전트가 버전 1.3.1로 업데이트 되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2020년 12월 22일
CodeDeploy 에이전트 v1.3.0 릴리스	AWS CodeDeploy 에이전트가 버전 1.3.0으로 업데이트 되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2020년 11월 10일
CodeDeploy 에이전트 v1.2.1 릴리스	AWS CodeDeploy 에이전트가 버전 1.2.1로 업데이트 되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2020년 9월 23일
CodeDeploy는에서 제공하는 Amazon VPC 엔드포인트를 지원합니다. AWS PrivateLink	Amazon Virtual Private Cloud(VPC)를 사용하여 AWS 리소스를 호스팅하는 경우 VPC와 CodeDeploy 간에 프라이빗 연결을 설정할 수 있습니다. 이 연결을 사용하면 CodeDeploy가 퍼블릭 인터넷을 통하지 않고 VPC의 리소스와 통신하게 할 수 있습니다. 자세한 내용은 Amazon Virtual Private Cloud에서 CodeDeploy 사용을 참조하세요.	2020년 8월 11일

업데이트된 CodeDeploy 서비스 한도	계정당 애플리케이션 수와 애플리케이션당 배포 그룹 수 모두에 대한 제한을 1,000으로 업데이트했습니다. CodeDeploy 서비스 한도에 대한 자세한 내용은 CodeDeploy 한도를 참조하세요.	2020년 8월 6일
CodeDeploy 에이전트 v1.1.2 릴리스	AWS CodeDeploy 에이전트가 버전 1.1.2로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2020년 8월 4일
CodeDeploy 에이전트 1.1.0 릴리스 및 Amazon EC2 Systems Manager와 통합	이제 CodeDeploy 에이전트 버전 1.1.0을 사용할 수 있습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요. 이제 Amazon EC2 Systems Manager를 사용하여 Amazon EC2 또는 온프레미스 인스턴스에서 CodeDeploy 에이전트 설치 및 업데이트를 자동으로 관리할 수 있습니다. 자세한 내용은 .Amazon EC2 Systems Manager를 사용하여 CodeDeploy 에이전트 설치를 참조하세요.	2020년 6월 30일

CodeDeploy는를 사용하여 Amazon ECS 블루/그린 배포 관리를 지원합니다. CloudFormation	이제 CloudFormation 를 사용하여 CodeDeploy를 통해 Amazon ECS 블루/그린 배포를 관리할 수 있습니다. 그린 및 블루 리소스를 정의하고 CloudFormation에서 사용할 트래픽 라우팅 및 안정화 설정을 지정하여 배포를 생성합니다. 자세한 내용은 를 통한 Amazon ECS 블루/그린 배포 생성을 CloudFormation 참조하세요.	2020년 5월 19일
CodeDeploy에서 Amazon ECS 블루/그린 배포를 위한 가중치 기반 트래픽 이동 지원	이제 CodeDeploy는 Amazon ECS 블루/그린 배포를 위한 가중치 기반 트래픽 이동을 지원합니다. 배포 구성을 선택 또는 생성하여 배포 시 트래픽 전환 간격 수와 각 간격에서 전환할 트래픽 백분율을 지정할 수 있습니다. 이 변경 사항을 반영하여 다음 주제가 업데이트되었습니다. Amazon ECS 컴퓨팅 플랫폼의 배포 구성	2020년 2월 6일
업데이트된 보안, 인증 및 액세스 제어 주제	CodeDeploy에 대한 보안, 인증 및 액세스 제어 정보가 새로운 보안 장으로 구성되었습니다. 자세한 내용은 보안을 참조하세요.	2019년 11월 26일
CodeDeploy에서 알림 규칙 지원	이제 알림 규칙을 사용하여 사용자에게 배포의 중요한 변경 사항을 알릴 수 있습니다. 자세한 내용은 알림 규칙 생성을 참조하세요.	2019년 11월 5일

업데이트된 주제	이제 아시아 태평양(홍콩) 리전(ap-east-1) 리전에서 CodeDeploy를 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다. 이 리전에 대한 액세스를 명시적으로 활성화해야 합니다. 자세한 내용은 AWS 리전 관리를 참조하세요.	2019년 4월 25일
업데이트된 주제	AWS CodeDeploy 는 이제 Amazon ECS 서비스에서 컨테이너화된 애플리케이션의 블루/그린 배포를 지원합니다. 새 Amazon ECS 컴퓨팅 플랫폼을 사용하는 CodeDeploy 애플리케이션은 동일한 Amazon ECS 서비스의 새로운 대체 작업 세트에 컨테이너화된 애플리케이션을 배포합니다. AWS CodeDeploy 컴퓨팅 플랫폼 개요, Amazon ECS 컴퓨팅 플랫폼에 배포, Amazon ECS 배포를 위한 AppSpec 파일 구조, Amazon ECS 서비스 배포용 애플리케이션 생성(콘솔)을 포함한 여러 주제가 이 변경 사항을 반영하도록 추가 및 업데이트되었습니다.	2018년 11월 27일

업데이트된 CodeDeploy 에이전트	AWS CodeDeploy 에이전트가 버전 1.0.1.1597로 업데이트되었습니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2018년 11월 15일
콘솔 업데이트	이 안내서의 절차가 CodeDeploy 콘솔의 새 설계에 맞게 업데이트되었습니다.	2018년 10월 30일
지원되는 새로운 최소 버전의 CodeDeploy 에이전트	에이전트의 지원되는 최소 버전은 이제 1.8.x AWS CodeDeploy 입니다. 자세한 내용은 CodeDeploy 에이전트의 버전 기록을 참조하세요.	2018년 8월 7일

이전 업데이트

다음 표에서는 2018년 6월 이전 AWS CodeDeploy 사용 설명서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다.

변경	설명	변경 날짜
주제 업데이트	이제 유럽(파리) 리전(eu-west-3) 리전에서 CodeDeploy를 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2017년 12월 19일
업데이트된 주제	CodeDeploy는 이제 중국(닝샤) 지역에서 사용할 수 있습니다 중국(베이징) 리전 또는 중국(닝샤) 리전에서 서비스를 이용하려면 이들 리전에 대한 계정 및 자격 증명이 있어야 합니다. 다른 AWS 리전의 계정 및 자격 증명은 베이징 및 닝샤 리전에서는 작동하지 않으며 그 반대의 경우도 마찬가지입니다.	2017년 12월 11일

변경	설명	변경 날짜
	CodeDeploy 리소스 키트 버킷 이름 및 CodeDeploy 에이전트 설치 절차와 같은 중국 리전의 일부 리소스 관련 정보는 이 CodeDeploy 사용 설명서 에디션에 포함되어 있지 않습니다. 자세한 내용: • 중국(베이징) 리전 AWS 에서 시작하기의 CodeDeploy • 중국 리전의 CodeDeploy 사용 설명서(영어 버전 | 중국어 버전)	
업데이트된 주제	이제 CodeDeploy에서 Lambda 함수의 배포를 지원합니다. AWS Lambda 배포가 수신 트래픽을 기존 Lambda 함수에서 업데이트된 Lambda 함수 버전으로 전환합니다. 배포 구성을 선택하거나 생성하여 배포의 트래픽 이동 간격 수와 각 간격에서 이동할 트래픽 비율을 지정합니다. AWS Lambda 배포는 AWS 서버리스 애플리케이션 모델(AWS SAM)에서 지원되므로 AWS SAM 배포 기본 설정을 사용하여 AWS Lambda 배포 시 트래픽이 이동하는 방식을 관리할 수 있습니다. 여러 주제가 CodeDeploy 컴퓨팅 플랫폼 개요, AWS Lambda 컴퓨팅 플랫폼에 배포, AWS Lambda 컴퓨팅 플랫폼 배포 생성(콘솔), AWS Lambda 함수 배포를 위한 애플리케이션 생성(콘솔) 및 AWS Lambda 배포를 위한 AppSpec 파일 추가을 포함한 이 변경 사항을 반영하도록 추가 및 업데이트되었습니다.	2017년 11월 28일
새 주제	이제 CodeDeploy는 CodeDeploy 에이전트가 설치된 로컬 머신 또는 인스턴스로 바로 배포가 가능하도록 지원합니다. 로컬에서 배포를 테스트할 수 있을 뿐만 아니라 오류가 발생한 경우 CodeDeploy 에이전트 오류 로그를 사용해 디버깅을 실행할 수 있습니다. 또한 로컬 배포를 사용하여 애플리케이션 개정의 무결성 및 AppSpec의 내용 등을 테스트할 수 있습니다. 자세한 내용은 CodeDeploy 에이전트를 사용하여 로컬 시스템에서 배포 패키지의 유효성을 검사합니다. 섹션을 참조하세요.	2017년 11월 16일

변경	설명	변경 날짜
업데이트된 주제	배포 그룹의 Elastic Load Balancing 로드 밸런서에 대한 CodeDeploy 지원이 블루/그린 배포와 인 플레이스(in-place) 배포에 대해 모두 Network Load Balancers를 포함하도록 확장되었습니다. 이제 배포 그룹에 대해 Application Load Balancer, Classic Load Balancer 또는 Network Load Balancer를 선택할 수 있습니다. 로드 밸런서는 블루/그린 배포에는 필수이고, 인 플레이스(In-Place) 배포에는 선택 사항입니다. Integrating CodeDeploy with Elastic Load Balancing, 현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔), 배포 사전 조건, Integrating CodeDeploy with Elastic Load Balancing 및 현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔) 등의 여러 주제가 이 지원을 반영하도록 업데이트되었습니다.	2017년 9월 12일
업데이트된 주제	배포 그룹의 Elastic Load Balancing 로드 밸런서에 대한 CodeDeploy 지원이 블루/그린 배포와 인 플레이스(in-place) 배포에 대해 모두 Application Load Balancers를 포함하도록 확장되었습니다. 이제 배포 그룹에 대해 Application Load Balancer와 Classic Load Balancer 간에서 선택할 수 있습니다. 로드 밸런서는 블루/그린 배포에는 필수이고, 인 플레이스(In-Place) 배포에는 선택 사항입니다. Integrating CodeDeploy with Elastic Load Balancing, CodeDeploy를 사용하여 애플리케이션 생성 및 CodeDeploy에서 배포 그룹 만들기 등의 주제가 이 추가 지원을 반영하도록 업데이트되었습니다.	2017년 8월 10일

변경	설명	변경 날짜
신규 및 업데이트된 주제	CodeDeploy는 이제 여러 태그 그룹을 사용하여 배포 그룹에 포함될 인스턴스의 합집합과 교집합을 식별하는 것을 지원합니다. 한 태그 그룹을 사용하는 경우 그룹의 하나 이상의 태그로 식별되는 모든 인스턴스가 배포 그룹에 포함됩니다. 여러 태그 그룹을 사용하는 경우 각 태그 그룹의 하나 이상의 태그로 식별되는 인스턴스만 포함됩니다. 배포 그룹에 인스턴스를 추가하는 새로운 방법에 대한 자세한 내용은 Tagging Instances for Deployments 단원을 참조하세요. 이 지원을 반영하도록 업데이트된 다른 주제에는 현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔), Blue/Green 배포를 위한 애플리케이션 생성(콘솔), 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔), EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔), Deployments 및 튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포의 5단계: 애플리케이션 및 배포 그룹 만들기 등이 있습니다.	2017년 7월 31일
업데이트된 주제	Windows Server 인스턴스에 CodeDeploy 에이전트를 설치하는 두 가지 방법이 Windows Server용 CodeDeploy 에이전트 설치에 추가되었습니다. Windows PowerShell 명령 외에도 이제 직접 HTTPS 링크 및 Amazon S3 복사 명령을 사용하여 설치 파일을 다운로드할 수 있습니다. 파일을 인스턴스에 다운로드하거나 복사한 후 설치를 수동으로 실행할 수 있습니다.	2017년 7월 12일

변경	설명	변경 날짜
업데이트된 주제	CodeDeploy에서는 GitHub 계정 및 리포지토리에 대한 연결 관리 방법이 개선되었습니다. 이제 CodeDeploy 애플리케이션을 GitHub 리포지토리와 연결하기 위해 GitHub 계정에 대한 연결을 최대 25개까지 만들어 저장할 수 있습니다. 각 연결은 여러 리포지토리를 지원할 수 있습니다. 최대 25개의 다른 GitHub 계정에 대한 연결을 만들거나 단일 계정에 대해 두 개 이상의 연결을 만들 수 있습니다. GitHub 계정에 애플리케이션을 연결하면 CodeDeploy에서는 사용자의 추가 작업 없이 필수 액세스 권한을 관리합니다. 이러한 지원 내용을 반영하기 위해 GitHub 리포지토리에 저장된 개정 버전에 대한 정보 지정, GitHub와 CodeDeploy 통합 및 튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포 단원이 업데이트되었습니다.	2017년 5월 30일
업데이트된 주제	과거에는 CodeDeploy 에이전트가 대상 위치에서 최근에 성공적으로 배포한 애플리케이션 개정 버전의 일부가 아닌 파일을 감지하면 기본적으로 현재 배포에 실패했습니다. 이제 CodeDeploy는 에이전트가 이러한 파일을 처리하는 방법(배포 실패, 콘텐츠 보존 또는 콘텐츠 덮어쓰기)에 대한 옵션을 제공합니다.이 지원을 반영하도록 CodeDeploy에서 배포 만들기이(가) 업데이트되었으며 새 단원 기존 컨텐츠의 롤백 동작이(가) CodeDeploy를 사용하여 재배포 및 배포 롤백에 추가되었습니다.	2017년 5월 16일

변경	설명	변경 날짜
업데이트된 주제	이제 CodeDeploy 콘솔 또는 AWS CLI을 사용하여 Elastic Load Balancing의 Classic Load Balancer를 배포 그룹에 할당할 수 있습니다. 인 플레이스(In-Place) 배포 시 로드 밸런서는 배포 대상 인스턴스로 인터넷 트래픽이 라우팅되지 않도록 하고 해당 인스턴스에 대한 배포가 완료되면 인스턴스가 다시 트래픽을 처리할 수 있도록 합니다. 이러한 새로운 지원 내용을 반영하기 위해 다른 AWS 서비스와의 통합, Integrating CodeDeploy with Elastic Load Balancing, 현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔), 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 및 AppSpec 'hooks' 섹션 등 여러 주제가 업데이트되었습니다. 실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결이라는 새로운 단원이 문제 해결 안내서에 추가되었습니다.	2017년 4월 27일
업데이트된 주제	이제 CodeDeploy 콘솔 또는 AWS CLI을 사용하여 Elastic Load Balancing의 Classic Load Balancer를 배포 그룹에 할당할 수 있습니다. 인 플레이스(In-Place) 배포 시 로드 밸런서는 배포 대상 인스턴스로 인터넷 트래픽이 라우팅되지 않도록 하고 해당 인스턴스에 대한 배포가 완료되면 인스턴스가 다시 트래픽을 처리할 수 있도록 합니다. 이러한 새로운 지원 내용을 반영하기 위해 다른 AWS 서비스와의 통합, Integrating CodeDeploy with Elastic Load Balancing, 현재 위치(in-place) 배포를 위한 애플리케이션 생성(콘솔), 인 플레이스(in-place) 배포에 사용할 수 있는 배포 그룹 만들기(콘솔) 및 AppSpec 'hooks' 섹션 등 여러 주제가 업데이트되었습니다. 실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결이라는 새로운 단원이 문제 해결 안내서에 추가되었습니다.	2017년 5월 1일

변경	설명	변경 날짜
업데이트된 주제	이제 중국(베이징) 리전에서 CodeDeploy를 사용할 수 있습니다 중국(베이징) 리전 또는 중국(닝샤) 리전에서 서비스를 이용하려면 이들 리전에 대한 계정 및 자격 증명이 있어야 합니다. 다른 AWS 리전의 계정 및 자격 증명은 베이징 및 닝샤 리전에서는 작동하지 않으며 그 반대의 경우도 마찬가지입니다. CodeDeploy 리소스 키트 버킷 이름 및 CodeDeploy 에이전트 설치 절차와 같은 중국 리전의 일부 리소스 관련 정보는 이 CodeDeploy 사용 설명서 에디션에 포함되어 있지 않습니다. 자세한 내용: • 중국(베이징) 리전 AWS 에서 시작하기의 CodeDeploy • 중국 리전의 CodeDeploy 사용 설명서(영어 버전 | 중국어 버전)	2017년 3월 29일
신규 및 업데이트된 주제	배포 그룹(원본 환경)의 인스턴스가 다른 인스턴스 세트(대체 환경)로 대체되는 배포 방법인 블루/그린 배포에 대한 새로운 CodeDeploy 지원을 반영하기 위해 새로운 여러 가지 주제가 도입되었습니다. 블루/그린 배포 개요에서는 CodeDeploy에서 사용하는 블루/그린 방법에 대한 자세한 설명을 제공합니다. 추가되는 새 주제에는 Blue/Green 배포를 위한 애플리케이션 생성(콘솔), EC2/온프레미스 블루/그린 배포에 사용할 수 있는 배포 그룹 만들기(콘솔), Elastic Load Balancing에서 CodeDeploy Amazon EC2 배포에 대한 로드 밸런서 설정 단원이 포함됩니다. CodeDeploy에서 배포 만들기, CodeDeploy에서 배포 구성 작업, CodeDeploy를 사용하여 애플리케이션 생성, CodeDeploy에서 배포 그룹 작업, CodeDeploy에서 배포 작업 및 AppSpec 'hooks' 섹션 단원을 비롯하여 여러 가지 주제가 업데이트되었습니다.	2017년 1월 25일

변경	설명	변경 날짜
신규 및 업데이트된 주제	새로운 주제인 에서는를 통해 생성된 주기적으로 새로 고침된 임시 자격 증명을 사용하여 온프레미스 인스턴스를 인증하고 등록하는 방법을 register-on-premises-instance 명령(IAM 세션 ARN)을 사용하여 온프레미스 인스턴스를 등록합니다.설명합니다 AWS Security Token Service. 이 접근 방식은 각 인스턴스에서 정적 IAM 사용자의 자격 증명만 사용하는 것보다 온프레미스 인스턴스의 대규모 플릿을 더욱 원활하게 지원합니다. 또한 이러한 새로운 지원을 반영하기 위해 Working with On-Premises Instances 단원이 업데이트되었습니다.	2016년 28월 12일
업데이트된 주제	CodeDeploy는 이제 유럽(런던)(eu-west-2)에서 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2016년 12월 13일
업데이트된 주제	이제 캐나다(중부) 리전(ca-central-1)에서 CodeDeploy를 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2016년 12월 8일
업데이트된 주제	이제 미국 동부(오하이오) 리전(us-east-2)에서 CodeDeploy를 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2016년 10월 17일
새로운 주제	새 단원인 인증 및 액세스 제어는 AWS Identity and Access Management (IAM) 및 CodeDeploy를 사용하여 자격 증명의 사용을 통해 리소스에 대한 액세스를 보호할 수 있는 방법에 대한 포괄적인 정보를 제공합니다. 이러한 자격 증명은 Amazon S3 버킷에서 애플리케이션 개정을 검색하고 Amazon EC2 인스턴스에서 태그를 읽는 등 AWS 리소스에 액세스하는 데 필요한 권한을 제공합니다.	2016년 10월 11일

변경	설명	변경 날짜
업데이트된 주제	Windows Server용 새 CodeDeploy 에이전트 업데이트 프로그램의 사용 가능 여부를 반영하기 위해 Windows Server에서 CodeDeploy 에이전트 업데이트이(가) 업데이트되었습니다. Windows Server 인스턴스에 설치되면 업데이트 프로그램이 새 버전이 있는지 정기적으로 확인합니다. 이미 설치되어 있는 경우 새 버전이 감지되면 최신 버전을 설치하기 전에 업데이트 프로그램이 에이전트의 현재 버전을 제거합니다.	2016년 10월 4일
업데이트된 주제	이제 CodeDeploy는 Amazon CloudWatch 경보와 통합되어 여러 연속 기간에 지정된 경보의 상태가 변경되면 경보 임계값에 지정된 대로 배포를 중지할 수 있게 되었습니다. 또한 CodeDeploy는 이제 특정 조건(예: 배포 실패 또는 활성화된 경보 등)이 충족되면 배포의 자동 롤백을 지원합니다. 이러한 변경 사항을 반영하기 위해 새로운 주제인 CodeDeploy에서 CloudWatch 경보를 사용하여 배포 모니터링 단원과 함께 CodeDeploy를 사용하여 애플리케이션 생성, CodeDeploy에서 배포 그룹 만들기, CodeDeploy에서 배포 그룹 설정 변경, Deployments, CodeDeploy를 사용하여 재배포 및 배포 롤백 및 CodeDeploy와 제품 및 서비스 통합 단원을 비롯한 여러 주제가 업데이트되었습니다.	2016년 9월 15일
신규 및 업데이트된 주제	CodeDeploy, 이제 Amazon CloudWatch Events와의 통합 제공 이제 배포 상태 또는 CodeDeploy 배포 그룹에 속한 인스턴스의 상태 변경이 감지되면 CloudWatch Events를 사용하여 작업을 하나 이상 시작할 수 있습니다. AWS Lambda 함수를 호출하거나, Kinesis 스트림 또는 Amazon SNS 주제에 게시하거나, Amazon SQS 대기열에 메시지를 푸시하거나, CloudWatch 경보 작업을 트리거하는 작업을 통합할 수 있습니다. 자세한 내용은 Amazon CloudWatch Events를 사용하여 배포 모니터링 섹션을 참조하세요.	2016년 9월 9일

변경	설명	변경 날짜
주제 업데이트	추가 로드 밸런싱 옵션을 반영하기 위해 Integrating CodeDeploy with Elastic Load Balancing 및 다른 AWS 서비스와의 통합 주제가 업데이트되었습니다. 이제 CodeDeploy는 Elastic Load Balancing에서 사용할 수 있는 Classic Load Balancer 및 Application Load Balancer를 지원합니다.	2016년 8월 11일
주제 업데이트	이제 아시아 태평양(뭄바이) 리전(ap-south-1)에서 CodeDeploy를 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2016년 6월 27일
주제 업데이트	CodeDeploy는 이제 아시아 태평양(서울) 리전(ap-northeast-2)에서 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다. 인스턴스, 배포 구성, 애플리케이션, 배포 그룹, 개정 및 배포 단원을 포함하기 위해 목차가 다시 구성되었습니다. CodeDeploy 자습서에 대한 새 단원이 추가되었습니다. 더욱 쉽게 사용할 수 있도록 CodeDeploy AppSpec 파일 참조 및 CodeDeploy 문제 해결 단원을 비롯한 긴 주제 여러 개가 더 짧은 주제로 나뉘었습니다. CodeDeploy 에이전트에 대한 구성 정보가 새로운 주제인 CodeDeploy 에이전트 구성 참조 단원으로 이동되었습니다.	2016년 6월 15일
신규 및 업데이트된 주제	에 대한 오류 코드 AWS CodeDeploy 단원에서는 CodeDeploy 배포에 실패할 때 표시될 수 있는 몇 가지 오류 메시지에 대한 정보를 제공합니다. 배포 문제 해결을 더욱 잘 지원할 수 있도록 CodeDeploy 문제 해결의 다음 단원이 업데이트되었습니다. • Amazon EC2 Auto Scaling 그룹의 EC2 인스턴스를 시작하지 못하고 "하트비트 제한 시간" 오류 발생 • 단일 Amazon EC2 Auto Scaling 그룹으로 여러 배포 그룹 연결 피하기	2016년 20월 4일

변경	설명	변경 날짜
주제 업데이트	CodeDeploy는 이제 남아메리카(상파울루)(sa-east-1) 리전에서 사용 가능합니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다. 새 :max_revisions: 구성 옵션을 반영하도록 CodeDeploy 에이전트 작업이(가) 업데이트되었습니다. 이를 사용하여 CodeDeploy 에이전트에서 아카이브하도록 하려는 배포 그룹의 애플리케이션 개정 버전의 개수를 지정합니다.	2016년 3월 10일
신규 및 업데이트된 주제	이제 CodeDeploy에서는 배포 또는 배포 그룹의 인스턴스와 관련된 이벤트에 대한 알림을 수신하기 위해 배포 그룹에 트리거를 추가할 수 있도록 지원합니다. 이러한 알림은 트리거 작업의 일부로 만든 Amazon Simple Notification Service 주제를 구독하는 수신자에게 전송됩니다. 또한 고유한 사용자 지정 알림 워크플로에서 트리거가 실행되면 생성되는 JSON 데이터도 사용할 수 있습니다. 자세한 내용은 Monitoring Deployments with Amazon SNS Event Notifications 섹션을 참조하세요. 애플리케이션 세부 정보 페이지의 새로운 디자인을 반영하기 위해 절차가 업데이트되었습니다. CodeDeploy 문제 해결의 배포 중 인스턴스가 종료되면 최대 1시간 동안 배포에 실패하지 않음 단원이 업데이트되었습니다. 단일 애플리케이션과 연결할 수 있는 배포 그룹 개수에 대한 개정된 제한, 최소 정상 인스턴스 설정에 대해 허용되는 값 및 AWS SDK for Ruby의 필수 버전을 반영하기 위해 CodeDeploy 할당량 단원이 업데이트되었습니다.	2016년 2월 17일

변경	설명	변경 날짜
신규 및 업데이트된 주제	이제 미국 서부(캘리포니아 북부) 리전(us-west-1)에서 CodeDeploy를 사용할 수 있습니다. CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가이 새 리전의 추가를 반영하도록 업데이트되었습니다. CodeDeploy 리포지토리 유형 선택 단원에는 CodeDeploy에서 현재 지원되는 리포지토리 유형 목록과 설명이 나와 있습니다. 다른 리포지토리 유형에 대한 지원이 도입되어 이 새로운 주제가 업데이트되었습니다. CodeDeploy 에이전트의 현재 버전을 보고하기 위해 인스턴스에 추가된 새 `.version` 파일에 대한 정보와 지원되는 에이전트 버전에 대한 정보로 CodeDeploy 에이전트 작업 관리 단원이 업데이트되었습니다. 이 사용 설명서에는 JSON 및 YAML 예제를 비롯한 코드 샘플에 대한 구문 강조 표시가 추가되었습니다. CodeDeploy에 대한 개정에 애플리케이션 사양 파일 추가 단원이 단계별 지침으로 재구성되었습니다.	2016년 1월 20일
새 주제	다른 AWS 계정에 애플리케이션 배포 에서는 다른 계정에 대한 전체 자격 증명 세트 없이 조직의 다른 계정에 속하는 배포를 시작하기 위한 설정 요구 사항 및 프로세스를 설명합니다. 이는 개발 및 테스트 환경과 연결된 계정과 프로덕션 환경과 연결된 또 다른 계정 등 여러 계정을 다양한 용도로 사용하는 조직에 가장 유용합니다.	2015년 12월 30일
주제 업데이트	CodeDeploy와 제품 및 서비스 통합 주제가 새롭게 디자인되었습니다. 이제 이 주제에는 커뮤니티의 통합 예제, 블로그 게시글 목록 및 CodeDeploy 통합 관련 비디오 예제를 소개하는 단원이 포함되어 있습니다.	2015년 12월 16일

변경	설명	변경 날짜
주제 업데이트	이제 아시아 태평양(싱가포르) 리전(ap-southeast-1)에서 CodeDeploy를 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2015년 12월 9일
주제 업데이트	CodeDeploy 에이전트 구성 파일의 새로운 `:proxy_uri:` 옵션을 반영하기 위해 CodeDeploy 에이전트 작업 단원이 업데이트되었습니다. 배포 수명 주기 이벤트 중 후크 스크립트에서 액세스할 수 있는 새 환경 변수 `DEPLOYMENT_GROUP_ID` 사용에 대한 정보로 CodeDeploy AppSpec 파일 참조 단원이 업데이트되었습니다.	2015년 12월 1일
주제 업데이트	CodeDeploy를 위한 서비스 역할 만들기의 새로운 절차를 반영하고 다른 개선 사항을 포함하기 위해 2단계: CodeDeploy에 대한 서비스 역할 생성 단원이 업데이트되었습니다.	2015년 11월 13일
주제 업데이트	CodeDeploy는 이제 유럽(프랑크푸르트)(eu-central-1)에서 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다. 인스턴스에 대한 시간 설정이 정확하도록 보장하는 방법으로 CodeDeploy 문제 해결 주제가 업데이트되었습니다.	2015년 10월 19일
새로운 주제	CloudFormation CodeDeploy 참조용 템플릿는 CodeDeploy 작업에 대한 새로운 CloudFormation 지원을 반영하기 위해 게시되었습니다. Primary Components 주제를 작성해 대상 개정의 정의를 소개했습니다.	2015년 10월 1일

변경	설명	변경 날짜
주제 업데이트	와일드카드 검색을 사용하여 배포 그룹에 대한 인스턴스를 찾는 기능을 소개하기 위해 CodeDeploy에서 배포 그룹 만들기 단원이 업데이트되었습니다. 최소 정상 인스턴스의 개념을 명확히하기 위해 Instance Health 단원이 업데이트되었습니다	2015년 8월 31일
주제 업데이트	CodeDeploy는 이제 아시아 태평양(도쿄) 리전(ap-northeast-1)에서 사용할 수 있습니다. 새 리전에서 사용 가능성을 반영하기 위해 CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 업데이트되었습니다.	2015년 8월 19일
주제 업데이트	이제 CodeDeploy는 Red Hat Enterprise Linux (RHEL) 온프레미스 인스턴스 및 Amazon EC2 인스턴스에 대한 배포를 지원합니다. 자세한 정보는 다음 주제를 참조하세요. • CodeDeploy 에이전트에서 지원하는 운영 체제 • CodeDeploy용 인스턴스 작업 • 튜토리얼: Amazon EC2 인스턴스(Amazon Linux 또는 Red Hat Enterprise Linux 및 Linux, macOS 또는 Unix)에 WordPress 배포 • 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux)	2015년 6월 23일
주제 업데이트	이제 CodeDeploy에서는 배포 스크립트에서 배포 중 사용할 수 있는 환경 변수 세트를 제공합니다. 이러한 환경 변수에는 현재 CodeDeploy 애플리케이션의 이름, 배포 그룹, 배포 수명 주기 이벤트, 현재 CodeDeploy 배포 식별자 등의 정보가 포함됩니다. 자세한 내용은 AppSpec 'hooks' 섹션의 CodeDeploy AppSpec 파일 참조 단원 끝 부분을 참조하세요.	2015년 5월 29일

변경	설명	변경 날짜
주제 업데이트	CodeDeploy는 이제 IAM에서 직접 동등한 정책을 수동으로 생성하는 대신 사용할 수 있는 AWS 관리형 정책 세트를 제공합니다. 다음이 포함됩니다. • 사용자가 CodeDeploy를 사용해서만 개정 버전을 등록한 다음 CodeDeploy를 통해 배포하도록 하는 정책 • 사용자에게 CodeDeploy 리소스에 대한 전체 액세스 권한을 제공하는 정책 • 사용자에게 CodeDeploy 리소스에 대한 읽기 전용 액세스 권한을 제공하는 정책 • 서비스 역할에 연결하는 정책. 따라서 CodeDeploy가 Amazon EC2 태그, 온프레미스 인스턴스 태그 또는 Amazon EC2 Auto Scaling 그룹 이름으로 Amazon EC2 인스턴스를 식별하고 이에 따라 식별한 인스턴스에 애플리케이션 개정을 배포할 수 있습니다. 자세한 내용은 인증 및 액세스 제어의 고객 관리형 정책 예제 단원을 참조하세요.	2015년 5월 29일
주제 업데이트	이제 유럽(아일랜드) 리전(eu-west-1) 및 아시아 태평양(시드니) 리전(ap-southeast-2)에서 CodeDeploy를 사용할 수 있습니다. CodeDeploy 에이전트 설정 지침이 포함된 주제를 비롯한 여러 주제가 이러한 새 리전의 가용성을 반영하도록 업데이트되었습니다.	2015년 5월 7일
새로운 주제	이제 CodeDeploy는 온프레미스 인스턴스 및 Amazon EC2 인스턴스에 대한 배포를 지원합니다. 이 새로운 지원을 설명하기 위해 다음 주제를 추가했습니다. • Working with On-Premises Instances • 튜토리얼: CodeDeploy를 사용하여 온프레미스 인스턴스에 애플리케이션 배포(Windows Server, Ubuntu Server 또는 Red Hat Enterprise Linux) • Working with On-Premises Instances	2015년 4월 2일

변경	설명	변경 날짜
새 주제	CodeDeploy 리소스 단원이 추가되었습니다.	2015년 4월 2일
주제 업데이트	CodeDeploy 문제 해결 단원이 업데이트되었습니다. • 새로운 오래 실행되는 프로세스로 인해 배포에 실패할 수 있음 단원에서는 오래 실행되는 프로세스로 인한 배포 실패를 식별하여 해결하기 위해 수행할 수 있는 단계를 설명합니다. • CodeDeploy가 CodeDeploy 에이전트에 대한 Amazon EC2 Auto Scaling 시간 제한 로직을 5분에서 1시간으로 늘렸음을 설명하기 위해 일반적인 Amazon EC2 Auto Scaling 문제 해결 단원이 업데이트되었습니다. • 새로운 일치하지 않는 Amazon EC2 Auto Scaling 수명 주기 후크로 인해 Amazon EC2 Auto Scaling 그룹에 대한 자동 배포가 중지되거나 실패할 수 있습니다. 단원에서는 Amazon EC2 Auto Scaling 그룹으로의 자동 배포 실패를 식별하고 해결하기 위해 수행할 수 있는 단계를 설명합니다.	2015년 4월 2일

변경	설명	변경 날짜
주제 업데이트	고유한 사용자 지정 정책을 만든 다음 IAM에서 사용자 및 역할에 연결하기 위한 새로운 권장 사항을 반영하기 위해 다음 주제가 업데이트되었습니다. • CodeDeploy 작업을 위한 Amazon EC2 인스턴스 구성 • 4단계: Amazon EC2 인스턴스에 대한 IAM 인스턴스 프로파일 만들기 • 2단계: CodeDeploy에 대한 서비스 역할 생성 CodeDeploy 문제 해결 단원에 다음 두 가지 단원이 추가되었습니다. • 일반적인 문제 해결 체크리스트 • Windows PowerShell 스크립트에서 기본적으로 Windows PowerShell 64비트 버전을 사용하지 못함 사용 가능한 배포 수명 주기 이벤트에 대해 보다 정확하게 설명하기 위해 CodeDeploy AppSpec 파일 참조의 AppSpec 'hooks' 섹션 단원이 업데이트되었습니다.	2015년 2월 12일
주제 업데이트	새 단원이 CodeDeploy 문제 해결: Amazon EC2 Auto Scaling 그룹의 EC2 인스턴스를 시작하지 못하고 "하트비트 제한 시간” 오류 발생에 추가되었습니다. CloudBees 단원이 CodeDeploy와 제품 및 서비스 통합에 추가되었습니다.	2015년 1월 28일

변경	설명	변경 날짜
주제 업데이트	다음 단원이 CodeDeploy 문제 해결에 추가되었습니다. • 일부 텍스트 편집기를 사용하여 AppSpec 파일 및 쉘 스크립트를 작성하면 배포에 실패할 수 있음 • macOS에서 Finder를 사용하여 애플리케이션 수정을 번들링하면 배포에 실패할 수 있음 • 실패한 ApplicationStop, BeforeBlockTraffic 또는 AfterBlockTraffic 배포 수명 주기 이벤트 문제 해결 • 실패하고 UnknownError: not opened for reading이 표시되는 DownloadBundle 배포 수명 주기 이벤트 문제 해결 • 일반적인 Amazon EC2 Auto Scaling 문제 해결	2015년 1월 20일
새로운 주제	다음 주제를 포함하기 위해 CodeDeploy와 제품 및 서비스 통합 단원이 업데이트되었습니다. • Amazon EC2 Auto Scaling과 CodeDeploy 통합 • 튜토리얼: CodeDeploy를 사용하여 Auto Scaling 그룹에 애플리케이션 배포 • Monitoring Deployments • Integrating CodeDeploy with Elastic Load Balancing • GitHub와 CodeDeploy 통합 • 튜토리얼: CodeDeploy를 사용하여 GitHub에서 애플리케이션 배포	2015년 1월 9일

변경	설명	변경 날짜
주제 업데이트	• CodeDeploy를 사용하여 CodePipeline에서 자동으로 배포 단원이 GitHub와 CodeDeploy 통합에 추가되었습니다. GitHub 리포지토리의 소스 코드가 변경될 때마다 해당 리포지토리에서 배포를 자동으로 트리거할 수 있습니다. • Amazon EC2 Auto Scaling 문제 해결 단원이 CodeDeploy 문제 해결에 추가되었습니다. 이 새로운 섹션에서는 Amazon EC2 Auto Scaling 그룹에 배포와 관련된 일반적인 문제를 해결하는 방법을 설명합니다. • 새로운 하위 단원인 "파일의 예"가 CodeDeploy AppSpec 파일 참조의 AppSpec 'files' 섹션(EC2/온프레미스 배포만 해당) 단원에 추가되었습니다. 이 새로운 하위 단원에는 AppSpec 파일의 `files` 단원을 사용하여 배포 중 특정 파일 또는 폴더를 Amazon EC2 인스턴스의 특정 위치로 복사하도록 CodeDeploy에 지시하는 방법을 보여주는 여러 가지 예가 포함되어 있습니다.	2015년 1월 8일
새 주제	Monitoring Deployments 단원이 추가되었습니다. CodeDeploy는 AWS 계정에서 CodeDeploy에 의해 또는 이를 대신하여 수행된 API 호출을 캡처하고 사용자가 지정한 Amazon S3 버킷에 로그 파일을 전송하는 AWS CloudTrail 서비스와 통합됩니다.	2014년 12월 17일
최초 공개 릴리스	이는 CodeDeploy 사용 설명서의 최초 공개 릴리스입니다.	2014년 11월 12일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조의 AWS 용어집을 참조하세요.

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