사용자 가이드

Amazon EC2 Auto Scaling

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon EC2 Auto Scaling: 사용자 가이드

Copyright © 2024 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

Amazon EC2 Auto Scaling이란 무엇입니까? 1
아마존 EC2 오토 스케일링의 특징 1
아마존 EC2 오토 스케일링 요금 3
시작 3
Auto Scaling 그룹 작업 3
Auto Scaling의 이점 4
예: 가변적인 수요에 대응 5
예: 웹 앱 아키텍처 7
예: 가용 영역 전반에 인스턴스 분산 8
인스턴스 수명 주기 11
스케일 아웃 12
서비스 상태의 인스턴스 13
축소 13
인스턴스 분리 14
인스턴스 연결 14
수명 주기 후크 14
대기 모드 시작 및 해지 15
아마존 EC2 오토 스케일링 할당량 15
Amazon EC2 Auto Scaling에 대한 요청 제한 API 17
EC2해지 요금 17
기타 서비스 17
설정 19
Amazon EC2 사용 준비 19
사용을 준비하세요. AWS CLI 19
시작 20
튜토리얼: 첫 번째 Auto Scaling 그룹 생성 20
연습 준비 21
1단계: 출범 템플릿 생성 21
2단계: 단일 인스턴스 Auto Scaling 그룹 생성 22
3단계: Auto Scaling 그룹 확인 23
4단계: Auto Scaling 그룹에서 인스턴스 해지 24
5단계: 다음 절차 25
6단계: 정리 25
자습서: 조정 및 로드 밸런싱된 애플리케이션 설정 26

사전 조건 ... 28
1단계: 시작 템플릿 또는 시작 구성 설정 ... 28
2단계: Auto Scaling 그룹 생성 ... 32
3단계: 로드 밸런서가 연결됐는지 확인 ... 33
4단계: 다음 단계 ... 34
5단계: 정리 ... 34
관련 리소스 ... 36
시작 템플릿 ... 37
시작 템플릿으로 작업할 수 있는 권한 ... 38
API 시작 템플릿에서 지원하는 작업 ... 38
Auto Scaling 그룹에 대한 시작 템플릿 생성 ... 38
시작 템플릿 생성(콘솔) ... 39
기본 네트워크 인터페이스 설정 변경(콘솔) ... 41
스토리지 구성 수정(콘솔) ... 43
기존 인스턴스에서 시작 템플릿 생성(콘솔) ... 46
관련 리소스 ... 46
제한 사항 ... 46
고급 설정을 사용하여 시작 템플릿 생성 ... 47
필수 설정 ... 47
고급 설정 ... 47
스팟 인스턴스 요청 ... 52
Capacity Blocks ML용 ... 53
Auto Scaling 그룹을 마이그레이션하여 템플릿 시작 ... 58
1단계: 시작 구성을 사용하는 Auto Scaling 그룹 찾기 ... 58
2단계: 시작 구성을 시작 템플릿에 복사 ... 61
3단계: 시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트 ... 62
4단계: 인스턴스 교체 ... 63
추가 정보 ... 63
CloudFormation 스택을 시작 템플릿으로 마이그레이션 ... 63
시작 구성을 사용하는 Auto Scaling 그룹 찾기 ... 64
시작 템플릿을 사용하도록 스택 업데이트 ... 65
스택 리소스의 업데이트 동작 이해 ... 68
마이그레이션 추적 ... 69
시작 구성 매핑 참조 ... 69
AWS CLI 시작 템플릿 작업 예제 ... 71
사용 예 ... 71

기본 시작 템플릿 생성 ... 72
시작 시 인스턴스에 태그를 지정하는 태그 지정 ... 73
인스턴스에 전달할 IAM 역할 지정 ... 73
퍼블릭 IP 주소 배정 ... 74
시작 시 인스턴스를 구성하는 사용자 데이터 스크립트 지정 ... 74
블록 디바이스 매핑 지정 ... 74
외부 공급 업체의 소프트웨어 라이선스를 가져오기 위한 전용 호스트 지정 ... 75
기존 네트워크 인터페이스 지정 ... 75
여러 네트워크 인터페이스 생성 ... 75
시작 템플릿 관리 ... 76
시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트 ... 79
대신 Systems Manager 파라미터 사용 AMI IDs ... 80
에 대한 파라미터를 지정하는 시작 템플릿 생성 AMI ... 80
시작 템플릿이 올바른 AMI ID를 가져오는지 확인 ... 85
관련 리소스 ... 86
제한 사항 ... 86
시작 구성 ... 87
시작 구성 생성 ... 87
시작 구성 생성 ... 88
IMDS 구성 ... 91
EC2 인스턴스를 사용하여 시작 구성 생성 ... 93
시작 구성 변경 ... 97
Auto Scaling 그룹 ... 99
시작 템플릿을 사용하여 Auto Scaling 그룹 생성 ... 100
시작 템플릿을 사용하여 그룹 생성 ... 101
EC2 시작 마법사를 사용하여 그룹 생성 ... 103
여러 인스턴스 유형 및 구매 옵션 사용 ... 107
시작 구성을 사용하여 Auto Scaling 그룹 생성 ... 149
시작 구성을 사용하여 그룹 생성 ... 150
EC2 인스턴스를 사용하여 그룹 생성 ... 153
Auto Scaling 그룹 업데이트 ... 158
Auto Scaling 인스턴스 업데이트 ... 159
그룹 및 인스턴스 태그 지정 ... 160
태그 이름 지정 및 사용 제한 ... 161
EC2 인스턴스 태그 지정 수명 주기 ... 162
Auto Scaling 그룹에 태그 지정 ... 162

태그 삭제 ... 165
보안을 위한 태그 ... 166
태그에 대한 액세스 통제 ... 167
태그를 사용하여 Auto Scaling 그룹 필터링 ... 168
인스턴스 유지 관리 정책 ... 172
개요 ... 172
그룹의 인스턴스 유지 관리 정책 설정 ... 179
수명 주기 후크 ... 183
수명 주기 후크 가용성 ... 184
고려 사항 및 제한 ... 184
관련 리소스 ... 186
Auto Scaling 그룹에서 라이프사이클 후크가 작동하는 방식 ... 186
수명 주기 후크 추가 준비 ... 188
대상 수명 주기 상태 검색 ... 196
Auto Scaling 그룹에 라이프사이클 후크 추가 ... 198
Auto Scaling 그룹에서 라이프사이클 작업 완료 ... 201
자습서: 인스턴스 메타데이터를 사용하여 수명 주기 상태 검색 ... 203
자습서: Lambda 함수를 호출하는 수명 주기 후크 구성 ... 211
웜 풀 ... 220
핵심 개념 ... 221
사전 조건 ... 223
웜 풀에서 인스턴스 업데이트 ... 225
관련 리소스 ... 225
제한 사항 ... 225
수명 주기 후크 사용 ... 226
Auto Scaling 그룹을 위한 웜 풀 생성 ... 229
건전성 체크의 상태 보기 ... 231
AWS CLI 웜 풀 사용 예시 ... 234
인스턴스 분리-연결 ... 237
인스턴스 분리 시 고려 사항 ... 237
인스턴스 연결 시 고려 사항 ... 238
분리 및 연결을 사용하여 인스턴스를 다른 그룹으로 이동 ... 239
일시적으로 인스턴스 제거 ... 244
대기 상태를 작동하는 방법 ... 244
고려 사항 ... 245
대기 상태의 인스턴스 상태 ... 245

인스턴스를 대기 상태로 설정하여 임시로 제거 244
Auto Scaling 인프라 삭제 250
Auto Scaling 그룹 삭제 250
(옵션) 시작 구성 삭제 251
(옵션) 시작 템플릿 삭제 252
(옵션) 로드 밸런서 및 대상 그룹 삭제 253
(선택 사항) 경보 삭제 CloudWatch 253
인스턴스 교체 255
인스턴스 새로 고침 255
인스턴스 새로 고침 작동 방식 256
기본값 이해 261
인스턴스 새로 고침 시작 264
인스턴스 새로 고침 모니터링 275
인스턴스 새로 고침 취소 278
롤백으로 변경 취소 279
매칭 건너뛰기 사용 284
체크포인트 추가 292
최대 인스턴스 수명 298
고려 사항 298
최대 인스턴스 수명 설정 299
제한 사항 300
그룹 조정 301
스케일링 방법 선택 301
스케일링 한도 설정 303
기본 인스턴스 워밍업 설정 304
스케일링 수행 고려 사항 305
기본 인스턴스 워밍업 시간을 선택합니다. 306
그룹에 대한 기본 인스턴스 워밍업 활성화 306
그룹의 기본 인스턴스 워밍업 시간을 확인하십시오. 308
이전에 설정한 인스턴스 워밍업 시간의 조정 정책을 찾아보세요. 309
스케일링 정책에 대해 이전에 설정한 인스턴스 워밍업을 지웁니다. 310
수동 조정 311
귀하의 Auto Scaling 그룹의 원하는 용량 변경 311
Auto Scaling 그룹에서 인스턴스 해지 (AWS CLI) 315
예약된 조정 316
예약된 조정 작동 방식 317

반복되는 일정 ... 317
시간대 ... 318
고려 사항 ... 318
제한 사항 ... 319
예약된 작업 생성 ... 319
예약된 작업 세부 정보 보기 ... 321
예약된 작업 삭제 ... 322
동적 조정 ... 323
동적 조정 정책 작동 방식 ... 324
여러 동적 조정 정책 ... 325
대상 추적 조정 정책 ... 326
단계별 조정 및 단순 조정 정책 ... 339
조정 휴지 ... 354
아마존 기반 규모 조정 정책 SQS ... 357
크기 조정 활동 확인 ... 364
조정 정책 비활성화 ... 366
Auto Scaling 그룹의 조정 정책 삭제 ... 368
AWS CLI 규모 조정 정책의 예 ... 371
예측 조정 ... 373
예측 조정의 작동 방식 ... 374
예측 규모 조정 정책 생성 ... 378
예측적 조정 정책 평가 ... 386
예측 재정의 ... 394
사용자 정의 지표 사용 ... 399
인스턴스 해지 제어 ... 409
해지 정책 시나리오 ... 409
종료 정책을 구성합니다. ... 413
Lambda를 사용하여 사용자 지정 해지 정책 생성 ... 419
인스턴스 스케일 인 방비 사용 ... 424
정상적인 인스턴스 해지를 위한 설계 ... 428
프로세스 일시 중지 후 재개 ... 431
프로세스 유형 ... 432
고려 사항 ... 433
프로세스 일시 중지 ... 434
프로세스 재개 ... 435
일시 중단된 프로세스가 다른 프로세스에 미치는 영향 ... 435

모니터링 440
상태 확인 442
상태 확인 설명 443
상태 확인 유예 기간 설정 449
손상된 Amazon EBS 볼륨 모니터링 451
사용자 지정 상태 점검 설정 454
상태 확인 불합격 이유 확인 456
비정상 인스턴스 문제 해결 457
를 사용하여 모니터링하십시오. AWS Health Dashboard 460
CloudWatch 지표 모니터링 462
Amazon EC2 Auto Scaling 콘솔에서 모니터링 그래프 보기 462
CloudWatch Amazon EC2 Auto Scaling에 대한 지표 466
Auto Scaling 인스턴스에 대한 모니터링 구성 473
를 사용하여 API 통화 로그 CloudTrail 475
의 Auto Scaling 관리 이벤트 CloudTrail 476
Auto Scaling 이벤트 예제 476
아마존 SNS 알림 옵션 478
SNS아마존과 아마존 EC2 오토 스케일링 478
다른 서비스와 함께 작동 484
용량 재조정 484
개요 485
용량 재조정 동작 486
고려 사항 486
용량 재분배 활성화 488
용량 예약 494
1단계: 용량 예약 생성 495
2단계: 용량 예약 그룹 생성 497
3단계: 시작 템플릿 생성 499
4단계: Auto Scaling 그룹 생성 500
관련 리소스 502
AWS CloudShell 503
AWS CloudFormation 503
Amazon EC2 Auto Scaling 및 템플릿 AWS CloudFormation 503
에 대해 자세히 알아보십시오. AWS CloudFormation 504
Compute Optimizer 504
제한 사항 505

조사 결과 ... 505
권장 사항 보기 ... 506
권장 사항 평가를 위한 고려 사항 ... 507
Elastic Load Balancing ... 508
Elastic Load Balancing 유형 ... 509
로드 밸런서 연결 준비 ... 510
로드 밸런서 연결 ... 512
로드 밸런서 구성 ... 515
연결 상태 확인 ... 517
가용 영역 추가 ... 518
가용 영역 제거 ... 520
로드 밸런서 분리 ... 515
AWS CLI Elastic Load Balancing 사용 예제 ... 522
VPC래티스 ... 529
대상 그룹을 연결할 준비 ... 531
VPC래티스 대상 그룹 연결 ... 534
연결 상태 확인 ... 538
EventBridge ... 539
Amazon EC2 Auto Scaling 이벤트 참조 ... 540
웜 풀 예 이벤트 및 패턴 ... 550
EventBridge 규칙 ... 556
Amazon VPC ... 560
기본 VPC ... 561
기본이 아닌 VPC ... 561
VPC 서브넷 선택 시 고려 사항 ... 562
VPC에서 IP 주소 지정 ... 562
VPC의 네트워크 인터페이스 ... 563
인스턴스 배치 테넌시 ... 563
AWS Outposts ... 563
VPC에 대해 자세히 알 수 있는 추가 리소스 ... 564
보안 ... 565
인프라 보안 ... 565
관련 리소스 ... 566
복원력 ... 566
관련 리소스 ... 567
데이터 보호 ... 568

Amazon EBS 볼륨을 암호화하는 AWS KMS keys 데 사용 569
관련 리소스 569
AWS KMS 암호화된 볼륨에 사용하기 위한 키 정책 569
ID 및 액세스 관리 575
액세스 제어 575
Amazon EC2 Auto Scaling과 함께 작동하는 방식 IAM 576
API 권한 585
관리형 정책 586
서비스 연결 역할 590
자격 증명 기반 정책 예시 595
교차 서비스 혼동된 대리인 방지 604
Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다. 606
Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 614
규정 준수 확인 617
PCIDSS규정 준수 618
VPC 엔드포인트를 사용한 프라이빗 연결 618
인터페이스 VPC 엔드포인트 생성 619
VPC 엔드포인트 정책 생성 619
작업 AWS SDKs 621
코드 예시 623
기본 사항 635
Auto Scaling 시작 637
기본 사항 알아보기 647
작업 746
시나리오 933
복원력이 뛰어난 서비스 구축 및 관리 933
문제 해결 1103
오류 메시지 검색 1103
스케일링 활동을 끕니다. 1105
추가 문제 해결 리소스 1106
인스턴스 시작 실패 1107
요청된 구성이 현재 지원되지 않습니다. 1108
보안 그룹 <보안 그룹 명칭>이(가) 존재하지 않습니다. EC2인스턴스 시작에 실패했습니다. 1108
키 페어 EC2 <인스턴스에 연결된 키 페어>가 존재하지 않습니다. 인스턴스 EC2 시작에 실패했습니다. 1109

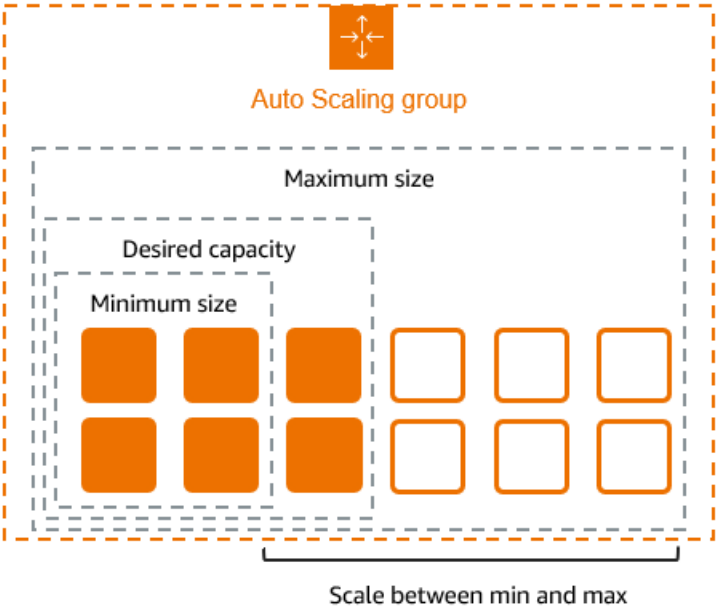
요청된 인스턴스 유형(<인스턴스 유형>)이 요청된 가용 영역(<인스턴스 가용 영역>)에서 지원되지 않습니다... 1109
스팟 요청 가격인 0.015가 필요한 최소 스팟 요청 이행 가격인 0.0735보다 낮습니다. 1109
잘못된 디바이스 명칭 <device name>/잘못된 디바이스 명칭 업로드. EC2인스턴스 시작에 실패했습니다. 1110
<name associated with the instance storage device>파라미터 값 () virtualName 이 유효하지 않습니다... EC2인스턴스 시작에 실패했습니다. 1110
EBS블록 디바이스 매핑은 인스턴스 스토어에 지원되지 않습니다. AMIs 1111
배치 그룹은 유형 '<인스턴스 유형>'의 인스턴스와 함께 사용할 수 없습니다. EC2인스턴스 시작에 실패했습니다. 1111
클라이언트. InternalError: 시작 시 클라이언트 오류가 발생했습니다. 1111
요청한 가용 영역에 현재 <인스턴스 유형> 용량이 부족합니다... EC2인스턴스 시작에 실패했습니다. 1112
요청된 예약에는 이 요청에 사용할 수 있는 호환 및 사용 가능한 용량이 충분하지 않습니다. EC2인스턴스 시작에 실패했습니다. 1113
용량 블록 <reservation id> 예약이 아직 활성화되지 않았습니다. EC2인스턴스 시작에 실패했습니다. 1114
요청과 일치하는 스팟 용량이 없습니다. EC2인스턴스 시작에 실패했습니다. 1114
<인스턴스 수>개의 인스턴스가 이미 실행 중입니다. EC2인스턴스 시작에 실패했습니다. 1114
AMI문제 1115
AMIID <귀하의 ID AMI > 이 (가) 존재하지 않습니다. EC2인스턴스 시작에 실패했습니다. 1115
AMI< AMI ID>는 보류 중이므로 실행할 수 없습니다. EC2인스턴스 시작에 실패했습니다. 1116
디바이스 명칭 <device name>이(가) 잘못되었습니다. EC2인스턴스 시작에 실패했습니다. 1116
지정된 인스턴스 유형의 'arm64' 아키텍처가 지정된 아키텍처 'x86_64'와 일치하지 않습니다... 인스턴스 시작에 실패했습니다. AMI EC2 1116
AMI'< AMI ID>'는 비활성화되어 실행할 수 없습니다. 인스턴스 EC2 시작에 실패했습니다. 1118
로드 밸런서 문제 1118
One or more target groups not found.(하나 이상의 대상 그룹을 찾을 수 없습니다.) 로드 밸런서 구성의 유효성 검사에 실패했습니다. 1119
Cannot find Load Balancer <your load balancer>.(로드 밸런서 <사용자의 로드 밸런서>를 찾을 수 없습니다.) 로드 밸런서 구성의 유효성 검사에 실패했습니다. 1119
<load balancer name>이름이 지정된 ACTIVE Load Balancer가 없습니다. 로드 밸런서 구성을 업데이트하는 데 실패했습니다. 1120
EC2<instance ID>인스턴스가 들어있지 않습니다. VPC 로드 밸런서 구성을 업데이트하는 데 실패했습니다. 1120
시작 템플릿 문제 1120

완전한 형태의 유효한 시작 템플릿을 사용해야 합니다(잘못된 값). 1120
시작 템플릿을 사용할 권한이 없음(권한 부족) 1121
관련 정보 1123
문서 이력 1125
..... mclviii

Amazon EC2 Auto Scaling이란 무엇입니까?

Amazon EC2 Auto Scaling을 사용하면 애플리케이션의 부하를 처리하는 데 사용할 수 있는 Amazon EC2 인스턴스 수가 정확한지 확인할 수 있습니다. Auto Scaling 그룹이라는 EC2 인스턴스 컬렉션을 생성합니다. 각 Auto Scaling 그룹의 최소 인스턴스 수를 지정할 수 있으며, Amazon EC2 Auto Scaling은 그룹이 이 크기 이하로 떨어지지 않도록 보장합니다. 각 Auto Scaling 그룹의 최대 인스턴스 수를 지정할 수 있으며, Amazon EC2 Auto Scaling은 그룹이 이 크기를 초과하지 않도록 보장합니다. 그룹을 생성할 때 또는 그룹을 생성한 후 언제든지 원하는 용량을 지정하면 Amazon EC2 Auto Scaling은 그룹에 이 만큼의 인스턴스가 있는지 확인합니다. 조정 정책을 지정하는 경우 Amazon EC2 Auto Scaling은 애플리케이션에 대한 수요가 증가하거나 감소함에 따라 인스턴스를 시작하거나 종료할 수 있습니다.

예를 들어, 다음 Auto Scaling 그룹의 최소 인스턴스 크기는 4개, 원하는 인스턴스 용량은 6개, 최대 인스턴스 크기는 12개입니다. 사용자가 정의한 조정 정책에 따라 인스턴스 수가 최소 및 최대 인스턴스 수 내에서 지정하는 조건에 따라 조절됩니다.



아마존 EC2 오토 스케일링의 특징

Amazon EC2 Auto Scaling을 사용하면 EC2 인스턴스가 Auto Scaling 그룹으로 구성되어 조정 및 관리를 위한 논리적 단위로 취급될 수 있습니다. Auto Scaling 그룹은 시작 템플릿 (또는 시작 구성) 을 EC2 인스턴스의 구성 템플릿으로 사용합니다.

Amazon EC2 Auto Scaling의 주요 기능은 다음과 같습니다.

실행 중인 인스턴스의 상태 모니터링

Amazon EC2 Auto Scaling은 EC2 상태 확인을 사용하여 인스턴스의 상태와 가용성을 자동으로 모니터링하고, 종료되거나 손상된 인스턴스를 교체하여 원하는 용량을 유지합니다.

사용자 지정 상태 확인

내장된 상태 확인 외에도 애플리케이션별 사용자 지정 상태 확인을 정의하여 예상대로 응답하는지 확인할 수 있습니다. 인스턴스가 사용자 지정 상태 확인에 실패하면 원하는 용량을 유지할 수 있도록 자동으로 교체됩니다.

가용 영역 간 용량 균형 조정

Auto Scaling 그룹에 여러 가용 영역을 지정할 수 있으며, Amazon EC2 Auto Scaling은 그룹이 확장됨에 따라 가용 영역 전체에 걸쳐 인스턴스의 균형을 균등하게 조정합니다. 이는 단일 위치에서 애플리케이션에 장애가 발생하지 않도록 보호함으로써 고가용성과 탄력성을 제공합니다.

여러 인스턴스 유형 및 구매 옵션

단일 Auto Scaling 그룹 내에서 여러 인스턴스 유형과 구매 옵션 (스팟 및 온디맨드 인스턴스) 을 시작하여 스팟 인스턴스 사용을 통해 비용을 최적화할 수 있습니다. 예약 인스턴스 및 Savings Plan 할인을 그룹의 온디맨드 인스턴스와 함께 사용하여 할인 혜택을 받을 수도 있습니다.

스팟 인스턴스의 자동 교체

그룹에 스팟 인스턴스가 포함된 경우, 스팟 인스턴스가 중단되면 Amazon EC2 Auto Scaling에서 자동으로 대체 스팟 용량을 요청할 수 있습니다. Amazon EC2 Auto Scaling은 용량 재조정을 통해 중단 위험이 높은 스팟 인스턴스를 모니터링하고 사전에 교체할 수도 있습니다.

로드 밸런싱

Elastic Load Balancing 부하 분산 및 상태 확인을 사용하여 애플리케이션 트래픽이 정상 인스턴스에 고르게 분배되도록 할 수 있습니다. 인스턴스가 시작되거나 종료될 때마다 Amazon EC2 Auto Scaling은 로드 밸런서에서 인스턴스를 자동으로 등록 및 등록 취소합니다.

확장성

또한 Amazon EC2 Auto Scaling은 Auto Scaling 그룹을 확장할 수 있는 여러 가지 방법을 제공합니다. Auto Scaling을 사용하면 최대 부하를 처리할 수 있는 용량을 추가하고 수요가 낮을 때는 용량을 제거하여 애플리케이션 가용성을 유지하고 비용을 절감할 수 있습니다. 필요에 따라 Auto Scaling 그룹의 크기를 수동으로 조정할 수도 있습니다.

인스턴스 새로 고침

인스턴스 새로 고침 기능은 템플릿을 AMI 업데이트하거나 시작할 때 순차적으로 인스턴스를 업데이트하는 메커니즘을 제공합니다. 카나리아 배포라고 하는 단계별 접근 방식을 사용하여 전체 그룹

에 배포하기 전에 작은 인스턴스 집합에서 새 템플릿 AMI 또는 시작 템플릿을 테스트할 수도 있습니다.

수명 주기 후크

라이프사이클 후크는 새 인스턴스 시작 시 또는 인스턴스 종료 전에 호출되는 사용자 지정 작업을 정의하는 데 유용합니다. 이 기능은 이벤트 기반 아키텍처를 구축하는 데 특히 유용하지만 수명 주기 전반에 걸쳐 인스턴스를 관리하는 데도 도움이 됩니다.

스테이트풀 워크로드 지원

라이프사이클 후크는 종료 시 상태를 유지하기 위한 메커니즘도 제공합니다. 상태 저장 애플리케이션의 연속성을 보장하기 위해 확장 보호 또는 사용자 지정 종료 정책을 사용하여 장기 실행 프로세스가 있는 인스턴스가 조기에 종료되지 않도록 할 수도 있습니다.

Amazon EC2 Auto Scaling의 이점에 대한 자세한 내용은 을 참조하십시오애플리케이션 아키텍처에 대한 Auto Scaling의 이점.

아마존 EC2 오토 스케일링 요금

Amazon EC2 Auto Scaling에는 추가 비용이 없으므로 쉽게 사용해 보고 AWS 아키텍처에 어떤 이점이 있는지 확인할 수 있습니다. 사용한 AWS 리소스 (예: EC2 인스턴스, EBS 볼륨, CloudWatch 경보) 에 대한 비용만 지불하면 됩니다.

시작

시작하려면 첫 번째 Auto Scaling 그룹 생성 튜토리얼을 완료하여 Auto Scaling 그룹을 생성하고 해당 그룹의 인스턴스가 종료될 때 그룹이 어떻게 반응하는지 확인하세요.

Auto Scaling 그룹 작업

다음 인터페이스 중 하나를 사용하여 Auto Scaling 그룹을 생성, 액세스 및 관리할 수 있습니다:

- AWS Management Console - Auto Scaling 그룹에 액세스할 때 사용할 수 있는 웹 인터페이스를 제공합니다. 에 가입한 경우 에 로그인하고 탐색 표시줄의 검색 상자를 사용하여 Auto Scaling 그룹을 검색한 다음 Auto Scaling 그룹을 선택하여 Auto Scaling 그룹에 액세스할 수 있습니다. AWS 계정 AWS Management Console

- AWS Command Line Interface (AWS CLI) — 다양한 명령을 제공하며 Windows AWS 서비스, macOS 및 Linux에서 지원됩니다. 시작하려면 사용을 준비하세요. AWS CLI 섹션을 참조하십시오. 자세한 설명은 AWS CLI 명령 참조의 autoscaling을 참조하세요.
- AWS Tools for Windows PowerShell— PowerShell 환경에서 스크립트를 작성하는 사용자를 위해 다양한 AWS 제품에 대한 명령을 제공합니다. 시작하려면 AWS Tools for Windows PowerShell 사용자 가이드를 참조하세요. 자세한 설명은 AWS Tools for PowerShell Cmdlet 참조를 참조하세요.
- AWS SDKs— 언어별 API 작업을 제공하고 서명 계산, 요청 재시도 처리, 오류 처리와 같은 다양한 연결 세부 정보를 처리합니다. 자세한 내용은 AWS SDKs를 참조하세요.
- 쿼리 API — 요청을 사용하여 호출하는 저수준 API 작업을 제공합니다. HTTPS 쿼리를 사용하는 API 것이 가장 직접적인 액세스 AWS 서비스방법입니다. 하지만 이를 사용하려면 애플리케이션에서 요청에 서명할 해시 생성 및 오류 처리와 같은 하위 수준의 세부 정보를 처리해야 합니다. 자세한 내용은 Amazon EC2 Auto Scaling API 레퍼런스를 참조하십시오.
- AWS CloudFormation— CloudFormation 템플릿을 사용하여 Auto Scaling 그룹을 생성할 수 있습니다. 자세한 내용은 AWS CloudFormation을 이용한 Auto Scaling 그룹 생성 단원을 참조하십시오.

프로그래밍 방식으로 AWS 서비스연결하려면 엔드포인트를 사용합니다.

애플리케이션 아키텍처에 대한 Auto Scaling의 이점

Amazon EC2 Auto Scaling을 애플리케이션 아키텍처에 추가하는 것은 AWS 클라우드의 이점을 극대화하는 한 가지 방법입니다. Amazon EC2 Auto Scaling을 사용하면 애플리케이션에서 다음과 같은 이점을 얻을 수 있습니다.

- 향상된 내결함성. Amazon EC2 Auto Scaling은 인스턴스가 비정상일 때를 감지하여 종료하고 인스턴스를 시작하여 이를 대체할 수 있습니다. 또한 여러 가용 영역을 사용하도록 Amazon EC2 Auto Scaling을 구성할 수 있습니다. 한 가용 영역을 사용할 수 없게 되면 Amazon EC2 Auto Scaling은 이를 보완하기 위해 다른 가용 영역에서 인스턴스를 시작할 수 있습니다.
- 가용성 향상. Amazon EC2 Auto Scaling은 애플리케이션이 현재의 트래픽 수요를 처리할 수 있는 적절한 용량을 항상 확보할 수 있도록 지원합니다.
- 비용 관리 향상. Amazon EC2 Auto Scaling은 필요에 따라 용량을 동적으로 늘리거나 줄일 수 있습니다. 사용한 EC2 인스턴스에 대해 비용을 지불하므로 필요할 때 인스턴스를 시작하고 필요하지 않을 때 종료하여 비용을 절약할 수 있습니다.

내용

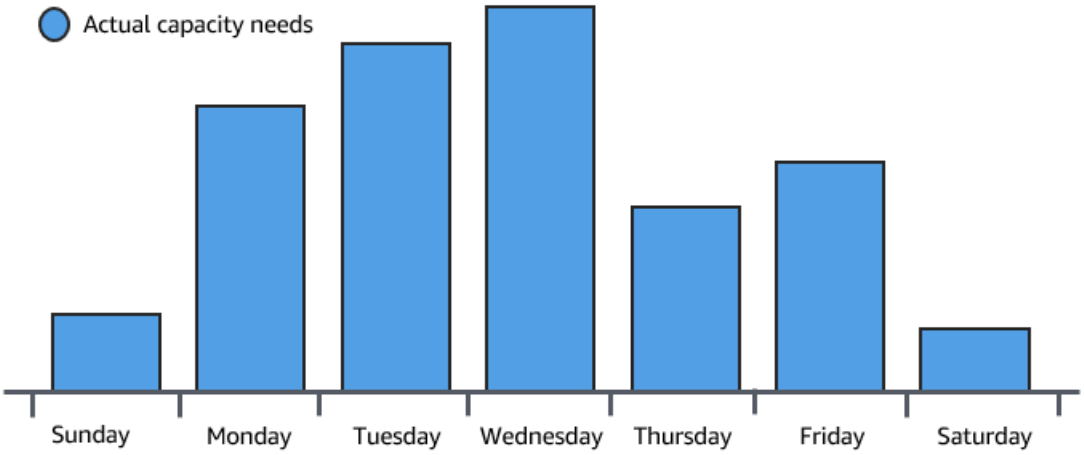
- 예: 가변적인 수요에 대응

- 예: 웹 앱 아키텍처
- 예: 가용 영역 전반에 인스턴스 분산
  - 인스턴스 분산
  - 재조정 활동

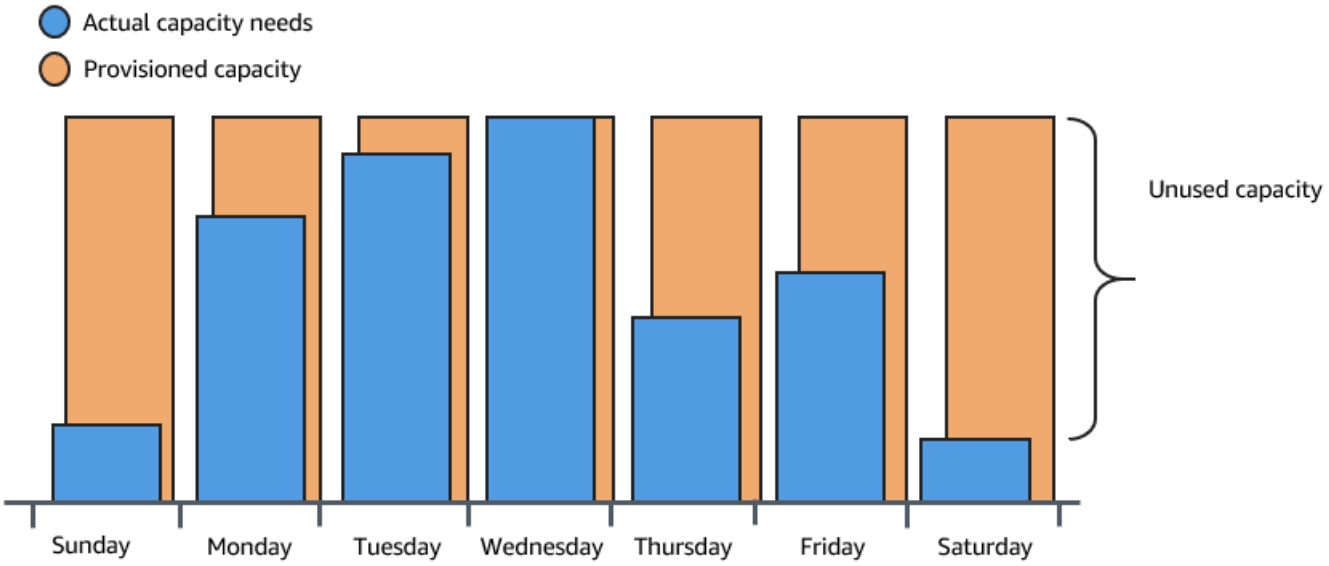
예: 가변적인 수요에 대응

Amazon EC2 Auto Scaling의 몇 가지 이점을 설명하기 위해 에서 실행되는 기본 웹 애플리케이션을 고려해 보십시오 AWS. 이 애플리케이션을 사용하여 직원들은 회의에 사용하려는 회의실을 찾을 수 있습니다. 한 주의 시작과 끝에는 이 애플리케이션의 사용량이 최소 수준입니다. 주 중반쯤에는 더 많은 직원이 회의 일정을 잡고 있으며, 따라서 애플리케이션에 대한 수요가 눈에 띄게 증가합니다.

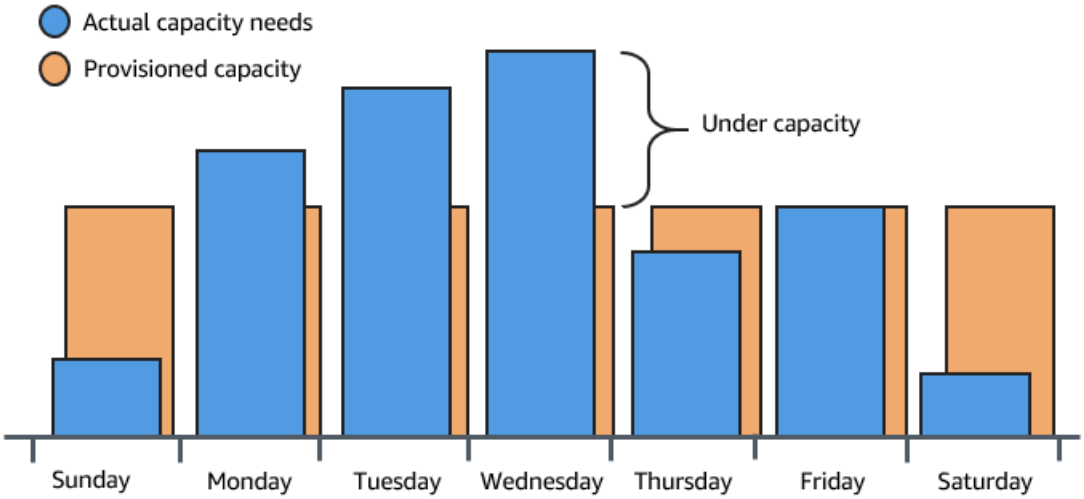
다음 그래프는 일주일 동안 애플리케이션의 용량이 얼마나 사용되었는지를 나타냅니다.



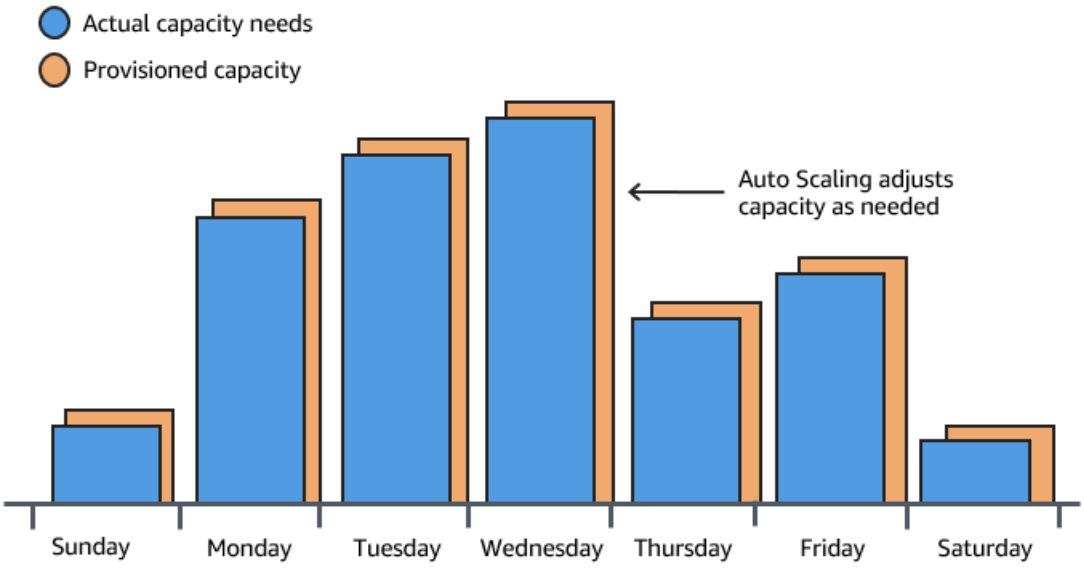
일반적으로 이러한 용량 변동을 계획하는 방법에는 두 가지가 있습니다. 첫 번째 옵션은 충분한 서버를 추가하여 애플리케이션에 항상 수요를 충족할 만큼의 충분한 용량이 확보되도록 하는 것입니다. 하지만 이 옵션의 단점은 애플리케이션에 이만큼의 용량이 필요 없는 날들도 있다는 것입니다. 여분의 용량이 미사용으로 남아 있다는 것은 결국 애플리케이션 실행 유지 비용의 상승을 의미합니다.



두 번째 옵션은 애플리케이션에 대한 평균 수요를 처리할 수 있는 충분한 용량이 확보되도록 하는 것입니다. 이 옵션을 선택하면 가끔 사용되는 장비를 구매하지 않아도 되므로 비용이 절감됩니다. 그러나 애플리케이션에 대한 수요가 용량을 초과할 경우, 부정적인 고객 경험을 초래할 위험이 있습니다.



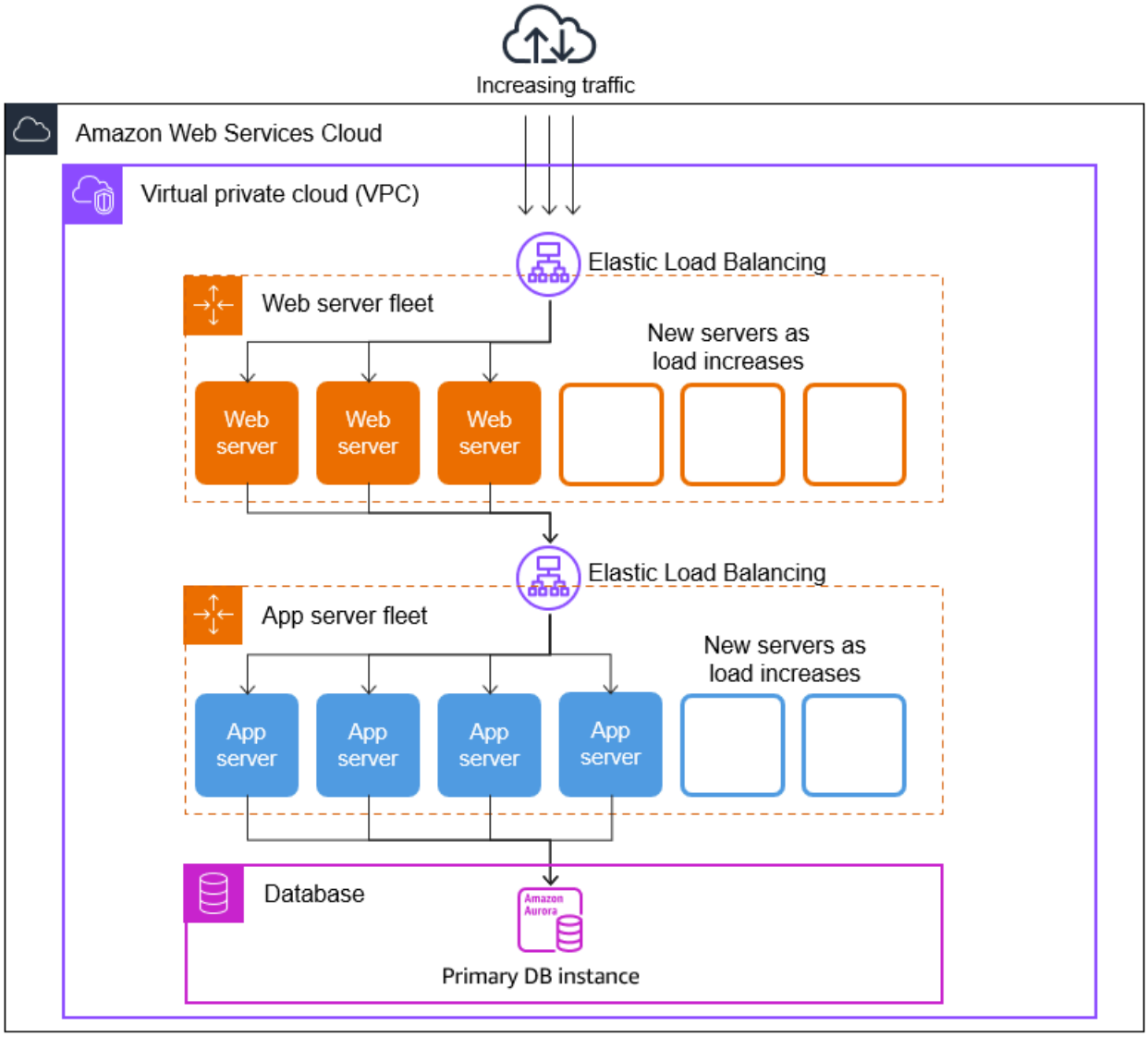
Amazon EC2 Auto Scaling을 이 애플리케이션에 추가하면 세 번째 옵션을 사용할 수 있습니다. 즉, 필요 시에만 애플리케이션에 새 인스턴스를 추가하고 더 이상 필요 없을 때 이를 해지할 수 있습니다. Amazon EC2 Auto Scaling은 EC2 인스턴스를 사용하기 때문에 사용자는 해당 인스턴스를 사용할 때 사용한 인스턴스에 대한 비용만 지불하면 됩니다. 이제 여러분은 비용을 최소화하면서도 최상의 고객 경험을 제공하는 비용 효율적인 아키텍처를 갖게 되었습니다.



예: 웹 앱 아키텍처

일반적인 웹 앱 시나리오에서는 고객 트래픽 볼륨을 처리하기 위해 여러 개의 앱 사본을 동시에 실행합니다. 이러한 여러 애플리케이션 사본은 동일한 EC2 인스턴스 (클라우드 서버) 에서 호스팅되며 각각 고객 요청을 처리합니다.

Amazon EC2 Auto Scaling은 사용자를 대신하여 이러한 EC2 인스턴스의 시작 및 종료를 관리합니다. Auto Scaling 그룹이 EC2 인스턴스를 시작하거나 종료하는 시기를 결정하는 일련의 기준 (예: Amazon CloudWatch 경보) 을 정의합니다. Auto Scaling 그룹을 네트워크 아키텍처에 추가하면 애플리케이션의 가용성과 내결함성을 향상시킬 수 있습니다.



필요한 만큼 Auto Scaling 그룹을 생성할 수 있습니다. 예컨대, 각 티어별로 Auto Scaling 그룹을 생성할 수 있습니다.

Auto Scaling 그룹의 인스턴스 간 트래픽을 분산하기 위해 로드 밸런서를 아키텍처에 도입할 수 있습니다. 자세한 설명은 Elastic Load Balancing 섹션을 참조하세요.

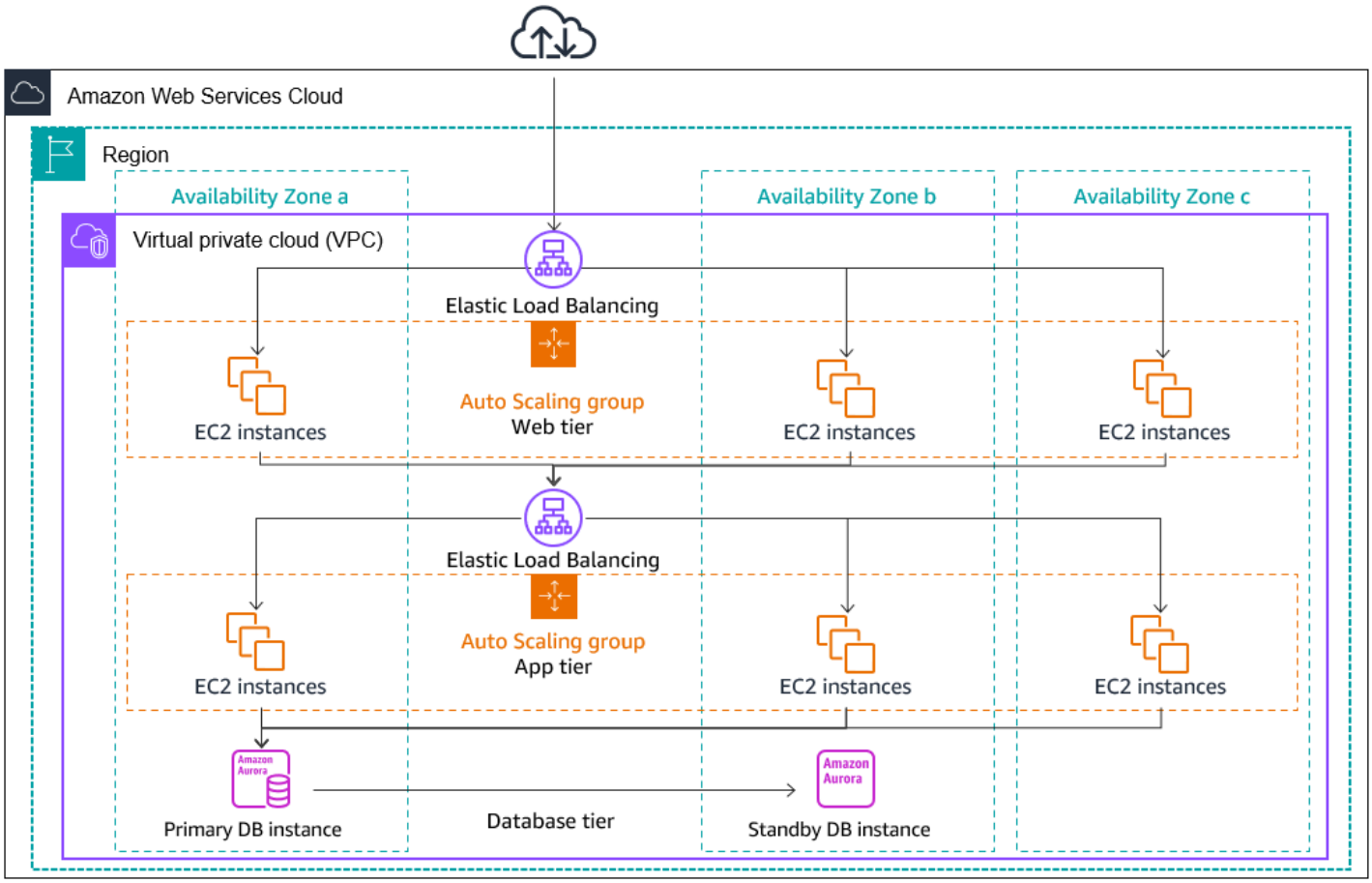
예: 가용 영역 전반에 인스턴스 분산

가용 영역은 주어진 AWS 리전내에 있는 격리된 위치입니다. 각 지역에는 지역에 대한 고가용성을 제공하도록 설계된 여러 가용 영역이 있습니다. 가용 영역은 독립적이므로 여러 영역을 사용하도록 애플리케이션을 설계하면 애플리케이션 가용성이 향상됩니다. 자세한 내용은 Amazon EC2 Auto Scaling의 레질리언스 단원을 참조하십시오.

가용 영역은 AWS 리전 코드와 문자 식별자 (예:`us-east-1a`) 로 식별됩니다. 기본값을 VPC 사용하지 않고 VPC 및 서브넷을 생성하는 경우 각 가용 영역에서 하나 이상의 서브넷을 정의할 수 있습니다. 각 서브넷은 단일 가용 영역 내에서만 존재해야 하며, 여러 영역으로 스케일 아웃할 수 없습니다. 자세한 내용은 Amazon VPC사용 설명서의 Amazon VPC 작동 방식을 참조하십시오.

Auto Scaling 그룹을 생성할 때는 Auto Scaling 그룹을 배포할 VPC 서브넷과 서브넷을 선택해야 합니다. Amazon EC2 Auto Scaling은 선택한 서브넷에서 인스턴스를 생성합니다. 따라서 각 인스턴스는 Amazon EC2 Auto Scaling에서 선택한 특정 가용 영역과 연결됩니다. Amazon EC2 Auto Scaling은 인스턴스가 시작되면 고가용성과 안정성을 위해 영역 간에 인스턴스를 균등하게 분배하려고 합니다.

다음 이미지는 3개의 가용 영역에 배치된 다중 계층 아키텍처의 개요를 보여 줍니다.



인스턴스 분산

Amazon EC2 Auto Scaling은 활성화된 각 가용 영역에서 동일한 수의 인스턴스를 자동으로 유지하려고 합니다. Amazon EC2 Auto Scaling은 가장 적은 수의 인스턴스로 가용 영역에서 새 인스턴스를 시작하려고 시도하여 이를 수행합니다. 가용 영역에 서브넷을 여러 개 선택한 경우 Amazon EC2 Auto

Scaling은 가용 영역에서 무작위로 서브넷을 선택합니다. 그러나 시도가 실패할 경우 Amazon EC2 Auto Scaling은 성공할 때까지 다른 가용 영역에서 인스턴스를 시작하려고 시도합니다.

가용 영역이 비정상이거나 사용할 수 없는 상황에서는 인스턴스가 가용 영역 전체에 고르지 않게 배치될 수 있습니다. 가용 영역이 복구되면 Amazon EC2 Auto Scaling은 Auto Scaling 그룹을 자동으로 재조정합니다. .인스턴스가 가장 적은 활성화된 가용 영역에서 인스턴스를 시작하고 다른 곳에서는 인스턴스를 해지하여 이를 수행합니다.

재조정 활동

재조정 활동은 가용 영역 재조정 및 용량 재조정라는 두 가지 카테고리로 나뉩니다.

가용 영역 재조정

특정 작업이 발생하면 Auto Scaling 그룹의 가용 영역 간에 불균형이 발생할 수 있습니다. Amazon EC2 Auto Scaling은 가용 영역을 재조정하여 이를 보완합니다. 다음 작업으로 인해 재조정 활동이 발생할 수 있습니다.

- Auto Scaling 그룹과 연결된 가용 영역을 변경합니다.
- 인스턴스를 명시적으로 해지 또는 분리하거나 인스턴스를 대기 상태로 두면 그룹이 불균형 상태가 됩니다.
- 이전에 용량이 부족했던 가용 영역이 복구되어 이제 추가적인 용량이 있습니다.
- 이전에 스팟 가격이 최대 가격보다 높았던 가용 영역이 이제 스팟 가격이 하락하여 최대 가격 아래로 떨어진 경우

재조정 시 Amazon EC2 Auto Scaling은 이전 인스턴스를 종료하기 전에 새 인스턴스를 시작합니다. 이렇게 하면 재조정가 애플리케이션의 성능이나 가용성을 손상시키지 않습니다.

Amazon EC2 Auto Scaling은 이전 인스턴스를 종료하기 전에 새 인스턴스를 시작하려고 시도하기 때문에 지정된 최대 용량에 도달하거나 근처에 있으면 재조정 작업이 방해되거나 완전히 중단될 수 있습니다.

이 문제를 피하기 위해 시스템은 재조정 활동 중에 그룹의 지정된 최대 용량을 일시적으로 초과할 수 있습니다. 기본적으로, 10% 또는 하나의 인스턴스 중 더 큰 차이에 의해 이를 수행할 수 있습니다. 마진은 그룹이 최대 용량이거나 거의 최대 용량에 도달하고 재조정가 필요한 경우에만 스케일 아웃됩니다. 스케일 아웃은 그룹에 재조정가 필요한 동안에만 유지됩니다(대개 몇 분 정도).

또는 인스턴스 유지 관리 정책을 사용하여 Auto Scaling 그룹에 대한 임계값을 설정할 수 있으며, 그룹은 해당 임계값 범위 내에서만 용량을 늘리거나 줄일 수 있습니다. 이렇게 하면 그룹이 자체적으로 재조정되는 속도를 제어할 수 있습니다. 자세한 설명은 인스턴스 유지 관리 정책 섹션을 참조하세요.

용량 재조정

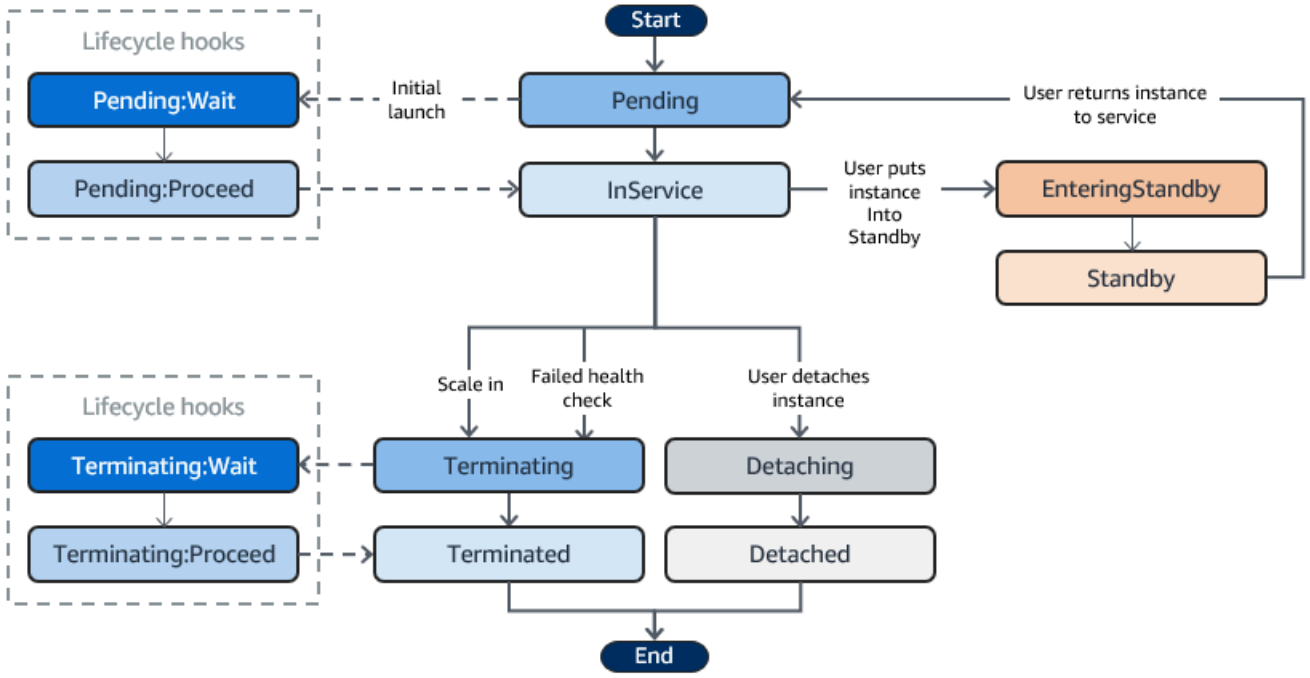
스팟 인스턴스를 사용할 때 Auto Scaling 그룹에 대해 용량 재조정을 켤 수 있습니다. 이를 통해 Amazon EC2 Auto Scaling은 스팟 인스턴스가 중단 위험이 높다고 EC2 보고할 때마다 스팟 인스턴스를 시작하려고 시도할 수 있습니다. 새 인스턴스를 시작한 다음 이전 인스턴스를 해지합니다. 자세한 내용은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리 단원을 참조하십시오.

아마존 EC2 Auto Scaling 인스턴스 라이프사이클

Auto Scaling 그룹의 EC2 인스턴스에는 다른 EC2 인스턴스의 경로나 수명 주기가 다릅니다. 수명 주기는 Auto Scaling 그룹이 인스턴스를 시작하고 서비스에 들어갈 때 시작됩니다. 수명 주기는 인스턴스를 해지하거나 Auto Scaling 그룹이 인스턴스를 서비스에서 제외시키고 이를 해지할 때 끝납니다.

> **Note**
> 인스턴스가 시작되는 즉시 인스턴스에 대한 요금이 청구되며, 아직 서비스되지 않는 시간도 포함됩니다.

다음 그림은 Amazon EC2 Auto Scaling 수명 주기의 인스턴스 상태 간 전환을 보여줍니다.



스케일 아웃

다음 스케일 아웃 이벤트는 Auto Scaling 그룹이 EC2 인스턴스를 시작하고 그룹에 연결하도록 지시합니다.

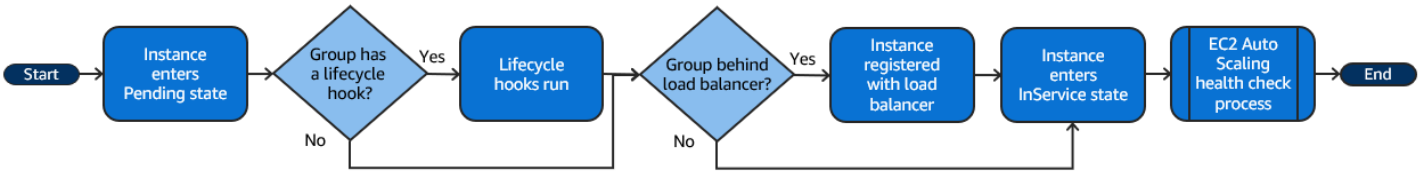
- 그룹의 크기를 수동으로 늘립니다. 자세한 내용은 기존 Auto Scaling 그룹의 원하는 용량 변경 단원을 참조하십시오.
- 지정된 수요 증가에 따라 그룹의 크기를 자동으로 늘리는 조정 정책을 만듭니다. 자세한 설명은 Amazon EC2 Auto Scaling을 위한 동적 스케일링 섹션을 참조하세요.
- 특정 시간에 그룹의 크기를 늘리도록 조정을 일정 기반으로 설정합니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 스케일링 예약 단원을 참조하십시오.

스케일 아웃 이벤트가 발생하면 Auto Scaling 그룹은 할당된 시작 템플릿을 사용하여 필요한 수의 EC2 인스턴스를 시작합니다. 이러한 인스턴스는 `Pending` 상태에서 시작됩니다. Auto Scaling 그룹에 수명 주기 후크를 추가하면 여기에서 사용자 지정 작업을 수행할 수 있습니다. 자세한 내용은 수명 주기 후크 단원을 참조하십시오.

각 인스턴스가 완전히 구성되어 Amazon EC2 상태 확인을 통과하면 Auto Scaling 그룹에 연결되고 `InService` 상태가 됩니다. 이 인스턴스는 원하는 Auto Scaling 그룹 용량에서 감산됩니다.

Auto Scaling 그룹이 Elastic Load Balancing 로드 밸런서로부터 트래픽을 수신하도록 구성된 경우, Amazon EC2 Auto Scaling은 인스턴스를 로 표시하기 전에 로드 밸런서에 인스턴스를 자동으로 등록합니다. `InService`

다음은 스케일 아웃 이벤트를 위해 로드 밸런서에 인스턴스를 등록하는 단계를 요약한 것입니다.



서비스 상태의 인스턴스

인스턴스는 다음 중 하나가 발생할 때까지 `InService` 상태로 유지됩니다.

- 스케일 인 이벤트가 발생하고 Amazon EC2 Auto Scaling은 Auto Scaling 그룹의 크기를 줄이기 위해 이 인스턴스를 종료하기로 선택합니다. 자세한 내용은 축소 시 해지할 Auto Scaling 인스턴스 제어 단원을 참조하십시오.
- 인스턴스를 `Standby` 상태로 설정합니다. 자세한 설명은 대기 모드 시작 및 해지 섹션을 참조하세요.
- Auto Scaling 그룹에서 인스턴스를 분리합니다. 자세한 내용은 Auto Scaling 그룹에서 인스턴스 분리 또는 연결 단원을 참조하십시오.
- 인스턴스가 필요한 수의 건전성 체크에 실패한 경우, Auto Scaling 그룹에서 제거, 해지 및 교체됩니다. 자세한 설명은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 섹션을 참조하세요.

축소

다음 스케일 인 이벤트는 Auto Scaling 그룹이 그룹에서 EC2 인스턴스를 분리하고 종료하도록 지시합니다.

- 그룹의 크기를 수동으로 줄입니다. 자세한 내용은 기존 Auto Scaling 그룹의 원하는 용량 변경 단원을 참조하십시오.
- 지정된 수요 감소에 따라 그룹의 크기를 자동으로 줄이는 조정 정책을 만듭니다. 자세한 설명은 Amazon EC2 Auto Scaling을 위한 동적 스케일링 섹션을 참조하세요.
- 특정 시간에 그룹의 크기를 줄이도록 조정을 일정 기반으로 설정합니다. 자세한 설명은 Amazon EC2 Auto Scaling을 위한 스케일링 예약 섹션을 참조하세요.

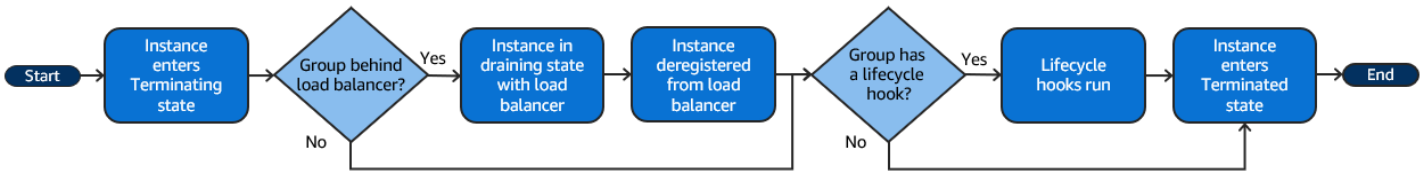
생성한 스케일 아웃 이벤트 각각에 대해 이에 상응하는 축소 이벤트를 생성하는 것이 중요합니다. 이렇게 하면 애플리케이션에 할당된 리소스와 그러한 리소스의 수요를 가능한 한 가깝게 일치시킬 수 있습니다.

축소 이벤트가 발생하면 Auto Scaling 그룹에서 하나 이상의 인스턴스를 해지합니다. Auto Scaling 그룹이 해지 정책을 사용하여 해지할 인스턴스를 결정합니다. Auto Scaling 그룹에서 해지되는 과정에 있는 인스턴스는 `Terminating` 상태로 전환되며, 다시 서비스 상태로 돌아갈 수 없습니다.

Auto Scaling 그룹이 Elastic Load Balancing 로드 밸런서로부터 트래픽을 수신하도록 구성된 경우 Amazon EC2 Auto Scaling은 종료 인스턴스를 로드 밸런서에서 자동으로 등록 취소합니다. 인스턴스를 등록 취소하면 모든 새 요청이 로드 밸런서의 대상 그룹에 있는 다른 인스턴스로 리디렉션되고 기존 인스턴스 연결은 등록 취소 지연이 만료될 때까지 계속될 수 있습니다.

Auto Scaling 그룹에 수명 주기 후크를 추가하면 해지 인스턴스에서 사용자 지정 작업을 수행할 수 있습니다. 자세한 설명은 수명 주기 후크 섹션을 참조하세요. 마지막으로 인스턴스가 완전히 해지되고 `Terminated` 상태로 들어갑니다.

다음은 스케일 인 이벤트를 위해 로드 밸런서에 인스턴스를 등록 취소하는 단계를 요약한 것입니다.



인스턴스 분리

Auto Scaling 그룹에서 인스턴스를 분리할 수 있습니다. 인스턴스를 분리한 후에는 이를 Auto Scaling 그룹과 별도로 관리하거나 다른 Auto Scaling 그룹에 연결할 수 있습니다.

자세한 내용은 Auto Scaling 그룹에서 인스턴스 분리 또는 연결 단원을 참조하십시오.

인스턴스 연결

특정 기준을 충족하는 실행 중인 EC2 인스턴스를 Auto Scaling 그룹에 연결할 수 있습니다. 인스턴스가 연결되면 Auto Scaling 그룹의 일부로 관리됩니다.

자세한 내용은 Auto Scaling 그룹에서 인스턴스 분리 또는 연결 단원을 참조하십시오.

수명 주기 후크

인스턴스를 시작하거나 해지할 때 사용자 지정 작업을 수행할 수 있도록 Auto Scaling 그룹에 수명 주기 후크를 추가할 수 있습니다.

Amazon EC2 Auto Scaling은 스케일 아웃 이벤트에 응답하면 하나 이상의 인스턴스를 시작합니다. 이러한 인스턴스는 `Pending` 상태에서 시작됩니다. Auto Scaling 그룹에 `autoscaling:EC2_INSTANCE_LAUNCHING` 수명 주기 후크를 추가한 경우, 인스턴스가 `Pending` 상태에서 `Pending:Wait` 상태로 이동합니다. 수명 주기 작업을 완료하면 인스턴스가 `Pending:Proceed` 상태로 들어갑니다. 인스턴스가 완전히 구성되면 Auto Scaling 그룹에 연결되고 `InService` 상태로 들어갑니다.

Amazon EC2 Auto Scaling이 스케일 인 이벤트에 응답하면 하나 이상의 인스턴스를 종료합니다. Auto Scaling 그룹에서 이러한 인스턴스가 분리되고 `Terminating` 상태로 들어갑니다. Auto Scaling 그룹에 `autoscaling:EC2_INSTANCE_TERMINATING` 수명 주기 후크를 추가한 경우, 인스턴스가 `Terminating` 상태에서 `Terminating:Wait` 상태로 이동합니다. 수명 주기 작업을 완료하면 인스턴스가 `Terminating:Proceed` 상태로 들어갑니다. 인스턴스가 완전히 해지되면 `Terminated` 상태로 들어갑니다.

자세한 설명은 Amazon EC2 Auto Scaling 수명 주기 후크 섹션을 참조하세요.

대기 모드 시작 및 해지

`InService` 상태인 인스턴스를 `Standby` 상태로 변경할 수 있습니다. 이를 통해 서비스에서 인스턴스를 제거하거나 문제를 해결하거나 변경한 다음 다시 서비스 상태로 되돌릴 수 있습니다.

`Standby` 상태의 인스턴스는 계속해서 Auto Scaling 그룹에서 관리됩니다. 그러나 이러한 인스턴스를 다시 서비스 상태로 되돌리기 전까지는 애플리케이션의 활성 부분이 아닙니다.

자세한 내용은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 단원을 참조하십시오.

Auto Scaling 리소스 및 그룹 할당량

AWS 계정 Your에는 각 서비스에 대한 기본 할당량 (이전에는 한도라고 함) 이 있습니다. AWS 다르게 표시되지 않는 한 리전별로 각 할당량이 적용됩니다. 일부 할당량에 대한 증가를 요청할 수 있으며 다른 할당량은 늘릴 수 없습니다.

Amazon EC2 Auto Scaling의 할당량을 보려면 서비스 할당량 콘솔을 여십시오. 탐색 창에서 AWS 서비스를 선택하고 Amazon EC2 Auto Scaling을 선택합니다.

할당량 증가를 요청하려면 Service Quotas 사용자 가이드의 할당량 증가 요청을 참조하세요. Service Quotas에서 아직 할당량을 사용할 수 없는 경우, Auto Scaling 제한 양식(Auto Scaling Limits form)을 사용합니다. 할당량 증가는 요청한 지역에서만 적용됩니다.

아마존 EC2 오토 스케일링 리소스

생성할 수 AWS 계정 있는 Auto Scaling 그룹 및 시작 구성의 수와 관련된 할당량은 다음과 같습니다.

Resource	기본 할당량
지역별 Auto Scaling 그룹	500
지역별 시작 구성	200

Auto Scaling 그룹 구성

Auto AWS 계정 Scaling 그룹 구성과 관련된 할당량은 다음과 같습니다. 변경할 수 없습니다.

리소스	할당량
Auto Scaling 그룹당 조정 정책	50
Auto Scaling 그룹당 예약된 작업	125
조정 정책 단계당 단계별 조정	20
Auto Scaling 그룹당 수명 주기 후크	50
SNSAuto Scaling 그룹별 주제	10
Auto Scaling 그룹당 Classic Load Balancer	50
Auto Scaling 그룹당 Elastic Load Balancing 대상 그룹	50
VPCAuto Scaling 그룹당 래티스 타겟 그룹	5

Auto Scaling 그룹 API 작업

Amazon EC2 Auto Scaling은 Auto Scaling 그룹을 일괄적으로 변경하는 API 작업을 제공합니다. 다음은 단일 작업에서 허용되는 최대 항목 수 (최대 배열 구성원) 에 대한 API 제한입니다. 변경할 수 없습니다.

Operation	최대 배열 구성원
AttachInstances	인스턴스 20개 IDs

Operation	최대 배열 구성원
AttachLoadBalancers	로드 밸런서 10개
AttachLoadBalancerTargetGroups	대상 그룹 10개
BatchDeleteScheduledAction	예약 작업 50개
BatchPutScheduledUpdateGroupAction	예약 작업 50개
DetachInstances	20개 인스턴스 IDs
DetachLoadBalancers	로드 밸런서 10개
DetachLoadBalancerTargetGroups	대상 그룹 10개
EnterStandby	20개 인스턴스 IDs
ExitStandby	20개 인스턴스 IDs
SetInstanceProtection	50개 인스턴스 IDs

Amazon EC2 Auto Scaling에 대한 요청 제한 API

Amazon EC2 Auto Scaling API 요청은 서비스 대역폭을 유지하기 위해 토큰 버킷 체계를 사용하여 조절됩니다. 자세한 내용은 Amazon EC2 Auto Scaling API 참조의 API 요청 비율을 참조하십시오.

EC2해지 요금

Amazon EC2 Auto Scaling은 Auto Scaling 그룹이 축소될 때 한 번에 수행할 수 있는 EC2 인스턴스 종료 작업 수를 동적으로 결정합니다. 즉, Auto Scaling 그룹에 따라 한 번에 해지되는 인스턴스 수에 차이가 있을 수 있습니다. 이러한 차이는 Amazon EC2 Auto Scaling에서 로드 밸런서를 사용하여 인스턴스 등록을 취소해야 하는지 여부와 같은 외부 고려 사항으로 인해 발생합니다.

기타 서비스

EC2Amazon VPC 및 Amazon과 같은 다른 서비스에 대한 할당량은 Auto Scaling 그룹에 영향을 미칠 수 있습니다. 를 Service Quotas 사용하여 EC2 인스턴스 및 내 다른 리소스의 할당량을 업데이트할 수 있습니다. AWS 계정 Service Quotas 콘솔에서 사용 가능한 모든 서비스 할당량을 확인하고 할당량 증

가를 요청할 수 있습니다. 자세한 설명은 Service Quotas 사용자 가이드에서 할당량 증가 요청을 참조하세요.

시작 템플릿에만 적용되는 할당량은 Amazon EC2 사용 설명서의 시작 템플릿 제한을 참조하십시오.

Amazon EC2 Auto Scaling을 사용하도록 설정

Amazon EC2 Auto Scaling을 사용하기 전에 먼저 다음 태스크를 완료합니다.

작업

- Amazon EC2 사용 준비
- 사용을 준비하세요. AWS CLI

Amazon EC2 사용 준비

Amazon EC2를 처음으로 사용하는 경우 Amazon EC2 문서에 나와 있는 태스크를 완료하세요. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2를 사용한 설정 또는 Amazon EC2 사용 설명서의 Amazon EC2를 사용한 설정을 참조하십시오.

사용을 준비하세요. AWS CLI

AWS 명령줄 도구를 사용하여 시스템 명령줄에서 명령을 실행하여 Amazon EC2 Auto Scaling 및 기타 AWS 작업을 수행할 수 있습니다.

AWS Command Line Interface (AWS CLI) 를 사용하려면 버전 1 또는 2를 다운로드, 설치 및 구성하십시오 AWS CLI. 동일한 Amazon EC2 Auto Scaling 기능을 버전 1과 2에서 사용할 수 있습니다. AWS CLI 버전 1을 설치하려면 AWS CLI 버전 1 사용 설명서의 AWS CLI설치, 업데이트 및 제거를 참조하세요. AWS CLI 버전 2를 설치하려면 버전 2 사용 설명서의 최신 버전 설치 또는 업데이트를 참조하십시오. AWS CLIAWS CLI

AWS CloudShell 개발 환경에 를 AWS CLI 설치하지 AWS Management Console 않고 대신 사용할 수 있습니다. 설치가 필요 없을 뿐만 아니라 보안 인증 정보를 구성하거나 지역을 지정할 필요도 없습니다. AWS Management Console 세션에서는 이 컨텍스트를 에 AWS CLI제공합니다. 지원되는 AWS CloudShell 환경에서 사용할 수 AWS 리전있습니다. 자세한 정보는 명령줄에서 다음을 사용하여 Auto Scaling 그룹을 생성합니다. AWS CloudShell을 참조하세요.

자세한 설명은 AWS CLI 명령 참조의 autoscaling을 참조하세요.

Amazon EC2 Auto Scaling 시작하기

Amazon EC2 Auto Scaling을 시작하려면 서비스를 소개하는 자습서를 따르십시오.

주제

- 튜토리얼: 첫 번째 Auto Scaling 그룹 생성
- 자습서: 조정 및 로드 밸런싱된 애플리케이션 설정

Auto Scaling 그룹에서 인스턴스의 수명 주기를 관리하기 위한 특정 도구에 초점을 맞춘 추가 자습서는 다음 주제를 참조하십시오.

- 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성. 이 자습서에서는 EventBridge Amazon을 사용하여 Auto Scaling 그룹의 인스턴스에 발생하는 이벤트를 기반으로 Lambda 함수를 호출하는 규칙을 생성하는 방법을 보여줍니다.
- 자습서: 데이터 스크립트와 인스턴스 메타데이터를 사용하여 수명 주기 상태 검색. 이 자습서는 인스턴스 메타데이터 서비스(IMDS)를 사용하여 인스턴스 자체 내에서 작업을 호출하는 방법을 보여 줍니다.

애플리케이션에 사용할 Auto Scaling 그룹을 생성하기 전에 AWS 클라우드에서 실행할 애플리케이션을 철저히 검토하세요. 다음을 고려하세요.

- Auto Scaling 그룹을 분산할 가용 영역 수
- 사용할 수 있는 기존 리소스(예: 보안 그룹 또는 Amazon Machine Image(AMI))
- 조정 기능을 사용하여 용량을 늘리거나 줄일지, 아니면 단지 일정한 수의 서버가 항상 실행되도록 하려고 하는지 Amazon EC2 Auto Scaling은 두 기능을 동시에 수행할 수 있습니다.
- 애플리케이션의 성능과 가장 연관성이 높은 지표
- 서버를 시작하고 프로비저닝하는 데 걸리는 시간

애플리케이션을 잘 이해할수록 Auto Scaling 아키텍처를 더 효과적으로 구축할 수 있습니다.

튜토리얼: 첫 번째 Auto Scaling 그룹 생성

이 자습서에서는 를 통해 Amazon EC2 Auto Scaling에 대한 실습을 소개합니다. AWS Management Console EC2 인스턴스를 정의하는 시작 템플릿과 단일 인스턴스가 포함된 Auto Scaling 그룹을 생성합니다. Auto Scaling 그룹을 시작한 후 인스턴스를 종료하고 해당 인스턴스가 서비스에서 제거되고 교

체되었는지 확인합니다. 인스턴스 수를 일정하게 유지하기 위해 Amazon EC2 Auto Scaling은 Amazon EC2 상태 및 연결성 확인을 자동으로 감지하여 응답합니다.

에 가입하면 프리 AWS티어를 사용하여 Amazon EC2 Auto Scaling을 무료로 시작할 수 있습니다.AWS 프리 티어를 사용하여 12개월 동안 무료로 `t2.micro` 인스턴스를 시작하고 사용할 수 있습니다(`t2.micro`를 사용할 수 없는 리전에서는 프리 티어에서 `t3.micro` 인스턴스를 사용할 수 있음). 프리 티어 외의 인스턴스를 시작하는 경우에는 인스턴스에 대하여 표준 Amazon EC2 사용 요금이 청구됩니다. 자세한 내용은 Amazon EC2 요금을 참조하세요.

Tasks

- 연습 준비
- 1단계: 출범 템플릿 생성
- 2단계: 단일 인스턴스 Auto Scaling 그룹 생성
- 3단계: Auto Scaling 그룹 확인
- 4단계: Auto Scaling 그룹에서 인스턴스 해지
- 5단계: 다음 절차
- 6단계: 정리

연습 준비

이 연습에서는 사용자가 EC2 인스턴스 출범 작업에 익숙하고 키 페어와 보안 그룹을 이미 생성한 것으로 가정합니다. 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2를 사용한 설정을 참조하십시오.

Amazon EC2 Auto Scaling을 사용하기 시작하려면 기본 VPC를 사용할 수 있습니다. AWS 계정기본 VPC에는 각 가용 영역의 기본 퍼블릭 서브넷과 VPC에 연결된 인터넷 게이트웨이가 포함됩니다. Amazon Virtual Private Cloud(Amazon VPC) 콘솔의 VPC 페이지에서 VPC를 확인할 수 있습니다.

1단계: 출범 템플릿 생성

이 단계에서는 Amazon EC2 Auto Scaling에서 자동으로 생성하는 EC2 인스턴스 유형을 지정하는 시작 템플릿을 생성합니다. 사용할 Amazon Machine Image(AMI) ID, 인스턴스 타입, 키 페어 및 보안 그룹 같은 정보를 포함시킵니다.

출범 템플릿 생성

1. Amazon EC2 콘솔을 열고 시작 템플릿 페이지로 이동합니다.

2. 상단의 탐색 모음에서 AWS 리전을 선택합니다. 생성한 출범 템플릿과 Auto Scaling 그룹은 지정한 지역에 연결됩니다.
3. Create launch template(출범 템플릿 생성)을 선택합니다.
4. Launch template name(출범 템플릿 이름)에 **`my-template-for-auto-scaling`**을 입력합니다.
5. Auto Scaling guidance(Auto Scaling 지침)에서 확인란을 선택합니다.
6. Application and OS Images(Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))의 경우, Quick Start(빠른 시작) 목록에서 Amazon Linux 2(HVM) 버전을 선택합니다. AMI는 인스턴스의 기본 구성 템플릿 역할을 합니다.
7. Instance type(인스턴스 타입)에서 지정한 AMI와 호환되는 하드웨어 구성을 선택합니다.
8. (옵션) Key pair (login)(키 페어(로그인))에서 기존 키 페어를 선택합니다. 키 페어를 사용하여 SSH를 통해 Amazon EC2 인스턴스에 연결합니다. 인스턴스 연결은 이 자습서의 내용에 포함되어 있지 않습니다. 따라서 SSH를 사용하여 인스턴스에 연결하려는 경우가 아닌 한 키 페어를 지정할 필요가 없습니다.
9. 네트워크 설정(Network settings)에서 고급 네트워크 구성(Advanced network configuration)을 스케일 아웃하고 다음을 수행하십시오:
   a. 네트워크 인터페이스 추가(Add network interface)를 선택하여 기본 네트워크 인터페이스를 구성합니다.
   b. 퍼블릭 IP 자동 할당의 경우 인스턴스가 퍼블릭 IPv4 주소를 수신할지 여부를 지정합니다. 기본적으로 Amazon EC2는 EC2 인스턴스가 기본 서브넷에서 시작되거나 퍼블릭 IPv4 주소를 자동으로 할당하도록 구성된 서브넷에서 인스턴스가 시작되는 경우 퍼블릭 IPv4 주소를 할당합니다. 인스턴스에 연결할 필요가 없는 경우 [Disable] 을 선택합니다.
   c. 보안 그룹 ID의 경우, Auto Scaling 그룹의 VPC로 사용할 계획인 동일한 VPC의 보안 그룹을 선택합니다. 보안 그룹을 지정하지 않을 경우, 인스턴스가 VPC의 기본 보안 그룹에 자동으로 연결됩니다.
   d. 종료 시 삭제의 경우 [Yes] 를 선택하여 인스턴스 삭제 시 네트워크 인터페이스를 삭제합니다.
10. Create launch template(출범 템플릿 생성)을 선택합니다.
11. 확인 페이지에서 Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

2단계: 단일 인스턴스 Auto Scaling 그룹 생성

다음 절차를 사용하여 시작 템플릿을 만든 후 중단한 부분부터 계속 진행하십시오.

Auto Scaling 그룹 생성

1. 출범 템플릿 또는 구성 선택 페이지에서 Auto Scaling 그룹 명칭에 **`my-first-asg`**를 입력합니다.
2. Next(다음)을 선택합니다.

   Auto Scaling 그룹에서 사용할 VPC 네트워크 설정을 선택할 수 있고 온디맨드 및 스팟 인스턴스를 시작하기 위한 옵션을 제공하는 인스턴스 시작 옵션 선택 페이지가 나타납니다.
3. 네트워크 섹션에서 VPC를 선택한 AWS 리전기본 VPC로 설정하거나 자체 VPC를 선택합니다. 기본 VPC는 인스턴스에 인터넷 연결을 제공하도록 자동으로 구성됩니다. 이 VPC는 해당 지역의 각 가용 영역에 퍼블릭 서브넷을 포함시킵니다.
4. Availability Zones and subnets(가용 영역 및 서브넷)에서 포함할 각 가용 영역의 서브넷을 선택합니다. 여러 가용 영역의 서브넷을 사용하여 가용성을 높일 수 있습니다. 자세한 정보는 VPC 서브넷 선택 시 고려 사항을 참조하세요.
5. 인스턴스 타입 요건(Instance type requirements) 섹션에서 기본 설정을 사용하여 이 단계를 단순화합니다. (출범 템플릿을 재정의하지 마세요.) 이 자습서에서는 출범 템플릿에 지정된 인스턴스 타입을 사용하여 하나의 온디맨드 인스턴스만 시작합니다.
6. 이 자습서의 나머지 기본값을 유지하고 Skip to review(검토로 이동)를 선택합니다.

   > **Note**
   >
   > 그룹의 초기 크기는 원하는 용량에 따라 결정됩니다. 기본값은 1 인스턴스입니다.

7. 검토 페이지에서 그룹에 대한 정보를 검토한 다음 Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

3단계: Auto Scaling 그룹 확인

이제 Auto Scaling 그룹을 생성했으므로 해당 그룹에서 EC2 인스턴스를 출범했는지 확인할 준비가 완료되었습니다.

> **Tip**
>
> 다음 절차에서는 Auto Scaling 그룹에 대한 Activity history(활동 기록) 및 Instances(인스턴스) 섹션을 살펴봅니다. 두 섹션 모두에 명명된 열이 이미 표시되어 있어야 합니다. 숨겨진 열을 표시하거나 표시되는 행 수를 변경하려면 각 섹션의 오른쪽 상단에 있는 톱니바퀴 아이콘을 선택하여 기본 설정 모달을 열고 필요에 따라 설정을 업데이트한 다음 Confirm(확인)을 선택합니다.

Auto Scaling 그룹에서 EC2 인스턴스를 출범했는지 확인하려면

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. 방금 생성한 Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다. 사용 가능한 첫 번째 탭은 Details(세부 정보) 탭으로, 여기에는 Auto Scaling 그룹에 대한 정보가 표시됩니다.
3. 두 번째 탭인 Activity(활동)를 선택합니다. Activity history(활동 기록)에서 Auto Scaling 그룹과 연결된 활동의 진행률을 볼 수 있습니다. 상태 열에는 인스턴스의 현재 상태가 표시됩니다. 인스턴스가 시작되는 동안 상태 열에 `Not yet in service`가 표시됩니다. 인스턴스가 시작되면 상태가 `Successful`로 변경됩니다. 새로 고침 버튼을 사용하여 인스턴스의 현재 상태를 볼 수도 있습니다.
4. 인스턴스 관리 탭의 인스턴스에서 인스턴스의 상태를 볼 수 있습니다.
5. 인스턴스가 성공적으로 시작되었는지 확인합니다. 인스턴스를 출범하는 데 약간 시간이 걸립니다.

   - 라이프사이클 열에는 인스턴스의 상태가 표시됩니다. 처음에는 인스턴스가 `Pending` 상태로 되어 있습니다. 인스턴스가 트래픽을 수신할 준비가 되면 상태가 `InService`로 전환됩니다.
   - 상태 열에는 인스턴스에 대한 Amazon EC2 Auto Scaling 상태 확인 결과가 표시됩니다.

4단계: Auto Scaling 그룹에서 인스턴스 해지

다음 단계를 사용하여 Amazon EC2 Auto Scaling 작동 방식, 특히 필요 시 새 인스턴스를 출범하는 방법에 대해 자세히 알아볼 수 있습니다. 이 자습서에서 생성한 Auto Scaling 그룹의 최소 크기는 하나의 인스턴스입니다. 따라서 실행 중인 인스턴스를 해지하는 경우에는 Amazon EC2 Auto Scaling에서 새 인스턴스를 출범해서 교체해야 합니다.

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.
3. 인스턴스 관리 탭의 인스턴스에서 인스턴스의 ID를 선택합니다.

   그러면 인스턴스를 해지할 수 있는 Amazon EC2 콘솔의 인스턴스 페이지로 이동합니다.
4. Actions(작업), Instance State(인스턴스 상태), Terminate(해지)를 차례로 선택합니다. 확인 메시지가 나타나면 Yes, Terminate(예, 해지)를 선택합니다.
5. 탐색 창의 Auto Scaling에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다. Auto Scaling 그룹을 선택하고 활동 탭을 선택합니다.

인스턴스 페이지에서 인스턴스를 종료하면 인스턴스를 종료한 후 새 인스턴스가 시작되기까지 1~2분 정도 걸립니다. 활동 기록에서 크기 조정 활동이 시작되면 첫 번째 인스턴스를 해지하기 위한 항목과 새 인스턴스를 출범하기 위한 항목이 표시됩니다. 새 항목이 표시될 때까지 새로 고침 버튼을 사용하십시오.

6. 인스턴스 관리 탭의 인스턴스 섹션에 새 인스턴스만 표시됩니다.
7. 탐색 창의 Instances(인스턴스)에서 Isntances(인스턴스)를 선택합니다. 이 페이지에는 해지된 인스턴스와 실행 중인 새로운 인스턴스가 모두 표시됩니다.

5단계: 다음 절차

방금 생성한 기본 인프라를 삭제하려면 다음 단계로 이동하십시오. 그렇지 않으면 이 인프라를 기본으로 사용하여 다음 중 하나 이상을 시도해 볼 수 있습니다.

- 세션 관리자 또는 SSH를 사용하여 Linux 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 세션 관리자를 사용하여 Linux 인스턴스에 연결 및 SSH를 사용하여 Linux 또는 macOS에서 Linux 인스턴스에 연결을 참조하십시오.
- Auto Scaling 그룹이 인스턴스를 출범하거나 해지할 때마다 알리도록 Amazon SNS 알림을 구성합니다. 자세한 설명은 아마존 SNS 알림 옵션 섹션을 참조하세요.
- Auto Scaling 그룹을 수동으로 조정하여 SNS 알림을 테스트합니다. 자세한 정보는 귀하의 Auto Scaling 그룹의 원하는 용량 변경을 참조하세요.

대상 추적 조정 정책에 대해 읽고 Auto Scaling 그룹 개념을 익히기 시작할 수도 있습니다. 애플리케이션의 로드가 변경되면 Auto Scaling 그룹이 최소 및 최대 용량 제한 사이에서 그룹의 원하는 용량을 조정하여 자동으로 스케일 아웃(인스턴스 추가)하고 스케일 인(더 적은 수의 인스턴스 실행)할 수 있습니다. 이러한 제한 설정에 대한 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.

6단계: 정리

조정 인프라를 삭제하거나 Auto Scaling 그룹만 삭제하고 시작 템플릿은 나중에 사용할 수 있도록 보관할 수 있습니다.

AWS 프리 티어 내에 없는 인스턴스를 출범한 경우, 추가 비용을 방지하기 위해 인스턴스를 해지해야 합니다. 인스턴스를 해지하면 인스턴스와 연결된 데이터도 삭제됩니다.

Auto Scaling 그룹을 삭제하려면

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹(`my-first-asg`) 옆의 확인란을 선택합니다.
3. 삭제를 선택합니다.
4. 확인 메시지가 표시되면 **`delete`**를 입력하여 지정된 Auto Scaling 그룹 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

   Name(이름) 열의 로딩 아이콘은 Auto Scaling 그룹이 삭제 중임을 나타납니다. 삭제가 수행되면 Desired(필요), Min(최소), Max(최대) 열에 Auto Scaling 그룹에 대해 `0`개의 인스턴스가 표시됩니다. 인스턴스를 해지하고 그룹을 삭제하는 데 몇 분 정도 걸립니다. 목록을 새로 고침하여 상태를 확인합니다.

현재 출범 템플릿을 유지하려면 아래 절차를 건너뜁니다.

출범 템플릿을 삭제하려면

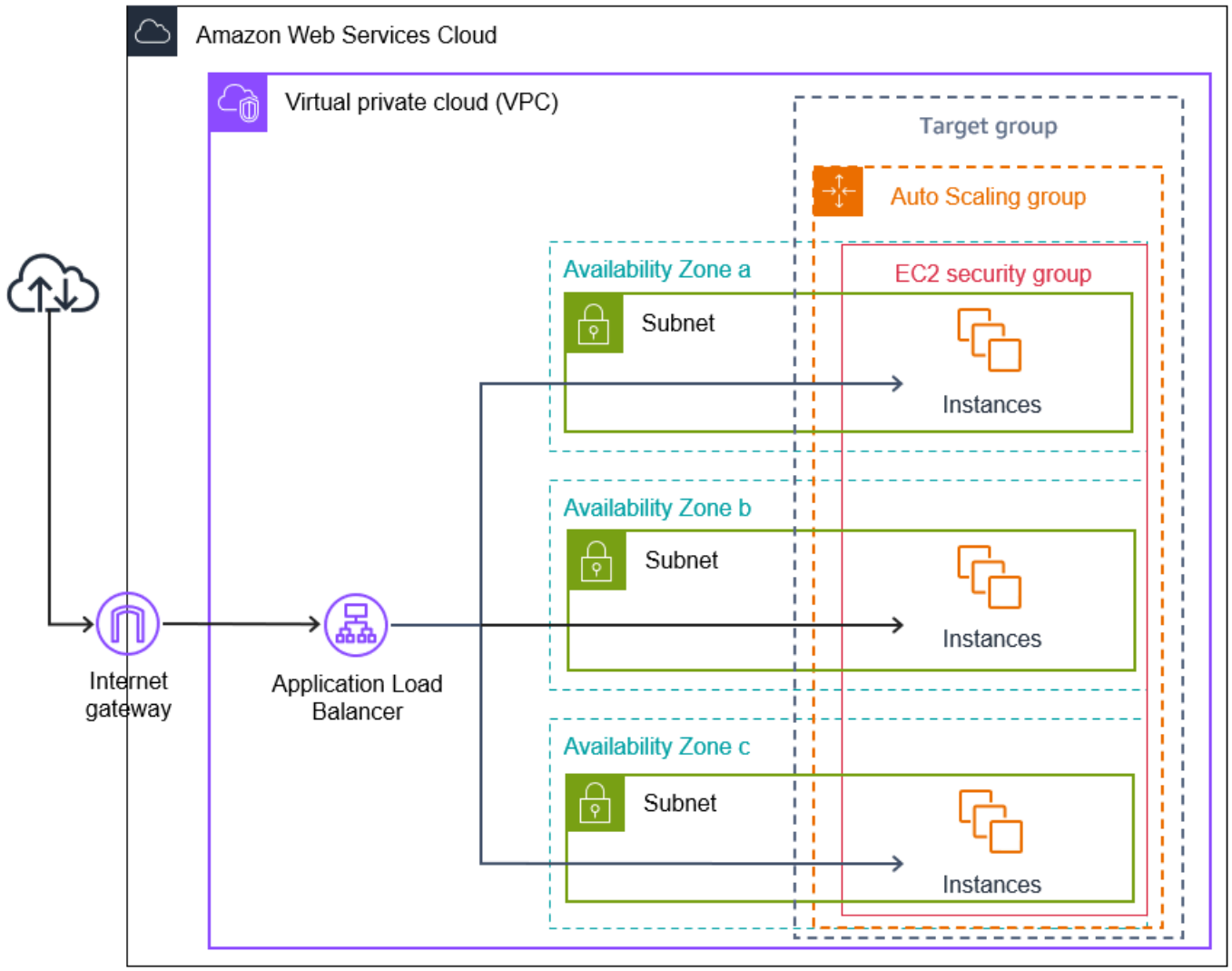
1. Amazon EC2 콘솔의 Launch templates(출범 템플릿) 페이지를 엽니다.
2. 출범 템플릿(`my-template-for-auto-scaling`)을 선택합니다.
3. Actions(작업)와 Delete template(템플릿 삭제)을 차례로 선택합니다.
4. 확인 메시지가 표시되면 **`Delete`**를 입력하여 지정된 출범 템플릿 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

자습서: 조정 및 로드 밸런싱된 애플리케이션 설정

> **Important**
>
> 이 자습서를 살펴보기 전에 먼저 다음 소개 자습서를 검토하는 것이 좋습니다. 첫 번째 Auto Scaling 그룹 만들기.

Auto Scaling 그룹을 Elastic Load Balancing 로드 밸런서에 등록하면 로드 밸런싱된 애플리케이션을 설정하는 데 도움이 됩니다. Elastic Load Balancing은 Amazon EC2 Auto Scaling과 함께 작동하여 들어오는 트래픽을 정상적인 Amazon EC2 인스턴스 전체에 분산합니다. 이렇게 하면 애플리케이션의 스케일 아웃성과 가용성이 향상됩니다. 여러 가용 영역 내에서 Elastic Load Balancing을 활성화하여 애플리케이션의 내결함성을 높일 수 있습니다.

이 자습서에서는 Auto Scaling 그룹을 만들 때 로드 밸런싱된 애플리케이션을 설정하는 기본 단계에 대해 설명합니다. 완료되면 아키텍처는 다음 다이어그램과 유사해야 합니다.



A: Elastic Load Balancing은 여러 유형의 로드 밸런서를 지원합니다. 이 자습서에서는 Application Load Balancer를 사용하는 것이 좋습니다.

아키텍처에 로드 밸런서를 도입하는 방법에 대한 자세한 설명은 Elastic Load Balancing을 사용하여 Auto Scaling 그룹에서 들어오는 애플리케이션 트래픽을 분산하십시오.을 참조하세요.

Tasks

- 사전 조건
- 1단계: 시작 템플릿 또는 시작 구성 설정
- 2단계: Auto Scaling 그룹 생성
- 3단계: 로드 밸런서가 연결됐는지 확인

- 4단계: 다음 단계
- 5단계: 정리
- 관련 리소스

사전 조건

- 로드 밸런서 및 대상 그룹. Auto Scaling 그룹에 사용할 로드 밸런서에 대해 동일한 가용 영역을 선택해야 합니다. 자세한 설명은 Elastic Load Balancing 사용자 가이드의 Elastic Load Balancing 시작하기를 참조하세요.
- 시작 템플릿 또는 시작 구성에 대한 보안 그룹. 보안 그룹은 리스너 포트 (일반적으로 HTTP 트래픽의 경우 포트 80) 와 Elastic Load Balancing이 상태 확인에 사용할 포트 모두에서 로드 밸런서로부터의 액세스를 허용해야 합니다. 자세한 설명은 다음과 같이 해당 문서를 참조하세요.
  - Application Load Balancer 사용자 가이드의 대상 보안 그룹
  - Network Load Balancer 사용자 가이드의 대상 보안 그룹

  필요에 따라 인스턴스에 퍼블릭 IP 주소가 있는 경우 인스턴스에 연결하는 데 SSH 트래픽을 허용할 수 있습니다.
- (선택 사항) 애플리케이션에 액세스 권한을 부여하는 IAM 역할 AWS.
- (선택 사항) Amazon EC2 인스턴스의 원본 템플릿으로 정의된 Amazon 머신 이미지 (AMI) 지금 생성하려면 인스턴스를 시작합니다. IAM역할 (생성한 경우) 과 사용자 데이터로 필요한 구성 스크립트를 지정합니다. 인스턴스에 연결하여 인스턴스를 사용자 지정합니다. 예를 들어, 소프트웨어 및 애플리케이션을 설치하고, 데이터를 복사하고, 추가 EBS 볼륨을 연결할 수 있습니다. 인스턴스에서 애플리케이션을 테스트하여 인스턴스가 올바르게 구성되었는지 확인합니다. 이 업데이트된 구성을 사용자 지정으로 저장합니다AMI. 이 인스턴스가 나중에 필요하지 않을 경우, 해지할 수 있습니다. 이 새 사용자 지정에서 시작된 인스턴스에는 생성 시 만든 사용자 지정이 AMI 포함됩니다. AMI
- 가상 사설 클라우드 (VPC). 이 자습서는 기본값을 VPC 참조하지만 직접 사용할 수도 있습니다. 자체 가용 영역을 사용하는 경우VPC, 작업 중인 지역의 각 가용 영역에 매핑된 서브넷이 있는지 확인하십시오. 로드 밸런서를 생성하려면 사용 가능한 퍼블릭 서브넷이 최소한 두 개 있어야 합니다. Auto Scaling 그룹을 생성한 다음 로드 밸런서에 등록하려면 프라이빗 서브넷 또는 퍼블릭 서브넷이 두 개 있어야 합니다.

1단계: 시작 템플릿 또는 시작 구성 설정

이 자습서에는 시작 템플릿 또는 시작 구성 중 하나를 사용합니다.

주제

- 시작 템플릿을 선택하거나 생성하십시오.
- 시작 구성 생성 또는 선택

시작 템플릿을 선택하거나 생성하십시오.

사용하려는 시작 템플릿이 이미 있으면 다음 절차를 사용하여 해당 구성을 선택합니다.

기존 시작 템플릿을 선택하려면

1. Amazon EC2 콘솔의 시작 템플릿 페이지를 엽니다.
2. 화면 상단의 탐색 모음에서 로드 밸런서가 생성된 지역을 선택합니다.
3. 시작 템플릿을 선택합니다.
4. Actions(작업), Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

또는 다음 절차를 사용하여 새 시작 템플릿을 생성합니다.

시작 템플릿 생성

1. Amazon EC2 콘솔의 시작 템플릿 페이지를 엽니다.
2. 화면 상단의 탐색 모음에서 로드 밸런서가 생성된 지역을 선택합니다.
3. Create launch template(시작 템플릿 생성)을 선택합니다.
4. 시작 템플릿의 이름을 입력하고 초기 버전에 대한 설명을 제공하세요.
5. 애플리케이션 및 OS 이미지 (Amazon 머신 이미지) 의 AMI 경우 인스턴스의 ID를 선택합니다. 사용 가능한 AMIs 모든 항목을 AMI 검색하거나 최근 또는 빠른 시작 목록에서 하나를 선택할 수 있습니다. 필요한 항목이 보이지 않는 경우 더 찾아보기를 선택하여 전체 AMIs AMI 카탈로그를 찾아보십시오. AMI
6. 인스턴스 유형에서 지정한 것과 호환되는 인스턴스의 하드웨어 구성을 선택합니다. AMI
7. (옵션) 키 페어 이름(로그인)에서 인스턴스와 연결할 때 사용할 키 페어를 선택합니다.
8. 네트워크 설정(Network settings)에서 고급 네트워크 구성(Advanced network configuration)을 스케일 아웃하고 다음을 수행하십시오:

    a. 네트워크 인터페이스 추가(Add network interface)를 선택하여 기본 네트워크 인터페이스를 구성합니다.

b. 퍼블릭 IP 자동 할당의 경우 인스턴스가 퍼블릭 IPv4 주소를 수신할지 여부를 지정합니다. 기본적으로 EC2 Amazon은 기본 서브넷에서 EC2 인스턴스를 시작하거나 퍼블릭 IPv4 주소를 자동으로 할당하도록 구성된 서브넷에서 인스턴스를 시작하는 경우 퍼블릭 주소를 할당합니다. IPv4 인스턴스에 연결할 필요가 없는 경우 [Disable] 을 선택하여 그룹의 인스턴스가 인터넷에서 직접 트래픽을 수신하지 못하게 할 수 있습니다. 이 경우, 로드 밸런서에서만 트래픽을 수신합니다.

c. 보안 그룹 ID의 경우 로드 밸런서와 VPC 동일한 인스턴스의 보안 그룹을 지정합니다.

d. 해지 시 삭제에 대해 예를 선택합니다. 그러면 Auto Scaling 그룹 축소 시 네트워크 인터페이스가 삭제되고 네트워크 인터페이스가 연결되는 인스턴스가 해지됩니다.

9. (선택 사항) 인스턴스에 자격 증명을 안전하게 배포하려면 고급 세부 정보, IAM인스턴스 프로필에 해당 IAM 역할의 Amazon 리소스 이름 (ARN) 을 입력합니다.
10. (옵션) 인스턴스에 대한 사용자 데이터 또는 구성 스크립트를 지정하려면 [Advanced details], [User data]에 붙여넣습니다.
11. Create launch template(시작 템플릿 생성)을 선택합니다.
12. 확인 페이지에서 Auto Scaling 그룹 생성을 선택합니다.

시작 구성 생성 또는 선택

> **Note**
>
> 시작 구성은 계획된 투자가 필요 없는 레거시 기능이므로 새 애플리케이션에서의 시작 구성은 사용하지 않는 것이 좋습니다. 또한 2023년 6월 1일 또는 그 이후에 생성된 새 계정에는 콘솔을 통해 새 시작 구성을 생성할 수 있는 옵션이 없습니다. 자세한 내용은 Auto Scaling 시작 구성 단원을 참조하십시오.

기존 시작 구성을 선택하려면

1. Amazon EC2 콘솔의 시작 구성 페이지를 엽니다.
2. 상단의 탐색 모음에서 로드 밸런서가 생성된 리전을 선택합니다.
3. 시작 구성을 선택합니다.
4. Actions(작업), Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

또는 다음 절차를 사용하여 새 시작 구성을 생성합니다.

시작 구성을 생성하려면

1. Amazon EC2 콘솔의 시작 구성 페이지를 엽니다. 확인 메시지가 표시되면 시작 구성 보기를 선택하여 시작 구성 페이지를 볼 것인지 확인합니다.
2. 상단의 탐색 모음에서 로드 밸런서가 생성된 리전을 선택합니다.
3. Create launch configuration(시작 구성 생성)을 선택하고 시작 구성의 이름을 입력합니다.
4. Amazon 머신 이미지 (AMI) 의 AMI 경우 인스턴스의 ID를 검색 기준으로 입력합니다.
5. Instance type(인스턴스 유형)으로 인스턴스의 하드웨어 구성을 선택합니다.
6. Additional configuration(추가 구성)에서 다음 필드에 주의하세요.

   a. (선택 사항) 자격 증명 (EC2예: 프로필) 을 IAM 인스턴스에 안전하게 배포하려면 IAM 역할을 선택합니다. 자세한 내용은 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 단원을 참조하십시오.

   b. (옵션) 인스턴스에 대한 사용자 데이터 또는 구성 스크립트를 지정하려면 Advanced details(고급 세부 정보), User data(사용자 데이터터)에 붙여 넣습니다.

   c. (옵션) Advanced details(고급 세부 정보), IP address type(IP 주소 유형)에 대해서는 기본값을 유지합니다. Auto Scaling 그룹을 생성할 때, 기본 서브넷과 같이 퍼블릭 IP 주소 지정 속성이 활성화된 서브넷을 사용하여 Auto Scaling 그룹의 인스턴스에 퍼블릭 IP 주소를 할당할 수 있습니다. VPC 또는 인스턴스에 연결할 필요가 없는 경우, 그룹의 인스턴스가 인터넷에서 직접 트래픽을 받지 못하도록 어느 인스턴스에도 퍼블릭 IP 주소를 할당하지 않습니다.를 선택할 수 있습니다. 이 경우, 로드 밸런서에서만 트래픽을 수신합니다.

7. 보안 그룹의 경우 로드 밸런서와 VPC 동일한 보안 그룹 중에서 기존 보안 그룹을 선택합니다. 새 보안 그룹 생성 옵션을 선택한 상태로 유지하면 Linux를 실행하는 Amazon EC2 인스턴스에 대한 기본 SSH 규칙이 구성됩니다. 기본 RDP 규칙은 Windows를 실행하는 Amazon EC2 인스턴스에 대해 구성되어 있습니다.
8. Key pair (login)(키 페어(로그인))로 Key pair options(키 페어 옵션) 아래에 있는 옵션을 선택합니다.

   Amazon EC2 인스턴스 키 쌍을 이미 구성한 경우 여기에서 선택할 수 있습니다.

   Amazon EC2 인스턴스 키 페어가 아직 없는 경우 [Create a new key pair] 를 선택하고 알아볼 수 있는 이름을 지정하십시오. Download key pair(키 페어 다운로드)를 선택하여 컴퓨터에 키 페어를 다운로드합니다.

> **Important**
>
> 인스턴스에 연결해야 하는 경우, Proceed without a key pair(키 페어 없이 계속)를 선택하지 마세요.

9. 승인 확인란을 선택한 다음 Create launch configuration(시작 구성 생성)을 선택합니다.
10. 새 시작 구성 이름 옆의 확인란을 선택하고 Actions(작업),Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

2단계: Auto Scaling 그룹 생성

시작 템플릿 또는 시작 구성을 선택 또는 생성한 후 중단한 시점부터 계속하려면 다음 절차를 따르세요.

Auto Scaling 그룹 생성

1. 시작 템플릿 또는 구성 선택 페이지에서 Auto Scaling 그룹 명칭에 Auto Scaling 그룹의 이름을 입력합니다.
2. [시작 템플릿만 해당] Launch template(시작 템플릿)에서 Auto Scaling 그룹이 스케일 아웃 시 시작 템플릿의 기본 버전을 사용할지, 최신 버전을 사용할지, 아니면 특정 버전을 사용할지 선택합니다.
3. Next(다음)를 선택합니다.

   Auto Scaling 그룹에서 사용할 VPC 네트워크 설정을 선택하고 온디맨드 및 스팟 인스턴스 (시작 템플릿을 선택한 경우) 를 시작하기 위한 옵션을 제공하는 인스턴스 시작 옵션 선택 페이지가 나타납니다.
4. Network 섹션에서 로드 밸런서에 VPC 사용한 항목을 선택합니다. VPC VPC기본값을 선택하면 인스턴스에 인터넷 연결을 제공하도록 자동으로 구성됩니다. VPC여기에는 지역 내 각 가용 영역의 퍼블릭 서브넷이 포함됩니다.
5. 가용 영역 및 서브넷(Availability Zones and subnets)의 경우, 로드 밸런서가 있는 가용 영역에 따라 포함할 각 가용 영역의 서브넷을 하나 이상 선택합니다. 자세한 설명은 VPC 서브넷 선택 시 고려 사항 섹션을 참조하세요.
6. [시작 템플릿만 해당] Instance type requirements(인스턴스 유형 요건) 섹션에서 기본 설정을 사용하여 이 단계를 단순화합니다. (시작 템플릿을 재정의하지 마세요.) 이 자습서에서는 시작 템플릿에 지정된 인스턴스 유형을 사용하여 온디맨드 인스턴스만 시작합니다.
7. 다음(Next)을 선택하고 고급 옵션 구성(Configure advanced options) 페이지로 이동합니다.

8. 그룹을 기존 로드 밸런서에 연결하려면 로드 밸런싱(Load balancing) 섹션에서 기존 로드 밸런서에 연결(Attach to an existing load balancer)을 선택합니다. 로드 밸런서 대상 그룹에서 선택(Choose from your load balancer target groups) 또는 Classic Load Balancer에서 선택(Choose from Classic Load Balancers)을 선택할 수 있습니다. 그런 다음 생성한 Application Load Balancer 또는 Network Load Balancer의 대상 그룹 명칭을 선택하거나 Classic Load Balancer의 이름을 선택할 수 있습니다.
9. (옵션) 건전성 체크, 추가 건전성 체크 유형의 경우, Elastic Load Balancing 건전성 체크 켜기를 선택하십시오.
10. (옵션) 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. 이 시간은 Amazon EC2 Auto Scaling 이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.
11. Auto Scaling 그룹 구성을 완료했으면 Skip to review(검토로 이동)를 선택합니다.
12. Review(검토) 페이지에서 Auto Scaling 그룹의 세부 정보를 검토합니다. [Edit]를 선택하여 변경할 수 있습니다. 모두 마쳤으면 Auto Scaling 그룹 생성을 선택합니다.

로드 밸런서가 연결된 Auto Scaling 그룹을 생성한 후, 로드 밸런서는 온라인 상태가 될 때 새 인스턴스를 자동으로 등록합니다. 이 시점에서는 인스턴스가 하나만 있으므로 등록할 사항이 많이 않습니다. 그러나 원하는 그룹의 용량을 업데이트하여 인스턴스를 더 추가할 수 있습니다. step-by-step 지침은 을 참조하십시오귀하의 Auto Scaling 그룹의 원하는 용량 변경.

3단계: 로드 밸런서가 연결됐는지 확인

로드 밸런서가 연결됐는지 확인하려면

1. Amazon EC2 콘솔의 Auto Scaling 그룹 페이지에서 Auto Scaling 그룹 옆의 확인란을 선택합니다.
2. Details(세부 정보) 탭에서 로드 밸런싱은 연결된 모든 로드 밸런서 대상 그룹 또는 Classic Load Balancer를 표시합니다.
3. Activity(활동)탭의 Activity history(활동 기록)에서 인스턴스가 성공적으로 시작되었는지 확인할 수 있습니다. Status(상태) 열에는 Auto Scaling 그룹에서 인스턴스를 시작했는지가 표시됩니다. 인스턴스가 시작되지 않으면 Amazon EC2 Auto Scaling의 문제 해결에서 일반적인 인스턴스 시작 문제에 대한 문제 해결 아이디어를 찾을 수 있습니다.
4. Instance management(인스턴스 관리) 탭의 Instance (인스턴스)에서 인스턴스가 트래픽을 수신할 준비가 되었는지 확인할 수 있습니다. 처음에는 인스턴스가 `Pending` 상태로 되어 있습니다. 인스턴스가 트래픽을 수신할 준비가 되면 상태가 `InService`로 전환됩니다. 상태 열에는 인스턴스에 대한 Amazon EC2 Auto Scaling 상태 확인 결과가 표시됩니다. 인스턴스가 건전 상태로 표시되더라도 로드 밸런서는 로드 밸런서 건전성 체크를 통과하는 인스턴스에만 트래픽을 전송합니다.

5. 인스턴스가 로드 밸런서에 등록되어 있는지 확인합니다. Amazon EC2 콘솔의 대상 그룹 페이지를 엽니다. 대상 그룹을 선택한 다음 대상 탭을 선택합니다. 인스턴스의 상태가 `initial`이면 인스턴스가 아직 등록 중이거나 건전성 체크 중이기 때문일 수 있습니다. 인스턴스의 상태가 `healthy`이면 사용할 준비가 된 것입니다.

4단계: 다음 단계

이 자습서를 완료했으므로 다음 내용을 자세히 알아볼 수 있습니다.

- Amazon EC2 Auto Scaling은 Auto Scaling 그룹에서 사용하는 상태 확인의 상태를 기반으로 인스턴스가 정상인지 여부를 결정합니다. 로드 밸런서 상태 확인을 활성화한 후 인스턴스가 상태 확인에 실패하면 Auto Scaling 그룹은 해당 인스턴스를 비정상으로 간주하여 교체합니다. 자세한 내용은 상태 확인 단원을 참조하십시오.
- 동일한 지역의 추가 가용 영역으로 애플리케이션을 스케일 아웃하여 서비스 중단 시 내결함성을 높일 수 있습니다. 자세한 내용은 ??? 단원을 참조하십시오.
- 대상 추적 조정 정책을 사용하도록 Auto Scaling 그룹을 구성할 수 있습니다. 이렇게 하면 인스턴스 수요가 변경될 때 인스턴스 수가 자동으로 늘어나거나 줄어듭니다. 이렇게 하면 그룹이 애플리케이션이 수신하는 트래픽 양의 변경을 처리할 수 있습니다. 자세한 설명은 대상 추적 조정 정책 섹션을 참조하세요.

5단계: 정리

이 자습서에 사용하기 위해 생성한 리소스를 다 사용한 후에는 불필요한 비용이 발생하지 않도록 리소스를 정리하는 것이 좋습니다.

Auto Scaling 그룹을 삭제하려면

1. Amazon EC2 콘솔의 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.
3. Delete(삭제)를 선택합니다.
4. 확인 메시지가 표시되면 **`delete`**를 입력하여 지정된 Auto Scaling 그룹 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

   Name(이름) 열의 로딩 아이콘은 Auto Scaling 그룹이 삭제 중임을 나타납니다. 삭제가 수행되면 Desired(필요), Min(최소), Max(최대) 열에 Auto Scaling 그룹에 대해 `0`개의 인스턴스가 표시됩니

다. 인스턴스를 해지하고 그룹을 삭제하는 데 몇 분 정도 걸립니다. 목록을 새로 고침하여 상태를 확인합니다.

현재 시작 템플릿을 유지하려면 아래 절차를 건너뜁니다.

시작 템플릿을 삭제하려면

1. Amazon EC2 콘솔의 시작 템플릿 페이지를 엽니다.
2. 시작 템플릿을 선택합니다.
3. Actions(작업)와 Delete template(템플릿 삭제)을 차례로 선택합니다.
4. 확인 메시지가 표시되면 **`Delete`**를 입력하여 지정된 시작 템플릿 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

현재 시작 구성을 유지하려면 아래 절차를 건너뜁니다.

시작 구성을 삭제하려면

1. Amazon EC2 콘솔의 시작 구성 페이지를 엽니다.
2. 시작 구성을 선택합니다.
3. Actions(작업), Delete launch configuration(시작 구성 삭제)을 선택합니다.
4. 확인 메시지가 나타나면 Delete(삭제)를 선택합니다.

로드 밸런서를 나중에도 계속 사용하고 싶은 경우에는 다음 절차를 건너뜁니다.

로드 밸런서를 삭제하려면

1. Amazon EC2 콘솔의 로드 밸런서 페이지를 엽니다.
2. 로드 밸런서를 선택한 다음 작업, 삭제를 선택합니다.
3. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.

대상 그룹을 삭제하려면

1. Amazon EC2 콘솔의 대상 그룹 페이지를 엽니다.
2. 대상 그룹을 선택하고 [Actions], [Delete]를 차례로 선택합니다.
3. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.

관련 리소스

를 사용하면 템플릿 파일을 사용하여 리소스 컬렉션을 단일 유닛 (스택) 으로 생성 및 삭제함으로써 예측 가능하고 반복적으로 AWS 인프라 배포를 생성하고 프로비저닝할 수 있습니다. AWS CloudFormation 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하십시오.

Auto Scaling 그룹과 Application Load Balancer를 프로비저닝하기 위해 스택 템플릿을 사용하는 방법은 AWS CloudFormation 사용자 가이드의 연습: 스케일링 및 로드 밸런싱 적용 생성을 참조하세요. 이러한 연습과 샘플 템플릿을 시작점으로 사용하여 귀하의 니즈에 맞는 유사한 템플릿을 생성하십시오.

Auto Scaling 시작 템플릿

시작 템플릿은 인스턴스 구성 정보를 지정한다는 점에서 시작 구성과 동일합니다. 여기에는 Amazon Machine Image의 ID(AMI), 인스턴스 유형, 키 페어, 보안 그룹 및 EC2 인스턴스를 시작하는 데 사용되는 기타 파라미터가 포함됩니다. 하지만, 시작 구성 대신에 시작 템플릿을 정의하면 여러 시작 템플릿 버전을 사용할 수 있습니다.

시작 템플릿 버전 관리를 사용하면 전체 파라미터 세트의 하위 세트를 생성할 수 있습니다. 그런 다음, 해당 세트를 재사용하여 동일한 시작 템플릿의 다른 버전을 만들 수 있습니다. 예를 들어 AMI 또는 사용자 데이터 스크립트 없이 기본 구성을 정의하는 시작 템플릿을 생성할 수 있습니다. 시작 템플릿을 생성한 후 새 버전을 생성하고 테스트를 위해 애플리케이션의 최신 버전이 있는 AMI 및 사용자 데이터를 추가할 수 있습니다. 이렇게 하면 두 가지 버전의 시작 템플릿이 생성됩니다. 기본 구성을 저장하면 필요한 일반 구성 파라미터를 유지 관리하는 데 도움이 됩니다. 필요할 때마다 기본 구성에서 시작 템플릿의 새 버전을 생성할 수 있습니다. 애플리케이션을 테스트하는 데 사용된 버전은 더 이상 필요하지 않으면 삭제할 수도 있습니다.

최신 기능과 향상된 기능에 액세스하려면 시작 템플릿을 사용하는 것이 좋습니다. 시작 구성을 사용할 때 일부 Amazon EC2 Auto Scaling 기능을 사용할 수 없습니다. 예를 들어, 스팟 및 온디맨드 인스턴스를 모두 시작하거나 여러 인스턴스 유형을 지정하는 Auto Scaling 그룹은 생성할 수 없습니다. 이러한 기능을 구성하려면 시작 템플릿을 사용해야 합니다. 자세한 내용은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 단원을 참조하십시오.

시작 템플릿을 사용하면 Amazon 의 최신 기능을 사용할 수도 있습니다EC2. 여기에는 Systems Manager 파라미터(AMI ID), EBS 프로비저닝된 IOPS 볼륨의 현재 생성(io2), EBS 볼륨 태그 지정, T2 무제한 인스턴스, 용량 예약,Capacity Blocks, 및 전용 호스트를 예로 들어 보겠습니다.

시작 템플릿을 생성할 때 모든 파라미터는 선택 사항입니다. 그러나 시작 템플릿이 를 지정하지 않으면 Auto Scaling 그룹을 생성할 AMI 때 를 추가할 수 AMI없습니다. 인스턴스 유형 AMI 없이 를 지정하는 경우 Auto Scaling 그룹을 생성할 때 하나 이상의 인스턴스 유형을 추가할 수 있습니다.

내용

- 시작 템플릿으로 작업할 수 있는 권한
- API 시작 템플릿에서 지원하는 작업
- Auto Scaling 그룹에 대한 시작 템플릿 생성
- 고급 설정을 사용하여 시작 템플릿 생성
- Auto Scaling 그룹을 마이그레이션하여 템플릿 시작
- 스택을 시작 템플릿으로 마이그레이션 AWS CloudFormation

- 를 사용하여 시작 템플릿을 생성하고 관리하는 예제 AWS CLI
- AMI IDs 시작 템플릿 대신 AWS Systems Manager 파라미터 사용

시작 템플릿으로 작업할 수 있는 권한

이 섹션의 절차에서는 사용자가 시작 템플릿을 생성하는 데 필요한 권한이 이미 있다고 가정합니다. 관리자가 권한을 부여하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 IAM 권한을 가진 시작 템플릿에 대한 액세스 제어를 참조하세요.

시작 템플릿에서 지정된 리소스를 사용하고 생성할 충분한 권한이 없는 경우 Auto Scaling 그룹에 대해 시작 템플릿을 지정하려고 시도할 때 시작 템플릿을 사용할 권한이 없다는 오류가 발생합니다. 자세한 내용은 Amazon EC2 Auto Scaling: 시작 템플릿 문제 해결 단원을 참조하십시오.

시작 템플릿을 사용하여 `CreateAutoScalingGroup`, `UpdateAutoScalingGroup`및 `RunInstances` API 작업을 호출할 수 있는 IAM 정책의 예는 섹션을 참조하세요Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다..

API 시작 템플릿에서 지원하는 작업

시작 템플릿에서 지원하는 API 작업 목록은 Amazon 참조 의 Amazon EC2 작업을 참조하세요. EC2 API

Auto Scaling 그룹에 대한 시작 템플릿 생성

시작 템플릿을 사용하여 Auto Scaling 그룹을 생성하려면 먼저 Amazon Machine Image()의 ID를 포함하여 인스턴스를 시작하는 구성 정보가 포함된 시작 템플릿을 생성해야 합니다AMI.

새 시작 템플릿을 생성하려면 다음 절차를 사용합니다.

내용

- 시작 템플릿 생성(콘솔)
- 기본 네트워크 인터페이스 설정 변경(콘솔)
- 스토리지 구성 수정(콘솔)
- 기존 인스턴스에서 시작 템플릿 생성(콘솔)
- 관련 리소스
- 제한 사항

> **Important**
>
> 시작 템플릿을 생성할 때 시작 템플릿 파라미터가 완전히 확인되지 않습니다. 파라미터에 잘못된 값을 지정하거나 지원되는 파라미터 조합을 사용하지 않으면 이 시작 템플릿을 사용하여 인스턴스를 시작할 수 없습니다. 해당 파라미터에 대해 올바른 값을 지정하고 지원되는 파라미터 조합을 사용하는지 확인합니다. 예를 들어 Arm 기반 AWS Graviton 또는 Graviton2 를 사용하여 인스턴스를 시작하려면 Arm 호환 인스턴스 유형을 지정AMI해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 시작 템플릿 제한을 참조하세요.

시작 템플릿 생성(콘솔)

다음 단계에서는 기본 시작 템플릿을 구성하는 방법을 설명합니다.

- 인스턴스를 시작할 Amazon Machine Image(AMI)를 지정합니다.
- AMI 지정한 와 호환되는 인스턴스 유형을 선택합니다.
- 예를 들어 를 사용하여 인스턴스에 연결할 때 사용할 키 페어를 지정합니다SSH.
- 인스턴스에 대한 네트워크 액세스를 허용하려면 하나 이상의 보안 그룹을 추가합니다.
- 각 인스턴스에 연결할 추가 볼륨을 지정합니다.
- 인스턴스 및 볼륨에 사용자 지정 태그(키-값 페어)를 추가합니다.

시작 템플릿 생성

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창의 Instances(인스턴스)에서 Launch Templates(시작 템플릿)을 선택합니다.
3. Create launch template(시작 템플릿 생성)을 선택합니다. 시작 템플릿의 이름을 입력하고 초기 버전에 대한 설명을 제공하세요.
4. (선택 사항) Auto Scaling 지침에서 확인란을 선택하여 Amazon EC2 Auto Scaling 에 사용할 템플릿을 생성하는 데 도움이 되는 지침을 Amazon에 EC2 제공하도록 합니다. Auto Scaling
5. Launch template contents(시작 템플릿 내용)에서 필요에 따라 각 필수 필드와 선택적 필드를 작성합니다.

    a. 애플리케이션 및 OS 이미지(Amazon Machine Image): (필수) 인스턴스에 AMI 대한 의 ID를 선택합니다. 사용 가능한 모든 AMI를 검색AMIs하거나 최근 또는 빠른 시작 목록에서 를 선택할 수 있습니다. 필요한 AMI 가 표시되지 않으면 자세히 찾아보AMIs기를 선택하여 전체 AMI 카탈로그를 찾습니다.

사용자 지정 을 선택하려면 먼저 사용자 지정 인스턴스AMI에서 를 생성AMI해야 합니다. 자세한 내용은 Amazon 사용 설명서의 지원 Amazon 생성을 참조EBS하세요AMI. EC2

b. 인스턴스 유형 에서 AMI 지정한 와 호환되는 단일 인스턴스 유형을 선택합니다.

또는 속성 기반 인스턴스 유형 선택을 사용하려면, 고급, 인스턴스 유형 속성 지정을 선택한 다음, 다음 옵션을 지정합니다.

- 수: vCPUs최소 및 최대 수를 입력합니다vCPUs. 제한이 없음을 나타내려면 최소값 0을 입력하고 최대값을 비워 둡니다.
- Amount of memory(MiB)(메모리 용량(MiB)): 최소 및 최대 메모리 양을 MiB 단위로 입력합니다. 제한이 없음을 나타내려면 최소값 0을 입력하고 최대값을 비워 둡니다.
- Optional instance type attributes(인스턴스 유형 속성(선택 사항))를 확장하고 Add attribute(속성 추가)를 선택하여 원하는 용량을 충족하는 데 사용할 수 있는 인스턴스 유형을 추가로 제한합니다. 각 속성에 대한 자세한 내용은 Amazon 참조InstanceRequirementsRequest의 섹션을 참조하세요. EC2 API
- 결과 인스턴스 유형 : , vCPUs메모리 및 스토리지와 같이 지정된 컴퓨팅 요구 사항과 일치하는 인스턴스 유형을 볼 수 있습니다.
- 인스턴스 유형을 제외하려면Add attribute(속성 추가)를 선택합니다. Attribute(속성) 목록에서 Excluded instance types(제외된 인스턴스 유형)을 선택합니다. Attribute value(속성 값) 목록에서 제외할 인스턴스 유형을 선택합니다.

c. Key pair (login)(키 페어(로그인)): Key pair name(키 페어 이름)에서 기존 키 페어를 선택하거나 Create new key pair(새로운 키 페어 생성)를 선택하여 새로 생성합니다. 자세한 내용은 Amazon 사용 설명서의 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요. EC2

d. Network settings(네트워크 설정): Firewall (security groups)(방화벽(보안 그룹))에서 보안 그룹을 하나 이상 사용하거나, 비워 두고 하나 이상의 보안 그룹을 네트워크 인터페이스의 일부로 구성합니다. 자세한 내용은 Amazon 사용 설명서의 Linux 인스턴스용 Amazon EC2 보안 그룹을 참조하세요. EC2

시작 템플릿에 보안 그룹을 지정하지 않으면 Amazon은 Auto Scaling 그룹이 인스턴스를 시작할 VPC 에 대한 기본 보안 그룹을 EC2 사용합니다. 기본적으로 이 보안 그룹은 외부 네트워크로부터의 인바운드 트래픽을 허용하지 않습니다. 자세한 내용은 Amazon VPC 사용 설명서의 에 대한 기본 보안 그룹을 VPCs 참조하세요.

e. 다음 중 하나를 수행합니다.

- 기본 네트워크 인터페이스 설정을 변경합니다. 예를 들어 서브넷의 퍼블릭 IPv4 주소 자동 할당 설정을 재정의하는 퍼블릭 IPv4 주소 지정 기능을 활성화하거나 비활성화할 수 있습니다. 자세한 내용은 기본 네트워크 인터페이스 설정 변경(콘솔) 단원을 참조하십시오.
- 기본 네트워크 인터페이스 설정을 유지하려면 이 단계를 건너뜁니다.

f. 다음 중 하나를 수행합니다.

- 스토리지 구성을 수정합니다. 자세한 내용은 스토리지 구성 수정(콘솔)(을)를 참조하세요.
- 기본 스토리지 구성을 유지하려면 이 단계를 건너뜁니다.

g. 리소스 태그(Resource tags)에서 키와 값의 조합을 제공하여 태그를 지정합니다. 시작 템플릿에 인스턴스 태그를 지정하고 Auto Scaling 그룹의 태그를 해당 인스턴스로 전파하도록 선택한 경우 모든 태그가 병합됩니다. 시작 템플릿의 태그와 Auto Scaling 그룹의 태그에 대해 동일한 태그 키가 지정된 경우 그룹의 태그 값이 우선합니다.

6. (선택 사항) 고급 설정을 구성합니다. 예를 들어 애플리케이션이 다른 AWS 리소스에 액세스할 때 사용할 수 있는 IAM 역할을 선택하거나 인스턴스 시작 후 일반적인 자동 구성 작업을 수행하는 데 사용할 수 있는 인스턴스 사용자 데이터를 지정할 수 있습니다. 자세한 내용은 고급 설정을 사용하여 시작 템플릿 생성 단원을 참조하십시오.
7. 시작 템플릿을 생성할 준비가 되었으면 Create launch template(시작 템플릿 생성)을 선택합니다.
8. Auto Scaling 그룹을 생성하려면 확인 페이지에서 Create an Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

기본 네트워크 인터페이스 설정 변경(콘솔)

네트워크 인터페이스는 VPC 및 인터넷의 다른 리소스에 대한 연결을 제공합니다. 자세한 내용은 Amazon VPC를 사용하여 Auto Scaling 인스턴스에 네트워크 연결 제공 단원을 참조하십시오.

이 섹션에서는 기본 네트워크 인터페이스 설정을 변경하는 방법을 보여줍니다. 예를 들어 서브넷의 퍼블릭 IPv4 주소 자동 할당 설정을 기본값으로 설정하는 대신 각 인스턴스에 퍼블릭 IPv4 주소를 할당할지 여부를 정의할 수 있습니다.

고려 사항 및 제한

기본 네트워크 인터페이스 설정을 변경할 때는 다음 고려 사항 및 제한에 유의하세요.

- 템플릿의 Security groups(보안 그룹) 섹션이 아니라 네트워크 인터페이스의 일부로 보안 그룹을 구성해야 합니다. 두 곳 모두에서 보안 그룹을 지정할 수 없습니다.
- 보조 IP 주소라고 하는 보조 프라이빗 IP 주소를 네트워크 인터페이스에 할당할 수 없습니다.

- 기존 네트워크 인터페이스 ID를 지정하는 경우 하나의 인스턴스만 시작할 수 있습니다. 이렇게 하려면 AWS CLI 또는 를 사용하여 Auto Scaling 그룹을 SDK 생성해야 합니다. 그룹을 생성할 때 서브넷 ID가 아니라 가용 영역을 지정해야 합니다. 디바이스 인덱스가 0인 경우에만 기존 네트워크 인터페이스를 지정할 수 있습니다.
- 둘 이상의 네트워크 인터페이스를 지정하는 경우 퍼블릭 IPv4 주소를 자동 할당할 수 없습니다. 또한 네트워크 인터페이스에서 중복 디바이스 인덱스를 지정할 수 없습니다. 주 네트워크 인터페이스와 보조 네트워크 인터페이스는 모두 동일한 서브넷에 있습니다.
- 인스턴스가 시작되면 프라이빗 주소가 각 네트워크 인터페이스에 자동으로 할당됩니다. 주소는 인스턴스가 시작되는 서브넷의 CIDR 범위에서 가져옵니다. VPC 또는 서브넷의 CIDR 블록(또는 IP 주소 범위)을 지정하는 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서 를 참조하세요.

기본 네트워크 인터페이스 설정을 변경하려면

1. Network settings(네트워크 설정)에서 Advanced network configuration(고급 네트워크 구성)을 확장합니다.
2. Add network interface(네트워크 인터페이스 추가)를 선택하고 다음 필드에 주의를 기울여 기본 네트워크 인터페이스를 구성합니다.

   a. Device index(디바이스 인덱스): 기본 네트워크 인터페이스(eth0)에 변경 내용을 적용하려면 기본값인 0을 유지합니다.

   b. 네트워크 인터페이스 : 인스턴스가 시작될 때 Amazon EC2 Auto Scaling이 새 네트워크 인터페이스를 자동으로 생성하도록 기본값인 새 인터페이스 를 유지합니다. 또한 디바이스 인덱스가 0인 사용 가능한 기존 네트워크 인터페이스를 선택할 수 있지만, 이렇게 하면 Auto Scaling 그룹이 하나의 인스턴스로 제한됩니다.

   c. Description(설명): (선택 사항) 설명적인 이름을 입력합니다.

   d. Subnet(서브넷): 기본값인 Don't include in launch template(시작 템플릿에 포함하지 않음) 설정을 유지합니다.

      이 네트워크 인터페이스에 서브넷을 AMI 지정하면 오류가 발생합니다. 해결 방법으로 Auto Scaling guidance(Auto Scaling 지침)을 끄는 것이 좋습니다. 변경한 후에는 오류 메시지를 수신하지 않습니다. 그러나, 서브넷이 지정된 위치에 관계없이 Auto Scaling 그룹의 서브넷 설정이 우선하며 재정의할 수 없습니다.

   e. 퍼블릭 IP 자동 할당 : 디바이스 인덱스가 0인 네트워크 인터페이스가 퍼블릭 IPv4 주소를 수신하는지 여부를 변경합니다. 기본적으로 기본 서브넷의 인스턴스는 퍼블릭 IPv4 주소를 수신하지만 기본이 아닌 서브넷의 인스턴스는 퍼블릭 주소를 수신하지 않습니다. 활성화 또는 비활성화를 선택하여 서브넷의 기본 설정을 재정의합니다.

f. Security groups(보안 그룹): 네트워크 인터페이스에 보안 그룹을 한 개 이상 선택합니다. Auto Scaling 그룹이 인스턴스를 시작할 VPC 에 대해 각 보안 그룹을 구성해야 합니다. 자세한 내용은 Amazon 사용 설명서의 Linux 인스턴스용 Amazon EC2 보안 그룹을 참조하세요. EC2

g. Delete on termination(종료 시 삭제): Yes(예)를 선택하여 인스턴스 종료 시 네트워크 인터페이스를 삭제하거나 No(아니요)를 선택해 네트워크 인터페이스를 유지합니다.

h. Elastic Fabric Adapter: 고성능 컴퓨팅 및 기계 학습 사용 사례를 지원하려면 네트워크 인터페이스를 Elastic Fabric Adapter 네트워크 인터페이스로 변경합니다. 자세한 내용은 Amazon EC2 사용 설명서의 Elastic Fabric Adapter를 참조하세요.

i. 네트워크 카드 인덱스(Network card index): 0을 선택하여 기본 네트워크 인터페이스를 디바이스 인덱스가 0인 네트워크 카드에 연결합니다. 이 옵션을 사용할 수 없는 경우 기본값인 Don't include in launch template(시작 템플릿에 포함하지 않음)을 유지합니다. 지원되는 인스턴스 유형인 경우에만 네트워크 인터페이스를 특정 네트워크 카드에 연결할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 네트워크 카드를 참조하세요.

j. ENA Express : ENA Express를 지원하는 인스턴스 유형의 경우 활성화를 선택하여 ENA Express를 활성화하거나 비활성화를 선택하여 비활성화합니다. 자세한 내용은 Amazon EC2 사용 설명서의 ENA Express on Linux 인스턴스를 사용하여 네트워크 성능 개선을 참조하세요.

k. ENA Express UDP: ENA Express 를 활성화하면 UDP 트래픽에 선택적으로 사용할 수 있습니다. 활성화를 선택하여 ENA Express를 활성화UDP하거나 비활성화를 선택하여 비활성화합니다.

3. 보조 네트워크 인터페이스를 추가하려면 Add network interface(네트워크 인터페이스 추가)를 선택합니다.

스토리지 구성 수정(콘솔)

Amazon EBS지원 AMI 또는 인스턴스 스토어 지원 에서 시작된 인스턴스의 스토리지 구성을 수정할 수 있습니다AMI. 인스턴스에 연결할 추가 EBS 볼륨을 지정할 수도 있습니다. 에는 루트 볼륨(볼륨 1(AMI루트))을 포함하여 하나 이상의 스토리지 볼륨이 AMI 포함됩니다.

스토리지 구성을 수정하려면

1. Configure storage(스토리지 구성)에서 볼륨의 크기 또는 유형을 수정합니다.

   볼륨 크기로 지정한 값이 볼륨 유형 제한을 넘거나 스냅샷 크기보다 작은 경우 오류 메시지가 표시됩니다. 문제를 해결하는 데 도움이 되도록 이 메시지는 필드에 허용되는 최소값 또는 최대값을 제공합니다.

Amazon EBS지원 볼륨과 연결된 볼륨만 AMI 표시됩니다. 인스턴스 스토어 지원 에서 시작된 인스턴스의 스토리지 구성에 대한 정보를 표시하려면 인스턴스 스토어 볼륨 섹션에서 세부 정보 표시를 AMI선택합니다.

모든 EBS 볼륨 파라미터를 지정하려면 오른쪽 상단 모서리의 고급 보기로 전환합니다.

2. 고급 옵션에서 수정하려는 볼륨을 확장하고 다음과 같이 볼륨을 구성합니다.

   a. 스토리지 유형 : 인스턴스와 연결할 볼륨(EBS 또는 임시)의 유형입니다. 인스턴스 스토어(임시) 볼륨 유형은 이를 지원하는 인스턴스 유형을 선택한 경우에만 사용할 수 있습니다. 자세한 내용은 Amazon 사용 설명서의 Amazon EBS 볼륨과 Amazon 사용 설명서의 Amazon EC2 인스턴스 스토어를 참조하세요. EBS EC2

   b. Device name(디바이스 이름): 볼륨에서 사용할 디바이스 이름을 목록에서 선택합니다.

   c. Snapshot(스냅샷): 볼륨 생성에 사용할 스냅샷을 선택합니다. Snapshot(스냅샷) 필드에 텍스트를 입력하여 사용 가능한 공유 및 퍼블릭 스냅샷을 검색할 수 있습니다.

   d. 크기(GiB): EBS 볼륨의 경우 스토리지 크기를 지정할 수 있습니다. 프리 티어에 적합한 AMI 및 인스턴스를 선택한 경우 프리 티어를 유지하려면 총 스토리지 30GiB 미만을 유지해야 합니다. 자세한 내용은 Amazon EBS 사용 설명서의 EBS 볼륨 크기 및 구성에 대한 제약 조건을 참조하세요.

   e. 볼륨 유형: EBS 볼륨에서 볼륨 유형을 선택합니다. 자세한 내용은 Amazon 사용 설명서의 Amazon EBS 볼륨 유형을 참조하세요. EBS

   f. IOPS: 프로비저닝(IOPS SSD `io1` 및 `io2`) 또는 범용SSD(gp3) 볼륨 유형을 선택한 경우 볼륨이 지원할 수 있는 초당 I/O 작업 수(IOPS)를 입력할 수 있습니다. io1, io2 및 gp3 볼륨에 필요합니다. gp2, st1, sc1 또는 표준 볼륨에서는 지원되지 않습니다.

   g. 종료 시 삭제 : EBS 볼륨의 경우 인스턴스가 종료될 때 볼륨을 삭제하려면 예를 선택하고 볼륨을 유지하려면 아니요를 선택합니다.

   h. 암호화된 : 인스턴스 유형이 EBS 암호화를 지원하는 경우 예를 선택하여 볼륨에 대한 암호화를 활성화할 수 있습니다. 이 리전에서 기본적으로 암호화를 활성화한 경우, 사용자에 대해 암호화가 활성화됩니다. 자세한 내용은 Amazon 사용 설명서의 Amazon EBS 암호화 및 기본 암호화 활성화를 참조하세요. EBS

   이 파라미터를 설정하는 데 따르는 기본 효과는 아래 표에 설명된 것과 같이 선택한 볼륨 소스에 따라 달라집니다. 모든 경우에 지정된 를 사용할 수 있는 권한이 있어야 합니다 AWS KMS key.

암호화 결과

Encrypted 파라미터 설정	볼륨 소스	기본 암호화 상태	참고
아니요	새(빈) 볼륨	암호화되지 않음*	N/A
	암호화되지 않은 소유 스냅샷	암호화되지 않음*	
	암호화된 소유 스냅샷	동일한 키로 암호화됨	
	암호화되지 않은 공유 스냅샷	암호화되지 않음*	
	암호화된 공유 스냅샷	기본 KMS 키로 암호화됨	
예	새 볼륨	기본 KMS 키로 암호화됨	기본값이 아닌 KMS 키를 사용하려면 KMS 키 파라미터의 값을 지정합니다.
	암호화되지 않은 소유 스냅샷	기본 KMS 키로 암호화됨	
	암호화된 소유 스냅샷	동일한 키로 암호화됨	
	암호화되지 않은 공유 스냅샷	기본 KMS 키로 암호화됨	
	암호화된 공유 스냅샷	기본 KMS 키로 암호화됨	

* 기본적으로 암호화가 활성화된 경우 새로 생성된 모든 볼륨(암호화된 파라미터가 예로 설정되어 있는지 여부)은 기본 KMS 키를 사용하여 암호화됩니다. 암호화된 파라미터와 KMS 키 파라미터를 모두 설정한 경우 기본값이 아닌 KMS 키를 지정할 수 있습니다.

i. KMS 키 : 암호화된 에 예를 선택한 경우 볼륨을 암호화하는 데 사용할 고객 관리형 키를 선택해야 합니다. 이 리전에서 기본적으로 암호화를 사용하도록 설정한 경우 기본 고객 관리형 키가 자동으로 선택됩니다. 다른 키를 선택하거나 를 사용하여 이전에 생성한 고객 관리형 키 ARN의 를 지정할 수 있습니다 AWS Key Management Service.

3. 이 시작 템플릿으로 시작된 인스턴스에 연결할 추가 볼륨을 지정하려면 새 볼륨 추가(Add new volume)를 선택합니다.

기존 인스턴스에서 시작 템플릿 생성(콘솔)

기존 인스턴스에서 시작 템플릿을 생성하려면

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창의 Instances(인스턴스)에서 Instances(인스턴스)를 선택합니다.
3. 인스턴스를 선택하고 Actions(작업), Image and templates(이미지 및 템플릿), Create template from instance(인스턴스에서 템플릿 생성)를 선택합니다.
4. 이름 및 설명을 입력합니다.
5. Auto Scaling guidance(Auto Scaling 지침)에서 확인란을 선택합니다.
6. 필요에 따라 설정을 조정하고 Create launch template(시작 템플릿 생성)을 선택합니다.
7. Auto Scaling 그룹을 생성하려면 확인 페이지에서 Create an Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

관련 리소스

AWS CloudFormation 스택 YAML 템플릿에서 시작 템플릿을 선언하는 방법을 이해하는 데 사용할 수 있는 몇 가지 JSON 및 템플릿 조각을 제공합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 섹션을 사용하여 AWS::EC2::LaunchTemplate 및 시작 템플릿 생성을 AWS CloudFormation 참조하세요.

시작 템플릿에 대한 자세한 내용은 Amazon EC2 사용 설명서의 시작 템플릿에서 인스턴스 시작을 참조하세요.

제한 사항

- 시작 템플릿에서 서브넷을 지정할 수 있지만 시작 템플릿만 사용하여 Auto Scaling 그룹을 생성하는 경우에는 서브넷을 지정할 필요가 없습니다. 시작 템플릿에서 서브넷을 지정하여 Auto Scaling 그룹의 서브넷을 지정할 수는 없습니다. Auto Scaling 그룹의 서브넷은 Auto Scaling 그룹의 자체 리소스 정의에서 가져옵니다.
- 사용자 정의 네트워크 인터페이스에 대한 기타 제한 사항은 기본 네트워크 인터페이스 설정 변경(콘솔)(을)를 참조하세요.

고급 설정을 사용하여 시작 템플릿 생성

이 주제에서는 에서 고급 설정을 사용하여 시작 템플릿을 생성하는 방법을 설명합니다 AWS Management Console.

고급 설정을 사용하여 시작 템플릿을 생성하려면

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창의 인스턴스에서 시작 템플릿 을 선택한 다음 시작 템플릿 생성 을 선택합니다.
3. 다음 주제에 설명된 대로 시작 템플릿을 구성합니다.
   - 필수 설정
   - 고급 설정
4. Create launch template(시작 템플릿 생성)을 선택합니다.

필수 설정

시작 템플릿을 생성할 때는 다음과 같은 필수 설정을 포함해야 합니다.

시작 템플릿 이름

시작 템플릿을 설명하는 고유한 이름을 입력합니다.

애플리케이션 및 OS 이미지(Amazon Machine Image)

사용하려는 Amazon Machine Image(AMI)를 선택합니다. 사용하려는 를 검색하거나 찾을 수 AMI 있습니다. 최적의 크기 조정 효율성을 위해 애플리케이션 코드로 인스턴스를 시작하도록 완전히 구성되어 있고 시작 시 수정이 거의 필요하지 AMI 않은 사용자 지정을 선택합니다.

인스턴스 유형

와 호환되는 인스턴스 유형을 선택합니다AMI. Auto Scaling 그룹의 자체 리소스 정의에 포함된 여러 인스턴스 유형을 사용하려면 시작 템플릿에 인스턴스 유형 추가를 건너뛸 수 있습니다. 인스턴스 유형은 혼합 인스턴스 그룹 를 생성할 계획이 없는 경우에만 필요합니다.

고급 설정

고급 설정은 선택 사항입니다. 고급 설정을 구성하지 않으면 인스턴스에 특정 기능이 추가되지 않습니다.

고급 세부 정보 섹션을 펼쳐 고급 설정을 확인하세요. 다음 섹션에서는 Auto Scaling 그룹에 대한 시작 템플릿을 생성할 때 중점을 두어야 할 가장 유용한 고급 설정을 설명합니다. 자세한 내용은 Amazon EC2 사용 설명서의 고급 세부 정보를 참조하세요.

IAM 인스턴스 프로파일

인스턴스 프로파일에는 사용하려는 IAM 역할이 포함되어 있습니다. Auto Scaling 그룹이 EC2 인스턴스를 시작하면 관련 IAM 역할에 정의된 권한이 인스턴스에서 실행되는 애플리케이션에 부여됩니다. 자세한 내용은 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 단원을 참조하십시오.

종료 방지

이 기능을 활성화하면 사용자가 Amazon EC2 콘솔, CLI 명령 및 API 작업을 사용하여 인스턴스를 종료할 수 없습니다. 종단 보호는 우발적인 종단으로부터 추가적인 보호 기능을 제공합니다. Amazon EC2 Auto Scaling이 인스턴스를 종료하는 것을 방지하지는 않습니다. Amazon EC2 Auto Scaling이 종료할 수 있는 인스턴스를 제어하려면 섹션을 참조하세요인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다..

세부 CloudWatch 모니터링

EC2 인스턴스가 CloudWatch 1분 간격으로 지표 데이터를 Amazon에 전송할 수 있도록 인스턴스에 대한 세부 모니터링을 활성화할 수 있습니다. 기본적으로 EC2 인스턴스는 5분 간격으로 지표 데이터를 CloudWatch 로 전송합니다. 추가 요금이 발생합니다. 자세한 내용은 Auto Scaling 인스턴스에 대한 모니터링 구성 단원을 참조하십시오.

크레딧 사양

AmazonEC2은 T2, T3 및 T3a와 같은 버스트 가능한 성능 인스턴스를 제공하여 필요한 경우 애플리케이션이 기준 CPU 성능을 초과하여 버스트할 수 있도록 합니다. 기본적으로 이러한 인스턴스는 CPU 사용량이 제한되기 전에 제한된 시간 동안 버스트될 수 있습니다. 인스턴스가 필요한 기간 동안 기준을 초과하여 버스트될 수 있도록 무제한 모드를 선택적으로 활성화할 수 있습니다. 이를 통해 애플리케이션은 필요할 CPU 때 고성능을 유지할 수 있습니다. 추가 요금이 적용될 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 Auto Scaling 그룹을 사용하여 버스트 가능한 성능 인스턴스를 무제한으로 시작을 참조하세요.

배치 그룹 이름

배치 그룹을 지정하고 클러스터 또는 파티션 전략을 사용하여 인스턴스가 AWS 데이터 센터에 물리적으로 위치하는 방식에 영향을 미칠 수 있습니다. 소규모 Auto Scaling 그룹의 경우 스프레드 전략을 사용할 수도 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 배치 그룹을 참조하세요.

Auto Scaling 그룹과 함께 배치 그룹을 사용할 때 몇 가지 고려 사항이 있습니다.

- 시작 템플릿과 Auto Scaling 그룹 모두에 배치 그룹이 지정된 경우 Auto Scaling 그룹의 배치 그룹이 우선합니다. 그룹을 생성한 후에는 Auto Scaling 그룹 설정에 지정된 배치 그룹을 변경할 수 없습니다.
- 에서 시작 템플릿에 배치 그룹을 정의하는 경우 AWS CloudFormation주의하십시오. Amazon EC2 Auto Scaling은 지정된 배치 그룹으로 인스턴스를 시작합니다. 그러나 Auto Scaling 그룹과 UpdatePolicy 함께 를 사용하는 경우(향후 변경될 수 있음) 해당 인스턴스에서 신호를 수신 CloudFormation 하지 않습니다.

구매 옵션

스팟 인스턴스 요청을 선택하여 스팟 인스턴스를 온디맨드 가격으로 제한하여 스팟 인스턴스를 요청하고 사용자 지정을 선택하여 기본 스팟 인스턴스 설정을 변경할 수 있습니다. Auto Scaling 그룹의 경우 종료 날짜가 없는 일회성 요청을 지정해야 합니다(기본값). 자세한 내용은 내결함성 및 유연한 애플리케이션을 위한 스팟 인스턴스 요청(을)를 참조하세요. 이 설정은 특수한 상황에서 유용할 수 있지만, 일반적으로 지정하지 않고 대신 혼합 인스턴스 그룹을 생성하는 것이 좋습니다. 자세한 내용은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 단원을 참조하십시오.

시작 템플릿에서 스팟 인스턴스 요청을 지정하는 경우 혼합 인스턴스 그룹을 생성할 수 없습니다. 혼합 인스턴스 그룹이 포함된 스팟 인스턴스를 요청하는 시작 템플릿을 사용하려고 하면 다음과 같은 오류 메시지가 나타납니다 - `Incompatible launch template: You cannot use a launch template that is set to request Spot Instances (InstanceMarketOptions) when you configure an Auto Scaling group with a mixed instances policy. Add a different launch template to the group and try again.`.

Capacity Reservation

용량 예약을 사용하면 특정 가용 영역에 있는 Amazon EC2 인스턴스의 용량을 언제든지 예약할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 온디맨드 용량 예약을 참조하세요.

인스턴스를 다음과 같이 시작할지 여부를 선택할 수 있습니다.

- 열려 있는 용량 예약(열림)
- 특정 용량 예약(ID별 대상)
- 용량 예약 그룹(그룹별 대상)

특정 용량 예약을 대상으로 하려면 시작 템플릿의 인스턴스 유형이 예약의 인스턴스 유형과 일치해야 합니다. Auto Scaling 그룹을 생성할 때 용량 예약과 동일한 가용 영역을 사용합니다. AWS 리전 선택한 에 따라 용량 블록을 대상으로 선택할 수 있습니다. 자세한 내용은 사용 Capacity Blocks 기계 학습 워크로드용 단원을 참조하십시오.

용량 예약 그룹을 대상으로 하려면 섹션을 참조하세요용량 예약을 통해 특정 가용 영역의 용량을 예약할 수 있습니다.. 용량 예약 그룹을 대상으로 하면 여러 가용 영역에 용량을 분산하여 복원력을 개선할 수 있습니다.

Tenancy

AmazonEC2은 EC2 인스턴스 테넌시에 대한 세 가지 옵션을 제공합니다.

- 공유(공유) - 여러 가 동일한 물리적 하드웨어를 공유할 AWS 계정 수 있습니다. 인스턴스를 시작할 때 기본 테넌시 옵션입니다.
- 전용 인스턴스(전용) - 인스턴스는 단일 테넌트 하드웨어에서 실행됩니다. 다른 AWS 고객은 동일한 물리적 서버를 공유하지 않습니다. 자세한 내용은 Amazon EC2 사용 설명서의 전용 인스턴스를 참조하세요.
- 전용 호스트(전용 호스트) - 인스턴스는 사용자 전용 물리적 서버에서 실행됩니다. 전용 호스트를 사용하면 전용 하드웨어 요구 사항이 있고 규정 준수 사용 사례를 EC2 충족하는 자체 라이선스(BYOL)를 더 쉽게 가져올 수 있습니다. 이 옵션을 선택하는 경우 테넌시 호스트 리소스 그룹 에 대한 호스트 리소스 그룹을 제공해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 전용 호스트를 참조하세요.

전용 호스트에 대한 지원은 호스트 리소스 그룹을 지정하는 경우에만 사용할 수 있습니다. 호스트 배치 선호도를 사용하거나 특정 호스트 ID를 대상으로 지정할 수 없습니다.

- 호스트 ID를 지정하는 시작 템플릿을 사용하려고 하면 다음과 같은 오류 메시지가 표시됩니다. `Incompatible launch template: Tenancy host ID is not supported for Auto Scaling.`
- 호스트 배치 선호도를 지정하는 시작 템플릿을 사용하려고 하면 다음과 같은 오류 메시지가 표시됩니다. `Incompatible launch template: Auto Scaling does not support host placement affinity.`

테넌시 호스트 리소스 그룹

를 사용하면 자체 라이선스를 에 가져오 AWS 고 중앙에서 관리할 AWS License Manager수 있습니다. 호스트 리소스 그룹은 특정 License Manager 라이선스 구성에 연결된 전용 호스트 그룹입니다. 호스트 리소스 그룹을 사용하면 소프트웨어 라이선스 요구 사항에 맞는 전용 호스트에서 EC2 인스턴스를 쉽게 시작할 수 있습니다. 전용 호스트를 미리 수동으로 할당할 필요는 없습니다. 필요에 따라 자동으로 생성됩니다. 를 라이선스 구성AMI에 연결할 때 한 번에 하나의 호스트 리소스 그룹에만 연결할 AMI 수 있습니다. 자세한 내용은 License Manager 사용 설명서에서 AWS License Manager의 호스트 리소스 그룹을 참조하세요.

라이선스 구성

이 설정을 사용하면 테넌시를 전용 호스트로 제한하지 않고 인스턴스에 대한 라이선스 구성을 지정할 수 있습니다. 라이선스 구성은 인스턴스에 배포된 소프트웨어 라이선스를 추적하므로 라이선스 사용량 및 규정 준수를 모니터링할 수 있습니다. 자세한 내용은 License Manager 사용 설명서의 자체 관리형 라이선스 생성을 참조하세요.

액세스 가능한 메타데이터

인스턴스 메타데이터 서비스의 HTTP 엔드포인트에 대한 액세스를 활성화 또는 비활성화할지 여부를 선택할 수 있습니다. 기본적으로 HTTP 엔드포인트가 활성화됩니다. 엔드포인트를 비활성화하도록 선택하면 인스턴스 메타데이터에 대한 액세스가 해제됩니다. HTTP 엔드포인트가 활성화된 IMDSv2 경우에만 필요한 조건을 지정할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 옵션 구성을 참조하세요.

메타데이터 버전

인스턴스 메타데이터를 요청할 때 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 사용하도록 선택할 수 있습니다. 값을 지정하지 않으면 기본값은 IMDSv1 및 를 모두 지원하는 것입니다IMDSv2. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 옵션 구성을 참조하세요.

메타데이터 토큰 응답 홉 제한

메타데이터 토큰에 허용되는 네트워크 홉 수를 설정할 수 있습니다. 값을 지정하지 않으면 기본값은 1입니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 옵션 구성을 참조하세요.

사용자 데이터

쉘 스크립트 또는 클라우드 시작 지침을 사용자 데이터로 지정하여 시작 시 인스턴스 구성을 사용자 지정하고 완료할 수 있습니다. 사용자 데이터는 인스턴스가 처음 시작될 때 실행되므로 시작 시 애플리케이션, 종속성 또는 사용자 지정을 자동으로 설치할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 시작 시 Linux 인스턴스에서 명령 실행을 참조하세요.

다운로드 횟수가 크거나 복잡한 스크립트가 있는 경우 인스턴스를 사용할 준비가 되는 데 걸리는 시간이 늘어납니다. 이 경우 인스턴스가 완전히 프로비저닝될 때까지 인스턴스가 `InService` 상태에 도달하는 것을 지연하도록 수명 주기 후크를 구성해야 할 수 있습니다. Auto Scaling 그룹에 수명 주기 후크를 추가하는 방법에 대한 자세한 내용은 섹션을 참조하세요Amazon EC2 Auto Scaling 수명 주기 후크.

내결함성 및 유연한 애플리케이션을 위한 스팟 인스턴스 요청

시작 템플릿에서 필요에 따라 종료 날짜나 기간이 없는 스팟 인스턴스를 요청할 수 있습니다. Amazon EC2 스팟 인스턴스는 EC2 온디맨드 가격과 비교하여 엄청나게 할인된 가격으로 사용할 수 있는 예비 용량입니다. 스팟 인스턴스는 애플리케이션이 실행되는 시간을 유연하게 조정할 수 있고 애플리케이션을 중단할 수 있는 경우에 선택하는 비용 효율적인 방법입니다. 스팟 인스턴스를 요청하는 시작 템플릿을 생성하는 방법에 대한 자세한 정보는 고급 설정을 사용하여 시작 템플릿 생성(을)를 참조하세요.

> **Important**
>
> 스팟 인스턴스는 일반적으로 온디맨드 인스턴스를 보완하는 데 사용됩니다. 이 시나리오에서는 Auto Scaling 그룹 설정의 일부로 스팟 인스턴스를 시작하는 데 사용되는 것과 동일한 설정을 지정할 수 있습니다. Auto Scaling 그룹의 일부로 설정을 지정하면 특정 수의 온디맨드 인스턴스를 시작한 후에만 스팟 인스턴스를 시작하도록 요청한 다음 그룹의 크기 조정에 따라 온디맨드 인스턴스와 스팟 인스턴스의 일부 조합을 계속 시작할 수 있습니다. 자세한 내용은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹(을)를 참조하세요.

이 항목에서는 Auto Scaling 그룹 자체가 아닌 시작 템플릿에서 설정을 지정하여 Auto Scaling 그룹에서 스팟 인스턴스만 시작하는 방법을 설명합니다. 이 주제의 정보는 시작 구성을 사용하여 스팟 인스턴스를 요청하는 Auto Scaling 그룹에도 적용됩니다. 차이점은 시작 구성에는 최고가가 필요하지만 시작 템플릿의 경우 최고가가 선택 사항이라는 것입니다.

스팟 인스턴스만 시작하는 시작 템플릿을 생성하려면 다음 사항을 고려하세요.

- 스팟 요금. 시작하는 스팟 인스턴스에 대해 현재 스팟 요금만 지불합니다. 이 요금은 시간이 지나면서 수요 및 공급의 장기 추세에 따라 서서히 변화합니다. 자세한 내용은 Amazon EC2 사용 설명서의 스팟 인스턴스 및 요금 및 할인을 참조하세요.
- 최고 가격 설정. 필요에 따라 시작 템플릿에 스팟 인스턴스의 시간당 최고가를 포함할 수 있습니다. 최대 가격이 현재 스팟 가격을 초과하는 경우 용량이 사용 가능한 경우 Amazon EC2 스팟 서비스가 즉시 요청을 이행합니다. 스팟 인스턴스의 요금이 Auto Scaling 그룹에서 실행 중인 인스턴스의 최고가보다 오르는 경우 인스턴스가 종료됩니다.

> **Warning**
>
> 스팟 인스턴스를 받지 못하면(예: 최고가가 너무 낮은 경우) 애플리케이션이 실행되지 않을 수 있습니다. 가능한 한 오랫동안 사용 가능한 스팟 인스턴스를 활용하려면 온디맨드 가격에 가깝게 최고가를 설정합니다.

- 여러 가용 영역을 포괄하며 밸런싱. 여러 가용 영역을 지정하는 경우 Amazon EC2 Auto Scaling은 지정된 영역에 스팟 요청을 배포합니다. 한 가용 영역에서 최대 가격이 너무 낮아 요청을 이행할 수 없는 경우 Amazon EC2 Auto Scaling은 요청이 다른 영역에서 이행되었는지 확인합니다. 이 경우 Amazon EC2 Auto Scaling은 실패한 요청을 취소하고 요청이 이행된 가용 영역에 다시 배포합니다. 이행된 요청이 없는 가용 영역의 가격이 향후 요청이 성공할 만큼 충분히 하락하면 Amazon EC2 Auto Scaling은 모든 가용 영역에서 리밸런싱합니다.
- 스팟 인스턴스 종료. 스팟 인스턴스는 언제든지 종료될 수 있습니다. Amazon EC2 스팟 서비스는 스팟 인스턴스의 가용성 또는 가격이 변경되면 Auto Scaling 그룹의 스팟 인스턴스를 종료할 수 있습니다. 상태 확인을 조정하거나 수행할 때 Amazon EC2 Auto Scaling은 온디맨드 인스턴스를 종료하는 것과 동일한 방식으로 스팟 인스턴스를 종료할 수도 있습니다. 인스턴스가 종료되면 모든 스토리지가 삭제됩니다.
- 원하는 용량 유지. 스팟 인스턴스가 종료되면 Amazon EC2 Auto Scaling은 그룹에 원하는 용량을 유지하기 위해 다른 스팟 인스턴스를 시작하려고 시도합니다. 현재 스팟 요금이 최고가보다 낮으면 스팟 인스턴스가 시작됩니다. 스팟 인스턴스에 대한 요청이 성공하지 않을 경우 시도를 계속합니다.
- 최고 가격 변경. 최고가를 변경하려면 새 시작 템플릿을 생성하거나 기존 시작 템플릿을 새로운 최고가로 업데이트한 다음 이를 Auto Scaling 그룹과 연결합니다. 기존 스팟 인스턴스는 해당 인스턴스에 사용된 시작 템플릿에 지정된 최고가가 현재 스팟 가격보다 높은 한 계속 실행됩니다. 최고가를 설정하지 않은 경우 기본 최고가는 온디맨드 가격입니다.

사용 Capacity Blocks 기계 학습 워크로드용

Capacity Blocks 는 단기 기계 학습(ML) 워크로드를 지원하기 위해 향후 날짜에 인기 있는 GPU 인스턴스를 예약하는 데 도움이 됩니다.

의 개요 Capacity Blocks 작동 방식 참조 Capacity Blocks Amazon 사용 설명서의 ML용 . EC2

사용을 시작하려면 Capacity Blocks에서 특정 가용 영역에 용량 예약을 생성합니다.Capacity Blocks 는 단일 가용 영역에서 `targeted` 용량 예약으로 전달됩니다. 시작 템플릿을 생성할 때 용량 블록의 예약 ID와 인스턴스 유형을 지정합니다. 그런 다음 생성한 시작 템플릿과 용량 블록의 가용 영역을 사용하도

록 Auto Scaling 그룹을 업데이트합니다. 용량 블록 예약이 시작되면 예약된 조정을 사용하여 용량 블록 예약과 동일한 수의 인스턴스를 시작합니다.

> **Important**
>
> Capacity Blocks 는 특정 Amazon EC2 인스턴스 유형 및 에만 사용할 수 있습니다 AWS 리전. 자세한 내용은 Amazon EC2 사용 설명서의 사전 요구 사항을 참조하세요.

내용

- 운영 지침
- 시작 템플릿에 용량 블록 지정
- 제한 사항
- 관련 리소스

운영 지침

다음은 Auto Scaling 그룹에서 용량 블록을 사용할 때 따라야 하는 기본 운영 지침입니다.

- 용량 블록 예약 종료 시간 30분 전까지 Auto Scaling 그룹을 0으로 스케일 인합니다. AmazonEC2은 용량 블록 종료 30분 전에 여전히 실행 중인 인스턴스를 종료합니다.
- 예약된 크기 조정을 사용하여 적절한 예약 시간에 스케일 아웃(인스턴스 추가) 및 스케일 인(인스턴스 제거)하는 것이 좋습니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 스케일링 예약 단원을 참조하십시오.
- 스케일 인을 할 경우 인스턴스 내에서 애플리케이션을 정상적으로 종료할 수 있도록 필요에 따라 수명 주기 후크를 추가합니다. Amazon이 용량 블록 예약 종료 시간 30분 전에 인스턴스EC2를 강제로 종료하기 전에 수명 주기 작업을 완료할 수 있도록 충분한 시간을 둡니다. 자세한 내용은 Amazon EC2 Auto Scaling 수명 주기 후크 단원을 참조하십시오.
- Auto Scaling 그룹이 전체 예약 기간 동안 올바른 버전의 시작 템플릿을 가리키는지 확인합니다. `$Default` 또는 `$Latest` 버전 대신 특정 버전의 시작 템플릿을 가리키는 것이 좋습니다.

> **Note**
>
> 예약이 끝날 때까지 용량 블록 인스턴스를 실행 상태로 두고 Amazon이 EC2 이를 회수하는 경우, 용량 블록이 끝날 때 의도적으로 회수되었지만 Auto Scaling 그룹에 대한 스케일링

활동은 “`taken out of service in response to an EC2 health check that indicated it had been terminated or stopped`”로 표시됩니다. 마찬가지로 Amazon EC2 Auto Scaling은 상태 확인에 실패한 모든 인스턴스와 동일한 방식으로 인스턴스를 교체하려고 시도합니다. 자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 단원을 참조하십시오.

시작 템플릿에 용량 블록 지정

Auto Scaling 그룹의 특정 용량 블록을 대상으로 하는 시작 템플릿을 생성하려면 다음 방법 중 하나를 사용합니다.

Console

시작 템플릿에서 용량 블록을 지정하려면(콘솔)

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 상단 탐색 모음에서 용량 블록을 생성한 AWS 리전 를 선택합니다.
3. 탐색 창의 Instances(인스턴스)에서 Launch Templates(시작 템플릿)을 선택합니다.
4. 시작 템플릿 생성을 선택하고 시작 템플릿을 생성합니다. 필요에 따라 Amazon Machine Image의 ID(AMI), 인스턴스 유형 및 기타 시작 템플릿 설정을 포함합니다.
5. 고급 세부 정보 섹션을 펼쳐 고급 설정을 확인하세요.
6. 구매 옵션에서 용량 블록을 선택합니다.
7. 용량 예약의 경우 ID별 대상을 선택한 다음 용량 예약 - ID별 대상에서 기존 용량 블록의 용량 예약 ID를 선택합니다.
8. 모두 마쳤으면 시작 템플릿 생성을 선택합니다.

   시작 템플릿을 사용하여 Auto Scaling 그룹을 생성하는 방법에 대한 자세한 내용은 섹션을 참조하세요시작 템플릿을 사용하여 Auto Scaling 그룹 생성.

AWS CLI

시작 템플릿에 용량 블록을 지정하려면(AWS CLI)

다음 create-launch-template 명령을 사용하여 기존 용량 블록 예약 ID를 지정하는 시작 템플릿을 생성합니다. 각각 바꾸기 *`user input placeholder`* 자신의 정보를 사용합니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-capacity-block
  \
  --version-description AutoScalingVersion1 --region us-east-2 \
  --launch-template-data file://config.json
```

> **Tip**
>
> 이 명령으로 인해 오류가 발생하는 경우 AWS CLI 로컬에서 를 최신 버전으로 업데이트했는지 확인합니다.

`config.json`의 콘텐츠.

```
{
    "ImageId": "ami-04d5cc9b88example",
    "InstanceType": "p4d.24xlarge",
    "SecurityGroupIds": [
        "sg-903004f88example"
    ],
    "KeyName": "MyKeyPair",
    "InstanceMarketOptions": {
        "MarketType": "capacity-block"
    },
    "CapacityReservationSpecification": {
        "CapacityReservationTarget": {
            "CapacityReservationId": "cr-02168da1478b509e0"
        }
    }
}
```

출력의 예시는 다음과 같습니다.

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b724example",
        "LaunchTemplateName": "my-template-for-capacity-block",
        "CreateTime": "2023-10-27T15:12:44.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
```

```
}
```

다음 describe-launch-template-versions 명령을 사용하여 시작 템플릿과 연결된 용량 블록 예약 ID를 확인할 수 있습니다.

```
aws ec2 describe-launch-template-versions --launch-template-names my-template-for-capacity-block \
  --region us-east-2
```

다음은 용량 블록 예약을 지정하는 시작 템플릿에 대한 출력의 예입니다.

```
{
    "LaunchTemplateVersions": [
        {
            "LaunchTemplateId": "lt-068f72b724example",
            "LaunchTemplateName": "my-template-for-capacity-block",
            "VersionNumber": 1,
            "CreateTime": "2023-10-27T15:12:44.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersion": true,
            "LaunchTemplateData": {
                "ImageId": "ami-04d5cc9b88example",
                "InstanceType": "p5.48xlarge",
                "SecurityGroupIds": [
                    "sg-903004f88example"
                ],
                "KeyName": "MyKeyPair",
                "InstanceMarketOptions": {
                    "MarketType": "capacity-block"
                },
                "CapacityReservationSpecification": {
                    "CapacityReservationTarget": {
                        "CapacityReservationId": "cr-02168da1478b509e0"
                    }
                }
            }
        }
    ]
}
```

제한 사항

- 지원 Capacity Blocks 는 Auto Scaling 그룹에 호환 가능한 구성이 있는 경우에만 사용할 수 있습니다. 혼합 인스턴스 그룹과 웜 풀은 지원되지 않습니다.
- 한 번에 하나의 용량 블록만 대상으로 지정할 수 있습니다.

관련 리소스

- P5 인스턴스 사용을 위한 사전 조건 및 권장 사항은 Amazon EC2 사용 설명서의 P5 인스턴스 시작하기를 참조하세요.
- Amazon은 의 사용을 EKS 지원합니다.Capacity Blocks 는 Amazon EKS 클러스터에서 단기간 기계 학습(ML) 워크로드를 지원합니다. 자세한 내용은 단원을 참조하세요.Capacity Blocks Amazon 사용 설명서의 ML용 . EKS
- 다음을 사용할 수 있음: Capacity Blocks 지원되는 인스턴스 유형 및 리전이 있습니다. 그러나 온디맨드 용량 예약은 다른 인스턴스 유형 및 리전에 대한 용량을 예약할 수 있는 유연성을 제공합니다. 온디맨드 용량 예약 옵션을 사용하는 방법을 보여주는 자습서는 섹션을 참조하세요용량 예약을 통해 특정 가용 영역의 용량을 예약할 수 있습니다..

Auto Scaling 그룹을 마이그레이션하여 템플릿 시작

2023년부터 2022년 12월 31일 이후에 릴리스된 `CreateLaunchConfiguration` 새 Amazon EC2 인스턴스 유형을 호출할 수 없습니다. 자세한 내용은 Auto Scaling 시작 구성 단원을 참조하십시오.

Auto Scaling 그룹을 시작 구성에서 시작 템플릿으로 마이그레이션하려면 다음 단계를 참조하세요.

> **Important**
> 계속 진행하기 전에 시작 템플릿을 사용하는 데 필요한 권한이 있는지 확인합니다. 자세한 내용은 시작 템플릿으로 작업할 수 있는 권한 단원을 참조하십시오.

1단계: 시작 구성을 사용하는 Auto Scaling 그룹 찾기

아직 시작 구성을 사용하고 있는 Auto Scaling 그룹이 있는지 확인하려면 를 사용하여 다음 describe-auto-scaling-groups 명령을 실행합니다 AWS CLI. Replace *REGION* 를 사용합니다 AWS 리전.

```
aws autoscaling describe-auto-scaling-groups --region REGION \
```

```
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

출력의 예시는 다음과 같습니다.

```
[
    {
 "AutoScalingGroupName": "group-1",
 "AutoScalingGroupARN": "arn",
 "LaunchConfigurationName": "my-launch-config",
 "MinSize": 1,
 "MaxSize": 5,
 "DesiredCapacity": 2,
 "DefaultCooldown": 300,
 "AvailabilityZones": [
           "us-west-2a",
           "us-west-2b",
           "us-west-2c"
       ],
 "LoadBalancerNames": [],
 "TargetGroupARNs": [],
 "HealthCheckType": "EC2",
 "HealthCheckGracePeriod": 300,
 "Instances": [
           {
               "ProtectedFromScaleIn": false,
               "AvailabilityZone": "us-west-2a",
               "LaunchConfigurationName": "my-launch-config",
               "InstanceId": "i-05b4f7d5be44822a6",
               "InstanceType": "t3.micro",
               "HealthStatus": "Healthy",
               "LifecycleState": "InService"
           },
           {
               "ProtectedFromScaleIn": false,
               "AvailabilityZone": "us-west-2b",
               "LaunchConfigurationName": "my-launch-config",
               "InstanceId": "i-0c20ac468fa3049e8",
               "InstanceType": "t3.micro",
               "HealthStatus": "Healthy",
               "LifecycleState": "InService"
           }
 ],
 "CreatedTime": "2023-03-09T22:15:11.611Z",
```

```
 "SuspendedProcesses": [],
 "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
 "EnabledMetrics": [],
 "Tags": [
            {
  "ResourceId": "group-1",
  "ResourceType": "auto-scaling-group",
  "Key": "environment",
  "Value": "production",
  "PropagateAtLaunch": true
            }
        ],
 "TerminationPolicies": [
     "Default"
 ],
 "NewInstancesProtectedFromScaleIn": false,
 "ServiceLinkedRoleARN": "arn",
       "TrafficSources": []
    },

    ... additional groups ...

]
```

또는 출력에서 해당 시작 구성 및 태그의 이름이 포함된 Auto Scaling 그룹 이름을 제외한 모든 이름을 제거하려면 다음 명령을 실행합니다.

```
aws autoscaling describe-auto-scaling-groups --region REGION \
  --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`].{AutoScalingGroupName:
 AutoScalingGroupName, LaunchConfigurationName: LaunchConfigurationName, Tags: Tags}'
```

다음은 출력의 예입니다.

```
[
    {
        "AutoScalingGroupName": "group-1",
        "LaunchConfigurationName": "my-launch-config",
        "Tags": [
            {
                "ResourceId": "group-1",
                "ResourceType": "auto-scaling-group",
                "Key": "environment",
                "Value": "production",
```

```
                "PropagateAtLaunch": true
            }
        ]
    },

    ... additional groups ...

]
```

필터링에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 출력 필터링을 참조하세요.

2단계: 시작 구성을 시작 템플릿에 복사

다음 절차를 사용하여 시작 구성을 시작 템플릿에 복사할 수 있습니다. 그런 다음, Auto Scaling 그룹에 추가할 수 있습니다.

여러 시작 구성을 복사하면 이름이 동일한 시작 템플릿이 만들어집니다. 복사 프로세스 중에 시작 템플릿에 지정된 이름을 변경하려면 시작 구성을 하나씩 복사해야 합니다.

> **Note**
>
> 복사 기능은 콘솔에서만 사용할 수 있습니다.

시작 구성을 시작 템플릿에 복사하려면(콘솔)

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 왼쪽 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택합니다.
3. 페이지 상단에서 시작 구성을 선택합니다. 확인 메시지가 표시되면 시작 구성 보기를 선택하여 시작 구성 페이지를 볼 것인지 확인합니다.
4. 복사할 시작 구성을 선택하고 Copy to launch template(시작 템플릿으로 복사)을 선택합니다. 이렇게 하면 새 시작 템플릿을 사용자가 선택한 시작 구성과 동일한 이름과 옵션으로 설정합니다.
5. New launch template name(새 시작 템플릿 이름)의 경우, 시작 구성의 이름(기본값)을 사용하거나 새 이름을 입력할 수 있습니다. 시작 템플릿 이름은 고유한 이름이어야 합니다.
6. (선택 사항) 새 템플릿을 사용하여 Auto Scaling 그룹 생성을 선택합니다.

   이 단계를 건너뛰고 시작 구성 복사를 완료할 수 있습니다. 새 Auto Scaling 그룹을 만들 필요가 없습니다.

7. 복사를 선택합니다.

시작 구성을 시작 템플릿에 복사하려면(콘솔)

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창의 Auto Scaling에서 시작 구성을 선택합니다.
3. Copy to launch template, Copy all(시작 템플릿에 복사, 모두 복사)을 선택합니다. 이렇게 하면 현재 리전의 각 시작 구성이 이름과 옵션이 동일한 새 시작 템플릿에 복사됩니다.
4. 복사를 선택합니다.

3단계: 시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트

시작 템플릿을 생성한 후 Auto Scaling 그룹에 추가할 수 있습니다.

시작 템플릿을 사용하도록 Auto Scaling 그룹을 업데이트하려면(콘솔)

1. 에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 https://console.aws.amazon.com/ec2/선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 아래쪽에 분할 창이 열리고 선택한 그룹에 대한 정보가 표시됩니다.
3. Details(세부 정보) 탭에서 Launch configuration(시작 구성), Edit(편집)을 선택합니다.
4. Switch to launch template(시작 템플릿으로 전환)을 선택합니다.
5. Launch template(시작 템플릿)에서 시작 템플릿을 선택합니다.
6. Version(버전)에서 필요한 시작 템플릿 버전을 선택합니다. 한 시작 템플릿을 여러 버전으로 만든 다음에는 Auto Scaling 그룹이 확장 시 시작 템플릿의 기본 버전을 사용할지 최신 버전을 사용할지 선택합니다.
7. 업데이트를 선택합니다.

시작 템플릿을 사용하도록 Auto Scaling 그룹을 업데이트하려면(AWS CLI)

다음 update-auto-scaling-group 명령은 지정된 시작 템플릿의 초기 버전을 사용하도록 지정된 Auto Scaling 그룹을 업데이트합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
```

```
--launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
```

CLI 명령을 사용하여 Auto Scaling 그룹을 업데이트하여 시작 템플릿을 사용하는 자세한 예는 섹션을 참조하세요시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트.

4단계: 인스턴스 교체

시작 구성을 시작 템플릿으로 교체하면 모든 새 인스턴스에서 새 시작 템플릿을 사용합니다. 기존 인스턴스는 영향을 받지 않습니다.

기존 인스턴스를 업데이트하려면 한 번에 몇 개의 인스턴스를 수동으로 교체하는 대신 인스턴스 새로 고침을 시작하여 Auto Scaling 그룹의 인스턴스를 교체할 수 있습니다. 자세한 내용은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다.(을)를 참조하세요. 그룹 규모가 큰 경우 인스턴스 새로 고침이 특히 유용할 수 있습니다.

또는 그룹의 종료 정책에 기반하여 자동 크기 조정을 통해 기존 인스턴스를 새 인스턴스로 점진적으로 교체하도록 허용하거나 인스턴스를 종료할 수 있습니다. 수동 종료는 Auto Scaling 그룹이 그룹에서 원하는 용량을 유지하기 위해 새 인스턴스를 시작하도록 강제합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하세요.

추가 정보

자세한 내용은 Amazon EC2 Auto Scaling에서 컴퓨팅 블로그의 Launch Configurations에 새 EC2 기능에 대한 지원을 더 이상 추가하지 않음을 참조하세요. AWS

시작 구성에서 시작 템플릿으로 AWS CloudFormation 스택을 마이그레이션하는 방법을 안내하는 주제는 섹션을 참조하세요스택을 시작 템플릿으로 마이그레이션 AWS CloudFormation.

스택을 시작 템플릿으로 마이그레이션 AWS CloudFormation

기존 AWS CloudFormation 스택 템플릿을 시작 구성에서 시작 템플릿으로 마이그레이션할 수 있습니다. 이렇게 하려면, 시작 템플릿을 기존 스택 템플릿에 직접 추가한 다음, 시작 템플릿을 스택 템플릿의 Auto Scaling 그룹과 연결합니다. 그런 다음, 수정된 템플릿을 사용하여 스택을 업데이트합니다.

시작 템플릿으로 마이그레이션할 때 이 주제에서는 CloudFormation 스택 템플릿의 시작 구성을 시작 템플릿으로 다시 작성하기 위한 지침을 제공하여 시간을 절약합니다. 시작 템플릿으로 시작 구성을 마이그레이션하기에 대한 자세한 내용은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작(을)를 참조하세요.

주제

- 시작 구성을 사용하는 Auto Scaling 그룹 찾기
- 시작 템플릿을 사용하도록 스택 업데이트
- 스택 리소스의 업데이트 동작 이해
- 마이그레이션 추적
- 시작 구성 매핑 참조

시작 구성을 사용하는 Auto Scaling 그룹 찾기

시작 구성을 사용하는 Auto Scaling 그룹을 찾으려면

- 다음 describe-auto-scaling-groups 명령을 사용하여 지정된 리전에서 시작 구성을 사용하는 Auto Scaling 그룹의 이름을 나열합니다. 결과를 CloudFormation 스택과 연결된 그룹으로 좁히는 `--filters` 옵션을 포함합니다(`aws:cloudformation:stack-name`태그 키로 필터링).

  ```
  aws autoscaling describe-auto-scaling-groups --region REGION \
    --filters Name=tag-key,Values=aws:cloudformation:stack-name \
    --query 'AutoScalingGroups[?LaunchConfigurationName!
  =`null`].AutoScalingGroupName'
  ```

  다음은 출력의 예입니다.

  ```
  [
      "{stack-name}-group-1",
      "{stack-name}-group-2",
      "{stack-name}-group-3"
  ]
  ```

  에서 마이그레이션할 Auto Scaling 그룹을 찾고 출력을 필터링하는 데 유용한 추가 AWS CLI 명령을 찾을 수 있습니다Auto Scaling 그룹을 마이그레이션하여 템플릿 시작.

> **Important**
>
> 스택 리소스`AWSEB`의 이름이 인 경우 이는 를 통해 생성된 것임을 의미합니다 AWS Elastic Beanstalk. 이러한 경우에는 Elastic Beanstalk가 시작 구성을 제거하고 해당 내용을 시작 템플릿으로 바꾸도록 Beanstalk 환경을 업데이트해야 합니다.

시작 템플릿을 사용하도록 스택 업데이트

이 섹션의 단계를 따라 다음을 수행합니다.

- 상응하는 시작 템플릿 속성을 사용하여 시작 구성을 시작 템플릿으로 다시 작성합니다.
- 새로운 시작 템플릿을 Auto Scaling 그룹에 연결합니다.
- 이 업데이트를 배포합니다.

스택 템플릿을 수정하고 스택을 업데이트하려면

1. AWS CloudFormation 사용 설명서의 스택 템플릿 수정에 설명된 것과 동일한 일반적인 스택 템플릿 수정 절차를 따릅니다.
2. 시작 구성을 시작 템플릿으로 다시 작성합니다. 다음 예를 참조하세요.

   예: 간단한 시작 구성

```
---
Resources:
  myLaunchConfig:
    Type: AWS::AutoScaling::LaunchConfiguration
    Properties:
      ImageId: ami-02354e95b3example
      InstanceType: t3.micro
      SecurityGroups:
        - !Ref EC2SecurityGroup
      KeyName: MyKeyPair
      BlockDeviceMappings:
        - DeviceName: /dev/xvda
          Ebs:
            VolumeSize: 150
            DeleteOnTermination: true
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum install -y aws-cfn-bootstrap
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource myASG
 --region ${AWS::Region}
```

   예: 상응하는 시작 템플릿

```
---
Resources:
  myLaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateName: !Sub ${AWS::StackName}-launch-template
      LaunchTemplateData:
        ImageId: ami-02354e95b3example
        InstanceType: t3.micro
        SecurityGroupIds:
          - Ref! EC2SecurityGroup
        KeyName: MyKeyPair
        BlockDeviceMappings:
          - DeviceName: /dev/xvda
            Ebs:
              VolumeSize: 150
              DeleteOnTermination: true
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash -x
            yum install -y aws-cfn-bootstrap
            /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
 myASG --region ${AWS::Region}
```

Amazon이 EC2 지원하는 모든 속성에 대한 참조 정보는 AWS CloudFormation 사용 설명서의 AWS::EC2::LaunchTemplate를 참조하세요.

시작 템플릿에 값이 `!Sub ${AWS::StackName}-launch-template`인 `LaunchTemplateName` 속성이 어떻게 포함되어 있는지 확인합니다. 이것은 스택 이름을 시작 템플릿의 이름에 포함시키길 원할 경우 필요합니다.

3. 시작 구성에 **`IamInstanceProfile`** 속성이 있는 경우 속성을 구조로 변환하고 인스턴스 프로파일ARN의 이름 또는 를 지정해야 합니다. 예제는 AWS::::EC2:::LaunchTemplate 참조하세요.
4. 시작 구성에 **`AssociatePublicIpAddress`**, **`InstanceMonitoring`** 또는 **`PlacementTenancy`** 속성이 있는 경우, 이러한 내용들을 구조로 변환해야 합니다. 예제는 AWS::::EC2::LaunchTemplate 참조하세요.

   단, Auto Scaling 그룹에 사용한 서브넷의 `MapPublicIpOnLaunch` 속성 값이 시작 구성의 `AssociatePublicIpAddress` 속성 값과 일치하는 경우는 예외입니다. 이 경우 `AssociatePublicIpAddress` 속성을 무시할 수 있습니다. `AssociatePublicIpAddress`

속성은 시작 시 인스턴스가 퍼블릭 IPv4 주소를 수신할지 여부를 변경하기 위해 `MapPublicIpOnLaunch` 속성을 재정의하는 데만 사용됩니다.

5. **`SecurityGroups`** 속성의 보안 그룹을 시작 템플릿의 두 위치 중 하나로 복사할 수 있습니다. 일반적으로, 보안 그룹을 `SecurityGroupIds` 속성에 복사합니다. 하지만, 시작 템플릿 내에 `NetworkInterfaces` 구조를 생성하여 `AssociatePublicIpAddress` 속성을 지정하는 경우에는 보안 그룹을 네트워크 인터페이스의 `Groups` 속성에 대신 복사해야 합니다.
6. 가 로 **`NoDevice`** 설정된 시작 구성에 `BlockDeviceMapping` 구조가 있는 `true`경우 Amazon이 디바이스를 생략하도록 시작 템플릿`NoDevice`에 EC2 에 대한 빈 문자열을 지정해야 합니다.
7. 시작 구성에 **`SpotPrice`** 속성이 있는 경우, 시작 템플릿에서 해당 속성을 생략하는 것이 좋습니다. 스팟 인스턴스가 현재 스팟 가격으로 시작됩니다. 이 가격은 온디맨드 가격을 초과하지 않습니다.

   스팟 인스턴스를 요청하려면 상호 배타적인 두 가지 옵션이 있습니다.

   - 첫 번째는 시작 템플릿의 `InstanceMarketOptions` 구조를 사용하는 것입니다(권장하지 않음). 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::EC2:::LaunchTemplate InstanceMarketOptions를 참조하세요.
   - 다른 방법은 Auto Scaling 그룹에 `MixedInstancesPolicy` 구조를 추가하는 것입니다. 이렇게 하면 요청을 만드는 방법에 대한 더 많은 옵션이 제공됩니다. 시작 템플릿의 스팟 인스턴스 요청은 Auto Scaling 그룹당 두 개 이상의 인스턴스 유형 선택을 지원하지 않습니다. 하지만, 혼합 인스턴스 정책은 Auto Scaling 그룹당 두 개 이상의 인스턴스 유형 선택을 지원합니다. 스팟 인스턴스 요청은 두 개 이상의 인스턴스 유형 중에서 선택할 수 있는 이점이 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS::AutoScaling:::AutoScalingGroup MixedInstancesPolicy를 참조하세요.
8. AWS::::AutoScaling:::AutoScalingGroup 리소스에서 **`LaunchConfigurationName`** 속성을 제거합니다. 시작 템플릿을 그 자리에 추가합니다.

   다음 예제에서 참조 내장 함수는 논리적 ID 를 사용하여 AWS:::EC2:LaunchTemplate 리소스의 ID를 가져옵니다`myLaunchTemplate`. GetAtt 함수는 `Version` 속성에 대한 시작 템플릿의 최신 버전 번호(예: 1)를 가져옵니다.

   예: 혼합 인스턴스 정책 미사용

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
```

```
      LaunchTemplate:
        LaunchTemplateId: !Ref myLaunchTemplate
        Version: !GetAtt myLaunchTemplate.LatestVersionNumber
      ...
```

예: 혼합 인스턴스 정책 사용

```
---
Resources:
  myASG:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MixedInstancesPolicy:
        LaunchTemplate:
          LaunchTemplateSpecification:
            LaunchTemplateId: !Ref myLaunchTemplate
            Version: !GetAtt myLaunchTemplate.LatestVersionNumber
      ...
```

Amazon EC2 Auto Scaling에서 지원하는 모든 속성에 대한 참조 정보는 AWS CloudFormation 사용 설명서의 AWS::AutoScaling::AutoScalingGroup를 참조하세요.

9. 이러한 업데이트를 배포할 준비가 되면 절차에 따라 CloudFormation 수정된 스택 템플릿으로 스택을 업데이트합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 스택 템플릿 수정을 참조하세요.

스택 리소스의 업데이트 동작 이해

CloudFormation 는 제공한 업데이트된 템플릿과 이전 버전의 스택 템플릿에서 설명한 리소스 구성 간의 변경 사항을 비교하여 스택 리소스를 업데이트합니다. 변경되지 않은 리소스 구성은 업데이트 프로세스 동안 영향을 받지 않는 상태로 유지됩니다.

CloudFormation 는 Auto Scaling 그룹의 UpdatePolicy 속성을 지원합니다. 업데이트 중에 이 로 `UpdatePolicy` 설정된 경우 이 절차의 단계를 수행한 후 가 `InService` 인스턴스를 `AutoScalingRollingUpdate` CloudFormation 대체합니다. `UpdatePolicy` 가 로 설정된 경우 Auto Scaling 그룹과 해당 웜 풀(있는 경우)을 `AutoScalingReplacingUpdate` CloudFormation 대체합니다.

Auto Scaling 그룹에 대한 `UpdatePolicy` 속성을 지정하지 않은 경우 시작 템플릿이 올바른지 확인되지만 Auto Scaling 그룹의 인스턴스에 변경 사항을 배포하지 CloudFormation 는 않습니다. 모든 새 인

스턴스는 시작 템플릿을 사용하지만, 기존 인스턴스는 (시작 구성이 존재하지 않음에도) 원래 시작된 시작 구성으로 계속 실행됩니다. 단, 구매 옵션을 변경하는 경우(예를 들어, 혼합 인스턴스 정책을 추가하는 방식을 통해)는 예외입니다. 이 경우 Auto Scaling 그룹은 새 구매 옵션에 맞게 기존 인스턴스를 새 인스턴스로 점진적으로 교체합니다.

시작 구성에서 시작 템플릿으로 이동하기 위해 변경 사항을 롤백해야 하는 경우 롤백 작업을 테스트해야 합니다.

마이그레이션 추적

마이그레이션을 추적하려면

1. AWS CloudFormation 콘솔에서 업데이트한 스택을 선택한 다음 [이벤트(Events)] 탭을 선택하여 스택 이벤트를 봅니다.
2. 이벤트 목록을 최신 이벤트로 업데이트하려면 CloudFormation 콘솔에서 새로 고침 버튼을 선택합니다.
3. 스택이 업데이트되는 동안, 각 리소스 업데이트에 대한 여러 이벤트를 확인하게 됩니다. 시작 템플릿을 만들려고 할 때 상태 사유 열에 문제가 있음을 나타내는 예외가 표시되는 경우, 잠재적 원인은 Amazon EC2 Auto Scaling: 시작 템플릿 문제 해결(을)를 참조하세요.
4. (선택 사항) `UpdatePolicy` 속성 사용에 따라 Amazon EC2 콘솔의 Auto Scaling 그룹 페이지에서 Auto Scaling 그룹의 진행 상황을 모니터링할 수 있습니다. Auto Scaling 그룹을 선택합니다. Activity(활동) 탭에서 Activity history(활동 기록)의 Status(상태) 열에 Auto Scaling 그룹이 성공적으로 인스턴스를 시작 또는 해지했는지와 크기 조정 활동이 아직 진행 중인지 여부가 표시됩니다.
5. 스택 업데이트가 완료되면 에서 `UPDATE_COMPLETE` 스택 이벤트를 CloudFormation 발행합니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 스택 업데이트 진행 상황 모니터링을 참조하세요.
6. 스택 업데이트가 완료되면 Amazon EC2 콘솔의 시작 템플릿 페이지와 시작 구성 페이지를 엽니다. 새 시작 템플릿이 생성되고 시작 구성이 삭제된 것을 확인할 수 있습니다.

시작 구성 매핑 참조

참조 목적으로 다음 표에는 AWS::::::: 리소스의 모든 최상위 속성과 ::AutoScalingLaunchConfiguration:AWS:EC2::LaunchTemplate 리소스의 해당 속성 나열되어 있습니다.

시작 구성 소스 속성	시작 템플릿 대상 속성
AssociatePublicIpAddress	NetworkInterfaces.AssociatePublicIpAddress
BlockDeviceMappings	BlockDeviceMappings
ClassicLinkVPCId	사용할 수 없음[1]
ClassicLinkVPCSecurityGroups	사용할 수 없음[1]
EbsOptimized	EbsOptimized
IamInstanceProfile	IamInstanceProfile.Arn 또는 IamInstanceProfile.Name 중 하나, 하지만 둘 다는 아님
ImageId	ImageId
InstanceId	InstanceId
InstanceMonitoring	Monitoring.Enabled
InstanceType	InstanceType
KernelId	KernelId
KeyName	KeyName
LaunchConfigurationName	LaunchTemplateName
MetadataOptions	MetadataOptions
PlacementTenancy	Placement.Tenancy
RamDiskId	RamDiskId
SecurityGroups	SecurityGroupIds 또는 NetworkInterfaces.Groups 중 하나, 하지만 둘 다는 아님

시작 구성 소스 속성	시작 템플릿 대상 속성
`SpotPrice`	`InstanceMarketOptions.SpotOptions.MaxPrice`
`UserData`	`UserData`

1 EC2-Classic을 더 이상 사용할 수 없으므로 `ClassicLinkVPCId` 및 `ClassicLinkVPCSecurityGroups` 속성을 시작 템플릿에서 사용할 수 없습니다.

를 사용하여 시작 템플릿을 생성하고 관리하는 예제 AWS CLI

AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 를 통해 시작 템플릿을 생성하고 관리할 수 있습니다SDKs. 이 섹션에서는 에서 Amazon EC2 Auto Scaling에 대한 시작 템플릿을 생성하고 관리하는 예를 보여줍니다 AWS CLI.

내용

- 사용 예
- 기본 시작 템플릿 생성
- 시작 시 인스턴스에 태그를 지정하는 태그 지정
- 인스턴스에 전달할 IAM 역할 지정
- 퍼블릭 IP 주소 배정
- 시작 시 인스턴스를 구성하는 사용자 데이터 스크립트 지정
- 블록 디바이스 매핑 지정
- 외부 공급 업체의 소프트웨어 라이선스를 가져오기 위한 전용 호스트 지정
- 기존 네트워크 인터페이스 지정
- 여러 네트워크 인터페이스 생성
- 시작 템플릿 관리
- 시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트

사용 예

```
{
```

```
    "LaunchTemplateName": "my-template-for-auto-scaling",
    "VersionDescription": "test description",
    "LaunchTemplateData": {
        "ImageId": "ami-04d5cc9b88example",
        "InstanceType": "t2.micro",
        "SecurityGroupIds": [
            "sg-903004f88example"
        ],
        "KeyName": "MyKeyPair",
        "Monitoring": {
            "Enabled": true
        },
        "Placement": {
            "Tenancy": "dedicated"
        },
        "CreditSpecification": {
            "CpuCredits": "unlimited"
        },
        "MetadataOptions": {
            "HttpTokens": "required",
            "HttpPutResponseHopLimit": 1,
            "HttpEndpoint": "enabled"
        }
    }
}
```

기본 시작 템플릿 생성

기본 시작 템플릿을 생성하려면 다음과 create-launch-template 같이 다음과 같이 명령을 사용합니다.

- `ami-04d5cc9b88example` 를 인스턴스를 시작할 AMI 의 ID로 바꿉니다.
- `t2.micro` 를 지정한 와 호환되는 인스턴스 유형으로 바꿉AMI니다.

이 예제에서는 이름이 인 시작 템플릿을 생성합니다.*my-template-for-auto-scaling*. 이 시작 템플릿에서 생성된 인스턴스가 기본 에서 시작되면 기본적으로 퍼블릭 IP 주소를 수신VPC합니다. 인스턴스가 기본값이 아닌 에서 시작VPC되는 경우 기본적으로 퍼블릭 IP 주소를 수신하지 않습니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

따옴표 JSON형식의 파라미터에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 에서 문자열로 따옴표 사용을 AWS CLI 참조하세요.

또는 구성 파일에서 JSON형식이 지정된 파라미터를 지정할 수 있습니다.

다음 예에서는 시작 템플릿 파라미터 값을 위해 구성 파일을 참조하는 기본 시작 템플릿을 생성합니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data file://config.json
```

`config.json`의 콘텐츠:

```
{
    "ImageId":"ami-04d5cc9b88example",
    "InstanceType":"t2.micro"
}
```

시작 시 인스턴스에 태그를 지정하는 태그 지정

다음 예제에서는 시작 시 인스턴스에 태그(예: `purpose=webserver`)를 추가합니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"TagSpecifications":[{"ResourceType":"instance","Tags":
[{"Key":"purpose","Value":"webserver"}]}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.
```

> **Note**
>
> 시작 템플릿에 인스턴스 태그를 지정하고 Auto Scaling 그룹의 태그를 해당 인스턴스로 전파하도록 선택한 경우 모든 태그가 병합됩니다. 시작 템플릿의 태그와 Auto Scaling 그룹의 태그에 대해 동일한 태그 키가 지정된 경우 그룹의 태그 값이 우선합니다.

인스턴스에 전달할 IAM 역할 지정

다음 예제에서는 시작 시 인스턴스에 전달할 IAM 역할과 연결된 인스턴스 프로파일의 이름을 지정합니다. 자세한 내용은 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 단원을 참조하십시오.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data '{"IamInstanceProfile":{"Name":"my-instance-
profile"},"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

퍼블릭 IP 주소 배정

다음 create-launch-template 예제에서는 기본값이 아닌 에서 시작된 인스턴스에 퍼블릭 주소를 할당하도록 시작 템플릿을 구성합니다VPC.

> **Note**
>
> 네트워크 인터페이스를 지정할 때 Auto Scaling 그룹이 인스턴스를 시작할 VPC 의 보안 그룹에 `Groups` 해당하는 값을 지정합니다. VPC 서브넷을 Auto Scaling 그룹의 속성으로 지정합니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","Instanc
```

시작 시 인스턴스를 구성하는 사용자 데이터 스크립트 지정

다음 예제에서는 시작 시 인스턴스를 구성하는 base64로 인코딩된 문자열로 사용자 데이터 스크립트를 지정합니다. create-launch-template 명령에는 base64로 인코딩된 사용자 데이터가 필요합니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
--launch-template-data
 '{"UserData":"IyEvYmluL2Jhc...","ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"}'
```

블록 디바이스 매핑 지정

다음 create-launch-template 예제에서는 블록 디바이스 매핑을 사용하여 시작 템플릿을 생성합니다. 에 매핑된 22기가바이트 EBS 볼륨입니다`/dev/xvdcz`. `/dev/xvdcz` 볼륨은 범용SSD(gp2) 볼륨 유형을 사용하며 연결된 인스턴스를 종료할 때 삭제됩니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"BlockDeviceMappings":[{"DeviceName":"/dev/xvdcz","Ebs":
{"VolumeSize":22,"VolumeType":"gp2","DeleteOnTermination":true}}],"ImageId":"ami-04d5cc9b88exam
```

외부 공급 업체의 소프트웨어 라이선스를 가져오기 위한 전용 호스트 지정

host 테넌시를 지정한 경우 호스트 리소스 그룹 및 License Manager 라이센스 구성을 지정하여 외부 공급 업체에서 적합한 소프트웨어 라이선스를 가져올 수 있습니다. 그런 다음 다음 create-launch-template 명령을 사용하여 EC2 인스턴스에서 라이선스를 사용할 수 있습니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"Placement":
{"Tenancy":"host","HostResourceGroupArn":"arn"},"LicenseSpecifications":
[{"LicenseConfigurationArn":"arn"}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro"
```

기존 네트워크 인터페이스 지정

다음 create-launch-template 예제에서는 기존 네트워크 인터페이스를 사용하도록 기본 네트워크 인터페이스를 구성합니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"DeviceIndex":0,"NetworkInterfaceId":"eni-
b9a5ac93","DeleteOnTermination":false}],"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.mi
```

여러 네트워크 인터페이스 생성

다음 create-launch-template 예제에서는 보조 네트워크 인터페이스를 추가합니다. 주 네트워크 인터페이스의 디바이스 인덱스는 0이고 보조 네트워크 인터페이스의 디바이스 인덱스는 1입니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":[{"DeviceIndex":0,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true},{"DeviceIndex":1,"Groups":
["sg-903004f88example"],"DeleteOnTermination":true}],"ImageId":"ami-04d5cc9b88example","Instanc
```

여러 네트워크 카드 및 탄력적 패브릭 어댑터(EFAs)를 지원하는 인스턴스 유형을 사용하는 경우 보조 네트워크 카드에 보조 인터페이스를 추가하고 다음 create-launch-template 명령을 EFA 사용하여 를 활성화할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 EFA 시작 템플릿에 추가를 참조하세요.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling --
version-description version1 \
  --launch-template-data '{"NetworkInterfaces":
[{"NetworkCardIndex":0,"DeviceIndex":0,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true},
{"NetworkCardIndex":1,"DeviceIndex":1,"Groups":
["sg-7c2270198example"],"InterfaceType":"efa","DeleteOnTermination":true}],"ImageId":"ami-09d95
```

> **Warning**
>
> p4d.24xlarge 인스턴스 유형은 이 섹션의 다른 예제보다 비용이 많이 발생합니다. P4d 인스턴스 요금에 대한 자세한 내용은 Amazon EC2 P4d 인스턴스 요금 섹션을 참조하세요.

> **Note**
>
> 동일한 서브넷의 여러 네트워크 인터페이스를 인스턴스에 연결하면 특히, Amazon Linux가 아닌 변형을 사용하는 인스턴스에서 비대칭 라우팅이 도입될 수 있습니다. 이러한 유형의 구성이 필요한 경우 OS 내에서 보조 네트워크 인터페이스를 구성해야 합니다. 예를 들어 AWS 지식 센터의 Ubuntu EC2 인스턴스에서 보조 네트워크 인터페이스를 작동하려면 어떻게 해야 합니까?를 참조하세요.

시작 템플릿 관리

에는 시작 템플릿을 관리하는 데 도움이 되는 몇 가지 다른 명령이 AWS CLI 포함되어 있습니다.

내용

- 시작 템플릿 나열 및 설명
- 시작 템플릿 버전 생성
- 시작 템플릿 버전 삭제
- 시작 템플릿 삭제

시작 템플릿 나열 및 설명

두 가지 AWS CLI 명령을 사용하여 시작 템플릿에 대한 정보를 가져올 수 있습니다describe-launch-templatesdescribe-launch-template-versions.

describe-launch-templates 명령을 사용하면 생성한 시작 템플릿의 목록을 가져올 수 있습니다. 옵션을 사용하여 시작 템플릿 이름에 대한 결과를 필터링하거나, 시간, 태그 키 또는 태그 키-값 조합을 만들 수 있습니다. 이 명령은 시작 템플릿 식별자, 최신 버전, 기본 버전 등을 비롯하여 시작 템플릿에 대한 요약 정보를 반환합니다.

다음 예제에서는 지정된 시작 템플릿에 대한 요약을 제공합니다.

```
aws ec2 describe-launch-templates --launch-template-names my-template-for-auto-scaling
```

다음은 응답의 예입니다.

```
{
    "LaunchTemplates": [
        {
            "LaunchTemplateId": "lt-068f72b729example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "CreateTime": "2020-02-28T19:52:27.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersionNumber": 1,
            "LatestVersionNumber": 1
        }
    ]
}
```

시작 템플릿 하나로 출력을 제한하기 위해 `--launch-template-names` 옵션을 사용하지 않는 경우 모든 시작 템플릿에 대한 정보가 반환됩니다.

다음 describe-launch-template-versions 명령은 지정된 시작 템플릿의 버전을 설명하는 정보를 제공합니다.

```
aws ec2 describe-launch-template-versions --launch-template-id lt-068f72b729example
```

다음은 응답의 예입니다.

```
{
```

```
    "LaunchTemplateVersions": [
        {
            "VersionDescription": "version1",
            "LaunchTemplateId": "lt-068f72b729example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "VersionNumber": 1,
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "LaunchTemplateData": {
                "TagSpecifications": [
                    {
                        "ResourceType": "instance",
                        "Tags": [
                            {
                                "Key": "purpose",
                                "Value": "webserver"
                            }
                        ]
                    }
                ],
                "ImageId": "ami-04d5cc9b88example",
                "InstanceType": "t2.micro",
                "NetworkInterfaces": [
                    {
                        "DeviceIndex": 0,
                        "DeleteOnTermination": true,
                        "Groups": [
                            "sg-903004f88example"
                        ],
                        "AssociatePublicIpAddress": true
                    }
                ]
            },
            "DefaultVersion": true,
            "CreateTime": "2020-02-28T19:52:27.000Z"
        }
    ]
}
```

시작 템플릿 버전 생성

다음 create-launch-template-version 명령은 시작 템플릿의 버전 1을 기반으로 새 시작 템플릿 버전을 생성하고 다른 AMI ID를 지정합니다.

```
aws ec2 create-launch-template-version --launch-template-id lt-068f72b729example --
version-description version2 \
  --source-version 1 --launch-template-data "ImageId=ami-c998b6b2example"
```

시작 템플릿의 기본 버전을 설정하려면 modify-launch-template 명령을 사용합니다.

시작 템플릿 버전 삭제

다음 delete-launch-template-versions 명령은 지정된 시작 템플릿 버전을 삭제합니다.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b729example --
versions 1
```

시작 템플릿 삭제

시작 템플릿이 더 이상 필요하지 않은 경우 다음 delete-launch-template 명령을 사용하여 삭제할 수 있습니다. 시작 템플릿을 삭제하면 모든 버전이 삭제됩니다.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b729example
```

시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트

update-auto-scaling-group 명령을 사용하여 기존 Auto Scaling 그룹에 시작 템플릿을 추가할 수 있습니다.

최신 버전의 시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트

다음 update-auto-scaling-group 명령은 지정된 시작 템플릿의 최신 버전을 사용하도록 지정된 Auto Scaling 그룹을 업데이트합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateId=lt-068f72b729example,Version='$Latest'
```

특정 버전의 시작 템플릿을 사용하도록 Auto Scaling 그룹 업데이트

다음 update-auto-scaling-group 명령은 지정된 시작 템플릿의 특정 버전을 사용하도록 지정된 Auto Scaling 그룹을 업데이트합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
```

```
--launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

AMI IDs 시작 템플릿 대신 AWS Systems Manager 파라미터 사용

이 섹션에서는 Amazon Machine Image(AMI) ID를 참조하는 파라미터를 지정하는 AWS Systems Manager 시작 템플릿을 생성하는 방법을 보여줍니다. 동일한 에 저장된 파라미터 AWS 계정, 다른 에서 공유된 파라미터 AWS 계정또는 에서 AMI 관리하는 퍼블릭에 대한 퍼블릭 파라미터를 사용할 수 있습니다 AWS.

Systems Manager 파라미터를 사용하면 AMI ID가 변경될 때마다 새 시작 템플릿 또는 새 버전의 시작 템플릿을 생성할 필요 AMI IDs 없이 Auto Scaling 그룹을 업데이트하여 새 를 사용할 수 있습니다. AMI가 최신 운영 체제 또는 소프트웨어 업데이트로 업데이트되는 경우와 같이 이러한 사항은 정기적으로 변경될 IDs 수 있습니다.

의 기능인 Parameter Store를 AWS Systems Manager 사용하여 자체 Systems Manager 파라미터를 생성, 업데이트 또는 삭제할 수 있습니다. 시작 템플릿에 사용하려면 먼저 Systems Manager 파라미터를 생성해야 합니다. 시작하려면 데이터 형식 로 파라미터를 생성하고 `aws:ec2:image`값에 의 ID를 입력합니다AMI. AMI ID의 형식은 `ami-<identifier>`입니다. 예를 들어 입니다`ami-123example456`. 올바른 AMI ID는 인스턴스 유형과 Auto Scaling 그룹을 시작하는 AWS 리전 위치에 따라 달라집니다.

AMI ID에 유효한 파라미터를 생성하는 방법에 대한 자세한 내용은 Systems Manager 파라미터 생성을 참조하세요.

에 대한 파라미터를 지정하는 시작 템플릿 생성 AMI

에 대한 파라미터를 지정하는 시작 템플릿을 생성하려면 다음 방법 중 하나를 AMI사용합니다.

Console

AWS Systems Manager 파라미터를 사용하여 시작 템플릿을 생성하려면

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창에서 시작 템플릿을 선택한 다음 시작 템플릿 생성을 선택합니다.
3. Launch template name에 대해 시작 템플릿의 설명이 포함된 이름을 입력하세요.
4. 애플리케이션 및 OS 이미지(Amazon Machine Image)에서 자세히 찾아보기 AMIs를 선택합니다.

5. 검색 창 오른쪽에 있는 화살표 버튼을 선택한 다음 사용자 지정 값/Systems Manager 파라미터 지정을 선택하세요.
6. 사용자 지정 값 또는 Systems Manager 파라미터 지정 대화 상자에서 다음을 수행합니다.

   a. AMI ID 또는 Systems Manager 파라미터 문자열 에 다음 형식 중 하나를 사용하여 Systems Manager 파라미터 이름을 입력합니다.

      공용 파라미터 참조:

      - **`resolve:ssm:`*`public-parameter`***

      동일한 계정에 저장된 파라미터 참조:

      - **`resolve:ssm:`*`parameter-name`***
      - **`resolve:ssm:`*`parameter-name`*`:`*`version-number`***
      - **`resolve:ssm:`*`parameter-name`*`:`*`label`***

      다른 AWS 계정에서 공유된 파라미터 참조:

      - **`resolve:ssm:`*`parameter-ARN`***
      - **`resolve:ssm:`*`parameter-ARN`*`:`*`version-number`***
      - **`resolve:ssm:`*`parameter-ARN`*`:`*`label`***

   b. 저장(Save)을 선택합니다.
7. 필요에 따라 다른 시작 템플릿 설정을 구성하고 시작 템플릿 생성을 선택합니다. 자세한 내용은 Auto Scaling 그룹에 대한 시작 템플릿 생성 단원을 참조하십시오.

AWS CLI

Systems Manager 파라미터를 지정하는 시작 템플릿을 생성하려면 다음 예제 명령 중 하나를 사용할 수 있습니다. 각각 바꾸기 *`user input placeholder`* 자신의 정보를 사용합니다.

예: AWS소유 퍼블릭 파라미터를 지정하는 시작 템플릿 생성

`resolve:ssm:`*`public-parameter`* 구문을 사용합니다. 여기서 `resolve:ssm`은 표준 접두사이고 *`public-parameter`*는 퍼블릭 파라미터의 경로와 이름입니다.

이 예제에서 시작 템플릿은 AWS에서 제공하는 퍼블릭 파라미터를 사용하여 프로필에 구성된 AMI의 최신 Amazon Linux 2를 사용하여 인스턴스 AWS 리전 를 시작합니다.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
 --version-description version1 \
  --launch-template-data file://config.json
```

`config.json`의 콘텐츠:

```
{
    "ImageId":"resolve:ssm:/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-
x86_64-gp2",
    "InstanceType":"t2.micro"
}
```

다음은 응답의 예입니다.

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-089c023a30example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2022-12-28T19:52:27.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

예: 동일한 계정에 저장된 파라미터를 지정하는 시작 템플릿 생성

`resolve:ssm:`*`parameter-name`* 구문을 사용합니다. 여기서 `resolve:ssm`은 표준 접두사이고 *`parameter-name`*은 Systems Manager 파라미터 이름입니다.

다음 예제에서는 라는 기존 Systems Manager 파라미터에서 AMI ID를 가져오는 시작 템플릿을 생성합니다*`golden-ami`*.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling \
  --launch-template-data file://config.json
```

`config.json`의 콘텐츠:

```
{
    "ImageId":"resolve:ssm:golden-ami",
    "InstanceType":"t2.micro"
```

```
}
```

지정되지 않은 경우 파라미터의 기본 버전은 최신 버전입니다.

다음 예제에서는 의 특정 버전을 참조합니다.*##-##* 파라미터. 이 예제에서는 버전을 사용합니다.*3* 의 *##-##* 유효한 버전 번호를 사용할 수 있습니다.

```
{
    "ImageId":"resolve:ssm:golden-ami:3",
    "InstanceType":"t2.micro"
}
```

다음 유사한 예제에서는 파라미터 레이블을 참조합니다.*prod* 의 특정 버전에 매핑되는 *##-##* 파라미터.

```
{
    "ImageId":"resolve:ssm:golden-ami:prod",
    "InstanceType":"t2.micro"
}
```

출력의 예시는 다음과 같습니다.

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b724example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2022-12-27T17:11:21.000Z",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

예: 다른 에서 공유된 파라미터를 지정하는 시작 템플릿 생성 AWS 계정

다음 구문을 사용합니다. 여기서 `resolve:ssm:`*parameter-ARN*`resolve:ssm`는 표준 접두사이고 *parameter-ARN*는 Systems Manager 파라미터ARN의 입니다.

다음 예제에서는 의 를 사용하여 기존 Systems Manager 파라미터에서 AMI ID를 가져오는 시작 템플릿을 생성합니다ARN*arn:aws:ssm:us-east-2:123456789012:parameter/MyParameter*.

```
aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
 --version-description version1 \
  --launch-template-data file://config.json
```

`config.json`의 콘텐츠:

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter",
    "InstanceType":"t2.micro"
}
```

지정되지 않은 경우 파라미터의 기본 버전은 최신 버전입니다.

다음 예제에서는 *MyParameter* 파라미터의 특정 버전을 참조합니다. 이 예제에서는 버전을 사용합니다.*3 MyParameter* 유효한 버전 번호를 사용할 수 있습니다.

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter:3",
    "InstanceType":"t2.micro"
}
```

다음 유사한 예제에서는 파라미터 레이블을 참조합니다.*prod* 는 *MyParameter* 파라미터의 특정 버전에 매핑됩니다.

```
{
    "ImageId":"resolve:ssm:arn:aws:ssm:us-east-2:123456789012:parameter/
MyParameter:prod",
    "InstanceType":"t2.micro"
}
```

다음은 응답의 예입니다.

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-00f93d4588example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreateTime": "2024-01-08T12:43:21.000Z",
```

```
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersionNumber": 1,
            "LatestVersionNumber": 1
        }
    }
```

시작 템플릿의 파라미터 스토어에서 파라미터를 지정하려면 지정된 파라미터에 대한 `ssm:GetParameters` 권한이 있어야 합니다. 시작 템플릿을 사용하는 모든 사용자에게는 파라미터 값을 검증하기 위한 `ssm:GetParameters` 권한도 필요합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 IAM 정책을 사용하여 Systems Manager 파라미터에 대한 액세스 제한을 참조하세요.

시작 템플릿이 올바른 AMI ID를 가져오는지 확인

describe-launch-template-versions 명령을 사용하고 `--resolve-alias` 옵션을 포함하여 파라미터를 실제 AMI ID로 확인합니다.

```
aws ec2 describe-launch-template-versions --launch-template-name my-template-for-auto-scaling \
   --versions $Default --resolve-alias
```

이 예제는 의 AMI ID를 반환합니다`ImageId`. 이 시작 템플릿을 사용하여 인스턴스를 시작하면 AMI ID가 로 확인됩니다`ami-0ac394d6a3example`.

```
{
    "LaunchTemplateVersions": [
        {
            "LaunchTemplateId": "lt-089c023a30example",
            "LaunchTemplateName": "my-template-for-auto-scaling",
            "VersionNumber": 1,
            "CreateTime": "2022-12-28T19:52:27.000Z",
            "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
            "DefaultVersion": true,
            "LaunchTemplateData": {
                "ImageId": "ami-0ac394d6a3example",
                "InstanceType": "t2.micro",
            }
        }
    ]
}
```

관련 리소스

시작 템플릿에서 Systems Manager 파라미터를 지정하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 AMI ID 대신 Systems Manager 파라미터 사용을 참조하세요.

Systems Manager 파라미터 작업에 대한 자세한 내용은 Systems Manager 설명서의 다음 참조 자료에서 확인할 수 있습니다.

- 파라미터 버전 및 레이블을 생성하려면 파라미터 버전 작업 및 파라미터 레이블 작업을 참조하세요.
- Amazon 에서 지원하는 AMI 퍼블릭 파라미터를 조회하는 방법에 대한 자세한 내용은 퍼블릭 파라미터 호출을 EC2참조하세요. AMI
- 다른 AWS 계정과 파라미터를 공유하거나 를 통해 파라미터를 공유하는 방법에 대한 자세한 내용은 공유 파라미터 작업을 AWS Organizations참조하세요.
- 파라미터가 성공적으로 생성되었는지 모니터링하는 방법에 대한 자세한 내용은 Amazon Machine Image에 대한 기본 파라미터 지원을 IDs참조하세요.

제한 사항

Systems Manager 파라미터를 사용할 때는 다음 제한 사항에 유의하세요.

- Amazon EC2 Auto Scaling은 파라미터AMIIDs로 지정만 지원합니다.
- Systems Manager 파라미터를 지정하는 시작 템플릿을 사용하여 속성 기반 인스턴스 유형 선택으로 혼합 인스턴스 그룹을 생성하거나 업데이트하는 것은 지원되지 않습니다.
- Auto Scaling 그룹이 Systems Manager 파라미터를 지정하는 시작 템플릿을 사용하는 경우 원하는 구성으로 또는 매칭 건너뛰기를 사용하여 인스턴스 새로 고침을 시작할 수 없습니다.
- Auto Scaling 그룹을 생성하거나 업데이트하기 위한 각 호출에서 Amazon EC2 Auto Scaling은 시작 템플릿의 Systems Manager 파라미터를 확인합니다. 고급 파라미터 또는 더 높은 처리량 제한을 사용하는 경우 파라미터 스토어에 자주 호출하면(즉, `GetParameters` 작업) 파라미터 스토어 API 상호 작용당 요금이 발생하므로 Systems Manager에 대한 비용이 증가할 수 있습니다. 자세한 내용은 AWS Systems Manager 요금을 참조하세요.

Auto Scaling 시작 구성

> **Important**
>
> 2022년 12월 31일 이후에 릴리스된 새로운 Amazon EC2 인스턴스 유형으로는 `CreateLaunchConfiguration`을 호출할 수 없습니다. 또한 2023년 6월 1일 이후에 생성된 새 계정에는 이 콘솔을 통해 새 시작 구성을 만들 수 있는 옵션이 없습니다. 2024년 10월 1일부터 새 계정은 콘솔, API, CLI 및 를 사용하여 새 시작 구성을 생성할 수 없습니다. CloudFormation 현재 또는 미래에 새 시작 구성을 만들 필요가 없도록 시작 템플릿으로 마이그레이션하십시오. Auto Scaling 그룹을 위한 시작 템플릿 생성에 대한 자세한 설명은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작을 참조하세요.

시작 구성은 Auto Scaling 그룹에서 EC2 인스턴스를 시작하는 데 사용하는 인스턴스 구성 템플릿입니다. 시작 구성을 만들 때 인스턴스에 대한 정보를 지정합니다. Amazon Machine Image(AMI)의 ID, 인스턴스 유형, 키 페어, 하나 이상의 보안 그룹, 블록 디바이스 매핑 등을 포함시키세요. 이전에 EC2 인스턴스를 시작한 적이 있었다면 바로 이러한 정보를 지정하여 인스턴스를 시작했을 것입니다.

여러 개의 Auto Scaling 그룹을 가진 시작 구성을 지정할 수 있습니다. 하지만 Auto Scaling 그룹에 대해서는 한 번에 한 개의 시작 구성만 지정할 수 있으며 시작 구성을 한 번 생성한 후에는 수정할 수 없습니다. Auto Scaling 그룹의 시작 구성을 변경하려면 시작 구성을 생성한 다음 해당 구성으로 Auto Scaling 그룹을 업데이트해야 합니다.

내용

- 시작 구성 생성
- Auto Scaling 그룹에 대한 시작 구성 변경

시작 구성 생성

> **Important**
>
> 2022년 12월 31일 이후에 릴리스된 새로운 Amazon EC2 인스턴스 유형으로는 `CreateLaunchConfiguration`을 호출할 수 없습니다. 또한 2023년 6월 1일 이후에 생성된 새 계정에는 이 콘솔을 통해 새 시작 구성을 만들 수 있는 옵션이 없습니다. 2024년 10월 1일부터 새 계정은 콘솔, API, CLI 및 를 사용하여 새 시작 구성을 생성할 수 없습니다. CloudFormation 현재 또는 미래에 새 시작 구성을 만들 필요가 없도록 시작 템플릿으로 마이

그레이션하십시오. Auto Scaling 그룹을 위한 시작 템플릿 생성에 대한 자세한 설명은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작을 참조하세요.

이 항목에서는 시작 구성을 만드는 방법을 설명합니다.

시작 구성을 생성한 후에는 수정할 수 없습니다. 대신 새 시작 구성을 만들어야 합니다.

새 시작 구성을 기존 Auto Scaling 그룹과 연결하려면 을 참조하십시오Auto Scaling 그룹에 대한 시작 구성 변경. 새 Auto Scaling 그룹을 만들려면 을 참조하십시오시작 구성을 사용하여 Auto Scaling 그룹 생성.

내용

- 시작 구성 생성
- 인스턴스 메타데이터 옵션 구성
- EC2 인스턴스를 사용하여 시작 구성 생성

시작 구성 생성

시작 구성 생성(콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 상단 내비게이션 바에서 AWS 지역을 선택합니다.
3. 왼쪽 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택합니다.
4. 페이지 상단에서 시작 구성을 선택합니다. 확인 메시지가 표시되면 시작 구성 보기를 선택하여 시작 구성 페이지를 볼 것인지 확인합니다.
5. Create launch configuration(시작 구성 생성)을 선택하고 시작 구성의 이름을 입력합니다.
6. Amazon Machine Image(AMI)에 AMI를 선택합니다. 특정 AMI를 찾으려면 AMI의 ID를 적어둔 다음 해당 ID를 검색 기준으로 입력하여 적합한 AMI를 찾을 수 있습니다.

   Amazon Linux 2 AMI의 ID를 얻으려면:

   a. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
   b. 왼쪽 탐색 창의 인스턴스에서 인스턴스를 선택한 후 인스턴스 시작을 선택합니다.
   c. Amazon Machine Image 선택 페이지의 Quick Start(빠른 시작) 탭에서 Amazon Linux 2 AMI(HVM) 옆에 있는 AMI의 ID를 기록해 둡니다.

7. 인스턴스 유형으로 인스턴스의 하드웨어 구성을 선택합니다.
8. Additional configuration(추가 구성)에서 다음 필드에 주의하세요.
   a. (옵션) Purchasing option(구매 옵션)의 경우, 온디맨드 가격이 최대 가격으로 제한된 스팟 가격으로 스팟 인스턴스를 요청하려면 Request Spot Instances(스팟 인스턴스 요청)를 선택할 수 있습니다. 경우에 따라 스팟 인스턴스에 인스턴스 시간당 최고 가격을 지정할 수 있습니다.

      > **Note**
      > 스팟 인스턴스는 애플리케이션이 실행되는 시간을 유연하게 조정할 수 있고 애플리케이션을 중단할 수 있는 경우, 선택할 수 있는 온디맨드 인스턴스에 비해 비용 효과적인 방법입니다. 자세한 설명은 내결함성 및 유연한 애플리케이션을 위한 스팟 인스턴스 요청 섹션을 참조하세요.

   b. (옵션) IAM instance profile(IAM 인스턴스 프로파일)로 인스턴스와 연결할 역할을 선택합니다. 자세한 정보는 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할을 참조하세요.
   c. (선택 사항) 모니터링의 경우 세부 모니터링을 활성화하여 인스턴스가 1분 간격으로 지표 데이터를 CloudWatch Amazon에 게시하도록 할지 여부를 선택합니다. 추가 요금이 발생합니다. 자세한 설명은 Auto Scaling 인스턴스에 대한 모니터링 구성 섹션을 참조하세요.
   d. (옵션) Advanced details(고급 세부 정보), User data(사용자 데이터)에서 시작 중 인스턴스를 구성하거나 인스턴스가 시작된 후 구성 스크립트를 실행할 때 사용할 사용자 데이터를 지정할 수 있습니다.
   e. (옵션) Advanced details(고급 세부 정보),IP address type(IP 주소 유형)에서 퍼블릭 IP 주소를 그룹의 인스턴스에 할당할지 선택합니다. 값을 설정하지 않은 경우, 기본적으로 인스턴스가 시작되는 서브넷의 자동 할당 퍼블릭 IP 설정을 사용합니다.
9. (옵션)Storage (volumes)(스토리지(볼륨))에서 추가 저장소가 필요하지 않으면 이 섹션을 건너뛸 수 있습니다. 그러지 않으면 AMI에서 지정한 볼륨 외에 인스턴스에 연결할 볼륨을 지정하기 위해 Add new volume(새 볼륨 추가)을 선택합니다. 그런 다음 원하는 옵션을 선택하고 Devices(디바이스),Snapshot(스냅샷), Size(크기), Volume type(볼륨 유형), IOPS, Throughput(처리량), Delete on termination(해지 시 삭제) 및 Encrypted에(암호화됨) 대한 관련 값을 선택합니다.
10. Security groups(보안 그룹)에서 그룹의 인스턴스와 연결할 보안 그룹을 생성하거나 선택합니다. Create a new security group(새 보안 그룹 생성) 옵션을 선택한 상태로 두면 Linux를 실행하는 Amazon EC2 인스턴스에 대해 기본 SSH 규칙이 구성됩니다. 기본 RDP 규칙은 Windows를 실행하는 Amazon EC2 인스턴스에 대해 구성되어 있습니다.

11. Key pair (login)(키 페어(로그인))로 Key pair options(키 페어 옵션) 아래에 있는 옵션을 선택합니다.

    Amazon EC2 인스턴스 키 페어를 이미 구성했다면 여기에서 선택할 수 있습니다.

    아직 Amazon EC2 인스턴스 키 페어가 없다면 Create a new key pair(새 키 페어 생성)을 선택하고 식별 가능한 이름을 지정합니다. Download key pair(키 페어 다운로드)를 선택하여 컴퓨터에 키 페어를 다운로드합니다.

    > **Important**
    > 인스턴스에 연결해야 하는 경우, Proceed without a key pair(키 페어 없이 계속)를 선택하지 마세요.

12. 승인 확인란을 선택한 다음 Create launch configuration(시작 구성 생성)을 선택합니다.

기존 시작 구성에서 시작 구성을 만들려면 (콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 상단 내비게이션 바에서 AWS 지역을 선택합니다.
3. 왼쪽 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택합니다.
4. 페이지 상단에서 시작 구성을 선택합니다. 확인 메시지가 표시되면 시작 구성 보기를 선택하여 시작 구성 페이지를 볼 것인지 확인합니다.
5. 시작 구성을 선택한 다음 Actions(작업), Copy launch configuration(시작 구성 복사)을 선택합니다. 이는 새 시작 구성을 원래 시작 구성과 동일한 옵션으로 설정하지만 이름에는 "Copy"를 추가합니다.
6. 시작 구성 복사 페이지에서 필요에 따라 구성 옵션을 편집하고 Create launch configuration(시작 구성 생성)을 선택합니다.

명령행을 사용하여 시작 구성을 만들려면

다음 명령 중 하나를 사용할 수 있습니다:

- create-launch-configuration (AWS CLI)
- New-AS () LaunchConfigurationAWS Tools for Windows PowerShell

인스턴스 메타데이터 옵션 구성

Amazon EC2 Auto Scaling은 시작 구성에서 인스턴스 메타데이터 서비스(IMDS) 구성을 지원합니다. 따라서 시작 구성을 사용하여 인스턴스 메타데이터를 요청하는 세션 지향 방법인 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 요구하도록 Auto Scaling 그룹의 Amazon EC2 인스턴스를 구성하는 옵션을 제공합니다. IMDSv2의 이점에 대한 자세한 설명은 EC2 인스턴스 메타데이터 서비스에 심층적인 방어 기능을 추가하기 위한 향상된 기능에 관한 AWS 블로그의 문서를 참조하세요.

IMDSv2 및 IMDSv1(기본값)을 둘 다 지원하거나 IMDSv2 사용을 요구하도록 IMDS를 구성할 수 있습니다. 하나 AWS CLI 또는 SDK 중 하나를 사용하여 IMDS를 구성하는 경우, IMDSv2를 사용하도록 요구하려면 최신 버전의 AWS CLI 또는 SDK를 사용해야 합니다.

다음에 대해 시작 구성을 구성할 수 있습니다.

- 인스턴스 메타데이터를 요청할 때 IMDSv2를 사용해야 하도록 설정
- PUT 응답 홉 제한 지정
- 인스턴스 메타데이터에 대한 액세스 비활성화

인스턴스 메타데이터 서비스 구성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 서비스 구성 주제를 참조하십시오.

시작 구성에서 IMDS 옵션을 구성하려면 다음 절차를 따르세요. 시작 구성을 생성했으면 Auto Scaling 그룹과 연결할 수 있습니다. 시작 구성을 기존 Auto Scaling 그룹과 연결하면 기존 시작 구성이 Auto Scaling 그룹에서 분리되고, 새 시작 구성에서 지정한 IMDS 옵션을 사용하기 위해 기존 인스턴스를 교체해야 합니다. 자세한 설명은 Auto Scaling 그룹에 대한 시작 구성 변경 섹션을 참조하세요.

시작 구성에서 IMDS를 구성하려면(콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 상단 탐색 표시줄에서 AWS 지역을 선택합니다.
3. 왼쪽 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택합니다.
4. 페이지 상단에서 시작 구성을 선택합니다. 확인 메시지가 표시되면 시작 구성 보기를 선택하여 시작 구성 페이지를 볼 것인지 확인합니다.
5. Create launch configuration(시작 구성 생성)을 선택하고 일반적인 방법으로 시작 구성을 생성합니다. Amazon Machine Image(AMI)의 ID, 인스턴스 유형 그리고 선택적으로 키 페어, 하나 이상의 보안 그룹, 인스턴스의 추가 EBS 볼륨 또는 인스턴스 스토어 볼륨을 포함하세요.

6. 이 시작 구성과 연결된 모든 인스턴스에 대해 인스턴스 메타데이터 옵션을 구성하려면 Additional configuration(추가 구성)의 Advanced details(고급 세부 정보)에서 다음을 수행하십시오:

   a. Metadata accessible(메타데이터 액세스 가능)에서 인스턴스 메타데이터 서비스의 HTTP 엔드포인트에 대한 액세스를 활성화할지 또는 비활성화할지를 선택합니다. 기본적으로 HTTP 엔드포인트는 활성화되어 있습니다. 엔드포인트를 비활성화하도록 선택하면 인스턴스 메타데이터에 대한 액세스가 해제됩니다. HTTP 엔드포인트가 활성화된 경우에만 IMDSv2를 요구하도록 조건을 지정할 수 있습니다.

   b. Metadata version(메타데이터 버전)에서 인스턴스 메타데이터를 요청하는 경우, 인스턴스 메타데이터 서비스 버전 2(IMDSv2)의 사용을 요구하도록 선택할 수 있습니다. 값을 지정하지 않으면 기본적으로 IMDSv1 및 IMDSv2를 둘 다 지원합니다.

   c. Metadata token response hop limit(메타데이터 토큰 응답 홉 제한)에서 인스턴스 메타데이터 토큰에 허용되는 네트워크 홉 수를 설정할 수 있습니다. 값을 지정하지 않으면 기본값은 1입니다.

7. 모두 마쳤으면 Create launch configuration(시작 구성 생성)을 선택합니다.

AWS CLI를 사용하여 시작 구성에서 IMDSv2 사용 요구

로 `--metadata-options` 설정된 상태에서 다음 create-launch-configuration명령을 사용합니다`HttpTokens=required`. 또한 `HttpTokens`의 값을 지정할 때 `HttpEndpoint`를 enabled로 설정해야 합니다. 메타데이터 검색 요청에 대해 보안 토큰 헤더가 required로 설정되어 있으므로 인스턴스 메타데이터를 요청할 때 인스턴스가 IMDSv2를 사용해야 합니다.

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imdsv2 \
  --image-id ami-01e24be29428c15b2 \
  --instance-type t2.micro \
      ...
  --metadata-options "HttpEndpoint=enabled,HttpTokens=required"
```

인스턴스 메타데이터에 대한 액세스를 끄려면

다음 create-launch-configuration명령을 사용하여 인스턴스 메타데이터에 대한 액세스를 끕니다. 나중에 modify-instance-metadata-options명령을 사용하여 액세스를 다시 활성화할 수 있습니다.

```
aws autoscaling create-launch-configuration \
  --launch-configuration-name my-lc-with-imds-disabled \
  --image-id ami-01e24be29428c15b2 \
```

```
  --instance-type t2.micro \
      ...
  --metadata-options "HttpEndpoint=disabled"
```

EC2 인스턴스를 사용하여 시작 구성 생성

실행 중인 EC2 인스턴스의 속성을 사용하여 시작 구성을 생성할 수도 있습니다.

처음부터 시작 구성을 새로 만드는 것과 기존 EC2 인스턴스에서 시작 구성을 만드는 것 사이에는 차이점이 있습니다. 처음부터 시작 구성을 새로 만드는 경우, 이미지 ID, 인스턴스 유형, 선택적 리소스(스토리지 디바이스 등) 및 옵션 설정(모니터링 등)을 지정합니다. 실행 중인 인스턴스에서 시작 구성을 생성할 경우, Amazon EC2 Auto Scaling은 지정된 인스턴스에서 시작 구성에 대한 속성을 가져옵니다. 또한 속성은 인스턴스가 시작된 AMI에 대한 블록 디바이스 매핑에서도 파생되며, 실행 후 추가된 추가 블록 디바이스는 무시합니다.

실행 중인 인스턴스를 사용하여 시작 구성을 만들 때 해당 속성을 동일한 요청의 일부로 지정하여 AMI, 블록 디바이스, 키 페어, 인스턴스 프로파일, 인스턴스 유형, 커널, 인스턴스 모니터링, 배치 테넌시, ramdisk, 보안 그룹, 스팟(최대) 가격, 사용자 데이터, 해당 인스턴스에 퍼블릭 IP 주소가 있는지, 해당 인스턴스가 EBS에 최적화되어 있는지 등의 속성을 재정의할 수 있습니다.

> **Note**
> 지정된 인스턴스에 현재 시작 구성에서 지원되지 않는 속성이 있는 경우, Auto Scaling 그룹에서 시작하는 인스턴스는 기존 EC2 인스턴스와 동일하지 않을 수도 있습니다.

> **Important**
> 지정된 인스턴스를 시작할 때 사용될 AMI가 늘 존재해야 합니다.

주제

- EC2 인스턴스에서 시작 구성 생성(AWS CLI)
- 인스턴스에서 시작 구성을 만들고 블록 디바이스 재정의(AWS CLI)
- 시작 구성 생성 및 인스턴스 유형 재정의(AWS CLI)

EC2 인스턴스에서 시작 구성 생성(AWS CLI)

다음 create-launch-configuration명령을 사용하여 인스턴스와 동일한 속성을 사용하는 인스턴스에서 시작 구성을 생성합니다. 시작 후 추가된 블록 디바이스는 무시됩니다.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance --instance-id i-a8e09d9c
```

다음 describe-launch-configurations명령을 사용하여 시작 구성을 설명하고 해당 속성이 인스턴스의 속성과 일치하는지 확인할 수 있습니다.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance
```

다음은 응답의 예입니다.

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-05355a6c",
            "CreatedTime": "2014-12-29T16:14:50.382Z",
            "BlockDeviceMappings": [],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8422d1eb"
            ],
            "LaunchConfigurationName": "my-lc-from-instance",
            "KernelId": "null",
            "RamdiskId": null,
            "InstanceType": "t1.micro",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

인스턴스에서 시작 구성을 만들고 블록 디바이스 재정의(AWS CLI)

기본적으로 Amazon EC2 Auto Scaling은 시작 구성을 만들기 위해 사용자가 지정한 EC2 인스턴스의 속성을 사용합니다. 하지만 블록 디바이스는 인스턴스가 아니라 인스턴스를 시작하는 데 사용된 AMI에서 파생됩니다. 시작 구성에 블록 디바이스를 추가하려면 시작 구성에서 블록 디바이스 매핑을 재정의해야 합니다.

다음 create-launch-configuration명령을 사용하여 EC2 인스턴스를 사용하지만 사용자 지정 블록 디바이스 매핑이 포함된 시작 구성을 만들 수 있습니다.

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-bdm --instance-id i-a8e09d9c \
  --block-device-mappings "[{\"DeviceName\":\"/dev/sda1\",\"Ebs\":{\"SnapshotId\":\"snap-3decf207\"}},{\"DeviceName\":\"/dev/sdf\",\"Ebs\":{\"SnapshotId\":\"snap-eed6ac86\"}}]"
```

다음 describe-launch-configurations명령을 사용하여 시작 구성을 설명하고 사용자 지정 블록 디바이스 매핑을 사용하는지 확인하십시오.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-bdm
```

다음 예 응답은 시작 구성에 대한 내용입니다.

```
{
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-c49c0dac",
            "CreatedTime": "2015-01-07T14:51:26.065Z",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/sda1",
                    "Ebs": {
                        "SnapshotId": "snap-3decf207"
                    }
                },
```

```
                {
                    "DeviceName": "/dev/sdf",
                    "Ebs": {
                        "SnapshotId": "snap-eed6ac86"
                    }
                }
            ],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8637d3e3"
            ],
            "LaunchConfigurationName": "my-lc-from-instance-bdm",
            "KernelId": null,
            "RamdiskId": null,
            "InstanceType": "t1.micro",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

시작 구성 생성 및 인스턴스 유형 재정의(AWS CLI)

기본적으로 Amazon EC2 Auto Scaling은 시작 구성을 만들기 위해 사용자가 지정한 EC2 인스턴스의 속성을 사용합니다. 요건에 따라 해당 인스턴스의 속성을 재정의하여 사용자가 요구하는 값을 사용하고자 할 수 있습니다. 예를 들면 사용자는 인스턴스 유형을 재정의할 수 있습니다.

다음 create-launch-configuration명령을 사용하여 EC2 인스턴스를 사용하지만 인스턴스 유형 (예:`t2.medium`) 이 인스턴스와는 다른 시작 구성을 생성합니다 (예:`t2.micro`).

```
aws autoscaling create-launch-configuration --launch-configuration-name my-lc-from-instance-changetype \
  --instance-id i-a8e09d9c --instance-type t2.medium
```

다음 describe-launch-configurations명령을 사용하여 시작 구성을 설명하고 인스턴스 유형이 재정의되었는지 확인합니다.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-lc-from-instance-changetype
```

다음 예 응답은 시작 구성에 대한 내용입니다.

```
{
```

```
    "LaunchConfigurations": [
        {
            "UserData": null,
            "EbsOptimized": false,
            "LaunchConfigurationARN": "arn",
            "InstanceMonitoring": {
                "Enabled": false
            },
            "ImageId": "ami-05355a6c",
            "CreatedTime": "2014-12-29T16:14:50.382Z",
            "BlockDeviceMappings": [],
            "KeyName": "my-key-pair",
            "SecurityGroups": [
                "sg-8422d1eb"
            ],
            "LaunchConfigurationName": "my-lc-from-instance-changetype",
            "KernelId": "null",
            "RamdiskId": null,
            "InstanceType": "t2.medium",
            "AssociatePublicIpAddress": true
        }
    ]
}
```

Auto Scaling 그룹에 대한 시작 구성 변경

> **Important**
>
> 시작 구성은 시작 구성에서 시작 템플릿으로 아직 마이그레이션하지 않은 고객을 위해 제공하고 있습니다. Auto Scaling 그룹을 위한 시작 템플릿 생성에 대한 자세한 설명은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작을 참조하세요.

이 주제에서는 다른 시작 구성을 Auto Scaling 그룹과 연결하는 방법을 설명합니다.

시작 구성을 변경하면 새 구성 옵션을 사용하여 새 인스턴스가 모두 시작되지만 기존 인스턴스는 영향을 받지 않습니다. 자세한 정보는 Auto Scaling 인스턴스 업데이트을 참조하세요.

Auto Scaling 그룹에 대한 시작 구성 변경(콘솔)

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

2. 왼쪽 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택합니다.
3. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
4. Details(세부 정보) 탭에서 Launch configuration(시작 구성), Edit(편집)을 선택합니다.
5. 시작 구성의 경우 시작 구성을 선택합니다.
6. 완료되면 Update(업데이트)를 선택합니다.

명령줄을 사용하여 Auto Scaling 그룹의 시작 구성을 변경하려면

다음 명령 중 하나를 사용할 수 있습니다:

- update-auto-scaling-group (AWS CLI)
- Update-as () AutoScalingGroupAWS Tools for Windows PowerShell

Auto Scaling 그룹

> **Note**
>
> Auto Scaling 그룹을 처음 사용하는 경우 첫 번째 Auto Scaling 그룹 생성 자습서의 단계를 진행하여 시작하고 그룹의 인스턴스가 종료될 때 Auto Scaling 그룹이 어떻게 응답하는지 확인하세요.

Auto Scaling 그룹에는 자동 조정 및 관리를 위해 논리적 그룹으로 취급되는 EC2 인스턴스 모음이 포함되어 있습니다. Auto Scaling 그룹을 사용하면 상태 확인 교체 및 조정 정책과 같은 Amazon EC2 Auto Scaling 기능을 사용할 수도 있습니다. Auto Scaling 그룹의 인스턴스 수를 유지하고 자동 조정을 수행하는 것은 모두 Amazon EC2 Auto Scaling 서비스의 핵심 기능입니다.

Auto Scaling 그룹의 크기는 사용자가 원하는 용량으로 설정한 인스턴스 수에 따라 달라집니다. 수동으로 또는 자동 크기 조정을 사용하여 수요에 맞게 크기를 조정할 수 있습니다.

Auto Scaling 그룹은 원하는 용량을 충족하도록 충분한 인스턴스를 시작하여 시작합니다. 그룹 내 인스턴스에 대한 주기적인 건전성 체크를 수행하여 이 인스턴스 수를 유지합니다. Auto Scaling 그룹은 인스턴스 상태가 이상이 있는 경우에도 고정된 수의 인스턴스를 계속 유지합니다. 인스턴스가 비건전 상태가 되면 그룹에서는 비건전 인스턴스를 해지하고 이를 교체할 다른 인스턴스를 시작합니다. 자세한 설명은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 섹션을 참조하세요.

조정 정책을 사용하여 바뀌는 조건을 충족하도록 그룹의 인스턴스 수를 동적으로 늘리거나 줄일 수 있습니다. 조정 정책의 효력이 발생되면, Auto Scaling 그룹이 해당 그룹의 희망 용량을 사용자가 지정하는 최소 및 최대 용량 값 사이에서 조절하고 필요에 따라 인스턴스를 시작 또는 해지합니다. 일정에서도 스케일 아웃이 가능합니다. 자세한 설명은 스케일링 방법 선택 섹션을 참조하세요.

Auto Scaling 그룹을 생성할 때 온디맨드 인스턴스를 띄울지, 스팟 인스턴스 또는 둘 모두를 띄울지를 선택할 수 있습니다. 를 사용하는 경우에만 Auto Scaling 그룹에 여러 구매 옵션을 지정할 수 있습니다.launch template. 자세한 내용은 섹션을 참조하세요여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹.

스팟 인스턴스를 사용하면 온디맨드 요금에 비해 엄청나게 할인된 가격으로 미사용 EC2 용량에 액세스할 수 있습니다. 자세한 내용은 Amazon EC2 스팟 인스턴스 를 참조하세요. 스팟 인스턴스와 온 디맨드 인스턴스 간에는 다음과 같은 주요 차이점이 있습니다.

- 스팟 인스턴스의 가격은 수요에 따라 변동합니다.

- AmazonEC2은 스팟 인스턴스의 가용성 또는 가격이 변경되면 개별 스팟 인스턴스를 종료할 수 있습니다.

스팟 인스턴스가 해지되면 Auto Scaling 그룹에서는 교체 인스턴스를 시작하여 그룹의 원하는 용량을 유지하려고 합니다.

인스턴스가 시작되고 사용자가 여러 가용 영역을 지정한 경우, 희망 용량이 이러한 가용 영역에 분산됩니다. 크기 조정 작업이 발생하면 Amazon EC2 Auto Scaling은 지정한 모든 가용 영역에 걸쳐 균형을 자동으로 유지합니다.

내용

- 시작 템플릿을 사용하여 Auto Scaling 그룹 생성
- 시작 구성을 사용하여 Auto Scaling 그룹 생성
- Auto Scaling 그룹 업데이트
- Auto Scaling 그룹 및 인스턴스에 태그 지정
- 인스턴스 유지 관리 정책
- Amazon EC2 Auto Scaling 수명 주기 후크
- 웜 풀을 사용하여 부팅 시간이 긴 애플리케이션의 지연 시간을 줄입니다.
- Auto Scaling 그룹에서 인스턴스 분리 또는 연결
- Auto Scaling 그룹에서 일시적으로 인스턴스 제거
- Auto Scaling 인프라 삭제

시작 템플릿을 사용하여 Auto Scaling 그룹 생성

시작 템플릿을 생성한 경우 시작 템플릿을 EC2 인스턴스의 구성 템플릿으로 사용하는 Auto Scaling 그룹을 생성할 수 있습니다. 시작 템플릿은 인스턴스의 AMI ID, 인스턴스 유형, 키 페어, 보안 그룹 및 블록 디바이스 매핑과 같은 정보를 지정합니다. 시작 템플릿 생성에 대한 자세한 설명은 Auto Scaling 그룹에 대한 시작 템플릿 생성 섹션을 참조하십시오.

Auto Scaling 그룹을 만들려면 충분한 권한이 있어야 합니다. 또한 Amazon EC2 Auto Scaling이 아직 존재하지 않는 경우 사용자를 대신하여 작업을 수행하는 데 사용하는 서비스 연결 역할을 생성할 수 있는 충분한 권한이 있어야 합니다. 관리자가 권한을 부여하기 위한 참조로 사용할 수 있는 IAM 정책의 예는 자격 증명 기반 정책 예시 및 섹션을 참조하세요Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다..

내용

- 시작 템플릿을 사용하여 Auto Scaling 그룹 생성
- Amazon EC2 시작 마법사를 사용하여 Auto Scaling 그룹 생성
- 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹

시작 템플릿을 사용하여 Auto Scaling 그룹 생성

Auto Scaling 그룹을 생성할 때는 Amazon EC2 인스턴스, 인스턴스의 가용 영역 및 VPC 서브넷, 원하는 용량, 최소 및 최대 용량 제한을 구성하는 데 필요한 정보를 지정해야 합니다.

Auto Scaling 그룹에서 시작하는 Amazon EC2 인스턴스를 구성하려면 시작 템플릿 또는 시작 구성을 지정하면 됩니다. 다음 절차는 시작 템플릿을 사용하여 Auto Scaling 그룹을 생성하는 방법을 보여줍니다.

사전 조건

- 시작 템플릿을 생성해야 합니다. 자세한 설명은 Auto Scaling 그룹에 대한 시작 템플릿 생성 섹션을 참조하세요.

시작 템플릿을 사용하여 Auto Scaling 그룹을 생성하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 화면 상단의 탐색 표시줄에서 시작 템플릿을 생성할 때 사용한 것과 동일한 AWS 리전 것을 선택합니다.
3. Create an Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.
4. 시작 템플릿 또는 구성 선택 페이지에서 다음을 수행하십시오:
   a. Auto Scaling group name(Auto Scaling 그룹 명칭)에 Auto Scaling 그룹 명칭을 입력합니다.
   b. 시작 템플릿에서 기존 시작 템플릿을 선택합니다.
   c. Launch template version(시작 템플릿 버전)에서 Auto Scaling 그룹이 스케일 아웃 시 시작 템플릿의 기본 버전을 사용할지, 최신 버전을 사용할지, 아니면 특정 버전을 사용할지를 선택합니다.
   d. 시작 템플릿이 사용하려는 모든 옵션을 지원하는지 확인한 후 다음을 선택합니다.
5. 인스턴스 시작 옵션 선택 페이지에서 여러 인스턴스 유형을 사용하지 않는 경우 인스턴스 유형 요구 사항 섹션을 건너뛰고 시작 템플릿에 지정된 EC2 인스턴스 유형을 사용할 수 있습니다.

여러 인스턴스 유형을 사용하려면 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹을 참조하십시오.

6. 네트워크에서 원하는 항목을 VPC선택합니다. VPC Auto Scaling 그룹은 시작 템플릿에서 지정한 보안 VPC 그룹과 동일하게 생성해야 합니다.

7. 가용 영역 및 서브넷의 경우 지정된 서브넷에서 하나 이상의 서브넷을 선택합니다. VPC 여러 가용 영역의 서브넷을 사용하여 가용성을 높일 수 있습니다. 자세한 설명은 VPC 서브넷 선택 시 고려 사항 섹션을 참조하세요.

8. 지정된 인스턴스 유형으로 시작 템플릿을 생성한 경우, 다음 단계를 계속 진행하여 시작 템플릿의 인스턴스 유형을 사용하는 Auto Scaling 그룹을 생성할 수 있습니다.

   또는 시작 템플릿에 인스턴스 유형이 지정되지 않았거나 자동 크기 조정에 여러 인스턴스 유형을 사용하려는 경우, 시작 템플릿 재정의(Override launch template) 옵션을 선택할 수 있습니다. 자세한 설명은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.

9. 다음(Next)을 선택하여 다음 단계로 계속 진행합니다.

   또는 나머지 기본값을 그대로 두고, 검토로 이동(Skip to review)을 선택할 수 있습니다.

10. (옵션) 고급 옵션 구성(Configure advanced options) 페이지에서 다음 옵션을 구성하고 다음(Next)을 선택합니다.

    a. (선택 사항) 상태 점검, 추가 상태 확인 유형의 경우 Amazon EBS 상태 확인 활성화를 선택합니다. 자세한 내용은 상태 확인을 사용하여 Amazon EBS 볼륨이 손상된 Auto Scaling 인스턴스를 모니터링합니다. 단원을 참조하십시오.

    b. (옵션) 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. 이 시간은 Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.

    c. 추가 설정, 모니터링에서 CloudWatch 그룹 지표 수집을 활성화할지 여부를 선택합니다. 이러한 지표는 해지 인스턴스 수 또는 보류 중인 인스턴스 수와 같은 잠재적 문제의 지표가 될 수 있는 측정값을 제공합니다. 자세한 내용은 Auto Scaling 그룹 및 인스턴스의 CloudWatch 메트릭을 모니터링합니다. 단원을 참조하십시오.

    d. 기본 인스턴스 웜업 활성화에서 이 옵션을 선택하고 애플리케이션의 웜업 시간을 선택합니다. 조정 정책이 있는 Auto Scaling 그룹을 생성하는 경우 기본 인스턴스 웜업 기능은 동적 조정에 사용되는 Amazon CloudWatch 지표를 개선합니다. 자세한 내용은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 단원을 참조하십시오.

11. (옵션) 그룹 크기 및 조정 정책 구성 페이지에서 다음 옵션을 구성하고 다음을 선택합니다.

a. 그룹 사이즈에서 원하는 용량에 시작시킬 초기 인스턴스 수를 입력합니다.

b. 스케일링 섹션의 스케일링 제한에서 희망 용량에 대한 새 값이 최소 희망 용량 및 최대 희망 용량보다 큰 경우, 최대 희망 용량은 자동으로 새 희망 용량 값으로 증가합니다. 필요에 따라 이러한 한도를 변경할 수 있습니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.

c. 자동 조정의 경우, 대상 추적 조정 정책을 생성할지 여부를 선택합니다. Auto Scaling 그룹을 생성한 후에 이 정책을 생성할 수도 있습니다.

   대상 추적 조정 정책을 선택하는 경우, 대상 추적 조정 정책 생성의 지침에 따라 정책을 생성하십시오.

d. 인스턴스 유지 관리 정책에서는 인스턴스 유지 관리 정책을 만들지 여부를 선택합니다. Auto Scaling 그룹을 생성한 후에 이 정책을 생성할 수도 있습니다. 인스턴스 유지 관리 정책 설정의 지침에 따라 정책을 만듭니다.

e. 인스턴스 스케일 인 보호(Instance scale-in protection)에서 인스턴스 스케일 인 보호를 활성화할지를 선택합니다. 자세한 설명은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 섹션을 참조하세요.

12. (옵션) 알림을 받으려면 알림 추가(Add notification)에 알림을 구성하고 다음(Next)을 선택합니다. 자세한 설명은 아마존 EC2 오토 스케일링에 대한 아마존 SNS 알림 옵션 섹션을 참조하세요.
13. (옵션) 태그를 추가하려면 태그 추가(Add tag)를 선택하고 각 태그에 태그 키와 값을 제공한 후 다음(Next)을 선택합니다. 자세한 설명은 Auto Scaling 그룹 및 인스턴스에 태그 지정 섹션을 참조하세요.
14. 검토(Review) 페이지에서 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.

명령행을 사용하여 Auto Scaling 그룹을 생성하려면

다음 명령 중 하나를 사용할 수 있습니다:

- create-auto-scaling-group (AWS CLI)
- New- ASAutoScalingGroup ()AWS Tools for Windows PowerShell

Amazon EC2 시작 마법사를 사용하여 Auto Scaling 그룹 생성

다음 절차는 Amazon EC2 콘솔에서 인스턴스 출범(Launch instance) 마법사를 사용하여 Auto Scaling 그룹을 생성하는 방법을 보여줍니다. 이 옵션은 인스턴스 출범(Launch instance) 마법사의 특정 구성 세부 정보로 출범 템플릿을 자동으로 채웁니다.

> **Note**
>
> 마법사는 지정한 인스턴스 수로 Auto Scaling 그룹을 채우지 않으며, Amazon Machine Image(AMI) 및 인스턴스 타입으로 출범 템플릿만 채웁니다. Auto Scaling 그룹 생성(Create Auto Scaling group) 마법사를 사용하여 시작할 인스턴스 수를 지정합니다.
> AMI는 인스턴스를 구성하는 데 필요한 정보를 제공합니다. 동일한 구성의 인스턴스가 여러 개 필요할 때는 한 AMI에서 여러 인스턴스를 출범할 수 있습니다. Auto Scaling 그룹에 속한 인스턴스를 재부팅할 경우, 인스턴스가 해지되지 않도록 이미 애플리케이션이 설치된 사용자 정의 AMI를 사용하는 것이 좋습니다. Amazon EC2 Auto Scaling에서 사용자 정의 AMI를 사용하려면 먼저 맞춤 인스턴스에서 AMI를 생성한 다음 AMI를 사용하여 Auto Scaling 그룹에 대한 출범 템플릿을 생성해야 합니다.

필수 조건

- Auto Scaling 그룹을 생성하려는 AWS 리전 곳과 동일한 곳에 사용자 지정 AMI를 생성해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 AMI 생성을 참조하십시오.

맞춤 AMI를 템플릿으로 사용

이 섹션에서는 Amazon EC2 시작 마법사를 사용하여 맞춤 AMI로 출범 템플릿을 자동으로 채웁니다. 또는 출범 템플릿을 처음부터 설정하거나 출범 템플릿에 대해 구성할 수 있는 파라미터에 대한 자세한 설명을 보려면 시작 템플릿 생성(콘솔)을(를) 참조하세요.

사용자 정의 AMI를 템플릿으로 사용

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 화면 상단의 탐색 표시줄에 AWS 리전 현재가 표시됩니다. Auto Scaling 그룹을 시작할 지역을 선택합니다.
3. 탐색 창에서 인스턴스를 선택합니다.
4. 인스턴스 출범(Launch instance)을 선택하고 다음을 수행하십시오:
   a. 이름 및 태그(Name and tags)에서 이름(Name)을 비워 둡니다. 이름은 출범 템플릿을 생성하는 데 사용되는 데이터의 일부가 아닙니다.
   b. 애플리케이션 및 OS 이미지(Amazon Machine Image)(Application and OS Images (Amazon Machine Image))에서 더 많은 AMI 찾아보기(Browse more AMIs)를 선택하여 전체 AMI 카탈로그를 찾아봅니다.

c. 내 AMI(My AMIs)를 선택하고 앞서 생성한 AMI를 찾은 다음 선택(Select)을 선택합니다.

d. 인스턴스 타입(Instance type)에서 인스턴스 타입을 선택합니다.

> **Note**
>
> AMI를 생성할 때 사용한 것과 동일한 인스턴스 타입 또는 더 강력한 인스턴스 타입을 선택합니다.

e. 화면 오른쪽의 요약(Summary)에서 인스턴스 수(Number of instances)에 원하는 숫자를 입력합니다. 여기에 입력하는 숫자는 중요하지 않습니다. 시작할 인스턴스 수는 Auto Scaling 그룹을 생성할 때 지정합니다.

   인스턴스 수(Number of instances) 필드 아래에 2개 이상의 인스턴스를 출범할 때 EC2 Auto Scaling 고려(When launching more than 1 instance, consider EC2 Auto Scaling)라는 메시지가 표시됩니다.

f. EC2 Auto Scaling 고려(consider EC2 Auto Scaling) 하이퍼링크 텍스트를 선택합니다.

g. Auto Scaling 그룹 시작(Launch into Auto Scaling Group) 확인 대화 상자에서 계속(Continue)을 선택하여 시작 인스턴스 마법사에서 선택한 AMI 및 인스턴스 타입이 이미 채워진 출범 템플릿 생성(Create launch template) 페이지로 이동합니다.

계속(Continue)을 선택하면 출범 템플릿 생성(Create launch template) 페이지가 열립니다. 다음 절차에 따라 출범 템플릿 생성을 완료하십시오.

출범 템플릿 생성

1. 출범 템플릿 이름 및 설명(Launch template name and description)에서 새 출범 템플릿의 이름과 설명을 입력합니다.
2. (옵션) 키 페어(로그인)(Key pair (login)) 아래의 키 페어 이름(Key pair name)에서 인스턴스에 연결할 때 사용할 이전에 생성한 키 페어의 이름을 선택합니다(예: SSH 사용).
3. (옵션) 네트워크 설정(Network settings) 아래의 보안 그룹(Security groups)에서 이전에 생성한 보안 그룹을 하나 이상 선택합니다.
4. (옵션) 스토리지 구성(Configure storage)에서 스토리지 구성을 업데이트합니다. AMI와 인스턴스 타입에 따라 기본 스토리지 구성이 결정됩니다.
5. 출범 템플릿 구성이 끝나면 출범 템플릿 생성(Create launch template)을 선택합니다.
6. 확인 페이지에서 Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

Auto Scaling 그룹 생성

> **Note**
>
> 이 주제의 나머지 부분에서는 Auto Scaling 그룹을 생성하는 기본 절차를 설명합니다. Auto Scaling 그룹에 대해 구성할 수 있는 파라미터에 대한 자세한 설명은 시작 템플릿을 사용하여 Auto Scaling 그룹 생성 섹션을 참조하세요.

Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택하면 Auto Scaling 그룹(Auto Scaling group) 마법사가 열립니다. 다음 절차에 따라 Auto Scaling 그룹을 생성합니다.

Auto Scaling 그룹 생성

1. 출범 템플릿 또는 구성 선택 페이지에서 Auto Scaling 그룹의 이름을 입력합니다.
2. 생성한 출범 템플릿이 이미 선택되어 있습니다.

   Launch template version(출범 템플릿 버전)에서 Auto Scaling 그룹이 스케일 아웃 시 출범 템플릿의 기본 버전을 사용할지, 최신 버전을 사용할지, 아니면 특정 버전을 사용할지를 선택합니다.
3. 다음(Next)을 선택하여 다음 단계로 계속 진행합니다.
4. 인스턴스 출범 옵션 선택 페이지에서 여러 인스턴스 타입을 사용하지 않는 경우, 인스턴스 타입 요건 섹션을 건너뛰고 출범 템플릿에 지정된 EC2 인스턴스 타입을 사용할 수 있습니다.

   여러 인스턴스 타입을 사용하려면 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹을 참조하십시오.
5. 네트워크(Network)의 VPC에서 VPC를 선택합니다. Auto Scaling 그룹은 출범 템플릿에서 지정한 보안 그룹과 동일한 VPC에 생성되어야 합니다.

   > **Tip**
   >
   > 출범 템플릿에 보안 그룹을 지정하지 않은 경우, 지정한 VPC의 기본 보안 그룹으로 인스턴스가 시작됩니다. 기본적으로 이 보안 그룹은 외부 네트워크로부터의 인바운드 트래픽을 허용하지 않습니다.
6. 가용 영역 및 서브넷(Availability Zones and subnets)에서 지정한 VPC에 있는 서브넷 하나 이상을 선택합니다.
7. 다음(Next)을 두 번 선택하여 그룹 크기 및 조정 정책 구성(Configure group size and scaling policies) 페이지로 이동합니다.

8. 그룹 크기에서 원하는 용량(Auto Scaling 그룹이 생성된 후 즉시 실행할 초기 인스턴스 수)을 정의합니다.
9. 스케일링 섹션의 스케일링 제한에서 희망 용량에 대한 새 값이 최소 희망 용량 및 최대 희망 용량보다 큰 경우, 최대 희망 용량은 자동으로 새 희망 용량 값으로 증가합니다. 필요에 따라 이러한 한도를 변경할 수 있습니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.
10. Skip to review(검토로 건너뛰기)를 선택합니다.
11. 검토(Review) 페이지에서 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.

다음 단계

Auto Scaling 그룹이 제대로 생성되었는지는 활동 기록을 통해 확인할 수 있습니다. 활동(Activity) 탭의 활동 기록(Activity history)에 있는 상태(Status) 열에 Auto Scaling 그룹이 성공적으로 인스턴스를 출범했는지가 표시됩니다. 인스턴스가 시작되지 않거나 시작되었지만 즉시 해지되는 경우, 가능한 원인 및 해결 방법은 다음 주제를 참조하세요.

- Amazon EC2 Auto Scaling 문제 해결: EC2 인스턴스 시작 실패
- Amazon EC2 Auto Scaling 문제 해결: AMI 문제
- Amazon EC2 Auto Scaling에서 비정상 인스턴스 문제 해결

이제 원하는 경우, Auto Scaling 그룹과 동일한 지역에서 로드 밸런서를 연결할 수 있습니다. 자세한 설명은 Elastic Load Balancing을 사용하여 Auto Scaling 그룹에서 들어오는 애플리케이션 트래픽을 분산하십시오. 섹션을 참조하세요.

여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹

단일 Auto Scaling 그룹 내에서 온디맨드 인스턴스 및 스팟 인스턴스 플릿를 자동으로 확장할 수 있습니다. 스팟 인스턴스 사용에 대한 할인뿐만 아니라 예약 인스턴스 또는 Savings Plan을 사용하여 일반 온디맨드 인스턴스 요금에서도 할인을 받을 수 있습니다. 이러한 요소를 활용하여 EC2 인스턴스의 비용 절감을 최적화하고 애플리케이션에서 원하는 규모 및 성능을 얻을 수 있습니다.

스팟 인스턴스는 EC2 온디맨드 요금과 비교하여 대폭 할인된 가격으로 사용할 수 있는 예비 용량입니다. 스팟 인스턴스는 애플리케이션이 실행되는 시간을 유연하게 조정할 수 있고 애플리케이션을 중단할 수 있는 경우에 선택하는 비용 효율적인 방법입니다. 내결함성이 있고 유연한 다양한 애플리케이션에 사용할 수 있습니다. 스테이트리스 웹 서버, API 엔드포인트, 빅데이터 및 분석 애플리케이션, 컨테이너식 워크로드, CI/CD 파이프라인, 고성능 및 고처리량 컴퓨팅 (HPC/HTC), 렌더링 워크로드, 기타 유연한 워크로드 등을 예로 들 수 있습니다.

자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 구매 옵션을 참조하십시오.

주제

- 혼합 인스턴스 그룹 생성을 위한 설정 개요
- 여러 인스턴스 유형에 대한 할당 전략
- 속성 기반 인스턴스 유형 선택을 사용하여 혼합 인스턴스 그룹 생성
- 인스턴스 유형을 수동으로 선택하여 혼합 인스턴스 그룹 생성
- 인스턴스 가중치를 사용하도록 Auto Scaling 그룹을 구성합니다.
- 인스턴스 유형에 대해 서로 다른 시작 템플릿 사용

혼합 인스턴스 그룹 생성을 위한 설정 개요

이 항목에서는 Auto Scaling 혼합 인스턴스 그룹을 생성하는 방법에 대한 개요와 모범 사례를 제공합니다.

내용

- 개요
- 인스턴스 유형 유연성
- 가용 영역 유연성
- 스팟 최고가
- 사전 대응 용량 재분배
- 크기 조정 동작
- 인스턴스 유형의 지역별 가용성
- 관련 리소스
- 제한 사항

개요

혼합 인스턴스 그룹 생성을 위한 두 가지 옵션은 다음과 같습니다.

- 속성 기반 인스턴스 유형 선택 - 특정 인스턴스 속성에 따라 인스턴스 유형을 자동으로 선택하도록 컴퓨팅 요구 사항을 정의합니다.
- 수동 인스턴스 유형 선택 - 워크로드에 적합한 인스턴스 유형을 수동으로 선택합니다.

Manual selection

다음 단계에서는 인스턴스 유형을 수동으로 선택하여 혼합 인스턴스 그룹을 생성하는 방법을 설명합니다.

1. EC2 인스턴스를 시작하기 위한 파라미터가 포함된 시작 템플릿을 선택합니다. 시작 템플릿의 파라미터는 선택 사항이지만 Amazon Machine Image(AMI) ID가 시작 템플릿에 없는 경우 Amazon EC2 Auto Scaling에서 인스턴스를 시작할 수 없습니다.
2. 시작 템플릿을 재정의하는 옵션을 선택합니다.
3. 워크로드에 적합한 인스턴스 유형을 수동으로 선택합니다.
4. 시작할 온디맨드 인스턴스 및 스팟 인스턴스의 비율을 지정합니다.
5. Amazon EC2 Auto Scaling이 가능한 인스턴스 유형에서 온디맨드 및 스팟 용량을 충족하는 방법을 결정하는 할당 전략을 선택하세요.
6. 인스턴스를 시작할 가용 영역 및 VPC 서브넷을 선택합니다.
7. 그룹의 초기 크기(원하는 용량)와 그룹의 최소 및 최대 크기를 지정합니다.

시작 템플릿에 선언된 인스턴스 유형을 재정의하고 Auto Scaling 그룹의 자체 리소스 정의에 포함된 여러 인스턴스 유형을 사용하려면 재정의가 필요합니다. 사용 가능한 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하십시오.

또한 각 인스턴스 유형에 대해 다음과 같은 선택적 파라미터를 구성할 수 있습니다.

- `LaunchTemplateSpecification`— 필요에 따라 인스턴스 유형에 다른 시작 템플릿을 할당할 수 있습니다. 현재 콘솔에서는 이 옵션을 사용할 수 없습니다. 자세한 정보는 인스턴스 유형에 대해 서로 다른 시작 템플릿 사용을 참조하세요.
- `WeightedCapacity`— 원하는 용량에 포함되는 인스턴스의 양을 그룹 내 나머지 인스턴스와 비교하여 결정합니다. 하나의 인스턴스 유형에 대해 `WeightedCapacity` 값을 지정하는 경우 모든 인스턴스 유형에 대해 `WeightedCapacity` 값을 지정해야 합니다. 기본적으로 각 인스턴스는 원하는 용량에 대해 1개로 계산됩니다. 자세한 내용은 인스턴스 가중치를 사용하도록 Auto Scaling 그룹을 구성합니다.(을)를 참조하세요.

Attribute-based selection

Amazon EC2 Auto Scaling에서 특정 인스턴스 속성에 따라 인스턴스 유형을 자동으로 선택하도록 하려면 다음 단계를 사용하여 컴퓨팅 요구 사항을 지정하고 혼합 인스턴스 그룹을 생성하세요.

1. EC2 인스턴스를 시작하기 위한 파라미터가 포함된 시작 템플릿을 선택합니다. 시작 템플릿의 파라미터는 선택 사항이지만 Amazon Machine Image(AMI) ID가 시작 템플릿에 없는 경우 Amazon EC2 Auto Scaling에서 인스턴스를 시작할 수 없습니다.
2. 시작 템플릿을 재정의하는 옵션을 선택합니다.
3. vCPU 및 메모리 요구 사항 등의 컴퓨팅 요구 사항과 일치하는 인스턴스 속성을 지정합니다.
4. 시작할 온디맨드 인스턴스 및 스팟 인스턴스의 비율을 지정합니다.
5. Amazon EC2 Auto Scaling이 가능한 인스턴스 유형에서 온디맨드 및 스팟 용량을 충족하는 방법을 결정하는 할당 전략을 선택하세요.
6. 인스턴스를 시작할 가용 영역 및 VPC 서브넷을 선택합니다.
7. 그룹의 초기 크기(원하는 용량)와 그룹의 최소 및 최대 크기를 지정합니다.

시작 템플릿에 선언된 인스턴스 유형을 재정의하고 컴퓨팅 요구 사항을 설명하는 인스턴스 속성 세트를 사용하려면 재정의가 필요합니다. 지원되는 속성은 Amazon EC2 Auto Scaling API 참조를 참조하십시오 InstanceRequirements. 또는 인스턴스 속성 정의가 이미 있는 시작 템플릿을 사용할 수 있습니다.

필요에 따라 인스턴스 요구 사항 세트에 다른 시작 템플릿을 할당하도록 재정의 구조 내에서 `LaunchTemplateSpecification` 파라미터를 구성할 수도 있습니다. 현재 콘솔에서는 이 옵션을 사용할 수 없습니다. 자세한 내용은 Amazon EC2 Auto Scaling API 레퍼런스를 참조하십시오 LaunchTemplateOverrides.

기본적으로 인스턴트 수를 Auto Scaling 그룹의 원하는 용량으로 설정합니다.

또는 원하는 용량 값을 vCPU 수 또는 메모리 양으로 설정할 수 있습니다. 이렇게 하려면, `CreateAutoScalingGroup` API 작업의 `DesiredCapacityType` 속성을 사용하거나 AWS Management Console의 원하는 용량 유형 드롭다운 필드를 사용합니다. 이는 인스턴스 가중치에 대한 유용한 대안입니다.

인스턴스 유형 유연성

가용성을 높이려면 애플리케이션을 여러 인스턴스 유형에 배포합니다. 용량 요구 사항을 충족하려면 여러 인스턴스 유형을 사용하는 것이 가장 좋습니다. 이러한 방식으로 Amazon EC2 Auto Scaling은 선택한 가용 영역에 인스턴스 용량이 부족한 경우 다른 인스턴스 유형을 시작할 수 있습니다.

스팟 인스턴스에 인스턴스 용량이 부족할 경우 Amazon EC2 Auto Scaling은 다른 스팟 인스턴스 풀에서 시작하려고 계속 시도합니다. (사용하는 풀은 선택한 인스턴스 유형 및 할당 전략에 따라 결정됩니

다.) Amazon EC2 Auto Scaling을 사용하면 온디맨드 인스턴스 대신 스팟 인스턴스를 시작하여 스팟 인스턴스의 비용 절감 효과를 활용할 수 있습니다.

각 워크로드에 대해 최소 10개의 인스턴스 유형을 유연하게 선택할 것을 권장합니다. 인스턴스 유형을 선택할 때 가장 많이 사용되는 새 인스턴스 유형으로 사용자 자신을 제한하지 마세요. 이전 세대의 인스턴스 유형을 선택하면 온디맨드 고객의 수요가 적기 때문에 스팟 중단이 적은 경향이 있습니다.

가용 영역 유연성

Auto Scaling 그룹을 여러 가용 영역으로 확장하는 것이 좋습니다. 여러 가용 영역을 사용하면 복원력을 높이기 위해 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션을 설계할 수 있습니다.

추가 이점으로, 단일 가용 영역의 그룹과 비교할 때 더 깊은 Amazon EC2 용량 풀에 액세스할 수 있습니다. 용량은 각 가용 영역에서 인스턴스 유형과 관계없이 변동되므로, 인스턴스 유형과 가용 영역이 둘 다 유연한 경우 대부분 더 많은 컴퓨팅 용량을 확보할 수 있습니다.

다중 가용 영역 사용에 대한 자세한 내용은 예: 가용 영역 전반에 인스턴스 분산(을)를 참조하세요.

스팟 최고가

AWS CLI 또는 SDK를 사용하여 Auto Scaling 그룹을 생성할 때 `SpotMaxPrice` 파라미터를 지정할 수 있습니다. `SpotMaxPrice` 파라미터는 스팟 인스턴스 시간당 지불하려는 최고 가격을 결정합니다.

오버라이드(그룹 수준의 `"DesiredCapacityType": "vcpu"` 또는 `"DesiredCapacityType": "memory-mib"`)에서 `WeightedCapacity` 파라미터를 지정하는 경우 최고 가격은 전체 인스턴스의 최고 가격이 아니라 최고 단가를 나타냅니다.

최고 가격을 지정하지 않는 것이 매우 좋습니다. 스팟 인스턴스를 받지 못하면(예: 최고가가 너무 낮은 경우) 애플리케이션이 실행되지 않을 수 있습니다. 최고 가격을 지정하지 않는 경우, 기본 최고 가격은 온디맨드 가격입니다. 시작하는 스팟 인스턴스에 대해 스팟 가격만 지불합니다. 스팟 인스턴스에서 제공하는 대폭 할인 금액은 여전히 적용됩니다. 이러한 할인은 스팟 요금 모델을 사용해 실현된 안정적인 스팟 요금 덕분에 가능해진 것입니다. 자세한 내용은 Amazon EC2 사용 설명서의 요금 및 절감을 참조하십시오.

사전 대응 용량 재분배

사용 사례에서 허용하는 경우, 용량 재분배를 권장합니다. 용량 재분배를 사용하면 실행 중인 스팟 인스턴스에 스팟 인스턴스 중단 2분 전 공지가 수신되기 전에 미리 새 스팟 인스턴스로 플릿을 보강할 수 있으므로 워크로드 가용성을 유지하는 데 도움이 됩니다.

용량 재분배를 활성화하면 Amazon EC2 Auto Scaling이 재분배 권장 사항을 받은 스팟 인스턴스를 사전에 교체하려고 합니다. 이는 중단 위험이 높지 않은 새로운 스팟 인스턴스로 워크로드를 재분배할 수 있는 기회를 제공합니다.

자세한 내용은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리(을)를 참조하세요.

크기 조정 동작

혼합 인스턴스 그룹을 생성하면 기본적으로 온디맨드 인스턴스가 사용됩니다. 스팟 인스턴스를 사용하려면 온디맨드 인스턴스로 시작할 그룹의 비율을 수정해야 합니다. 온디맨드 백분율에 0에서 100까지의 숫자를 지정할 수 있습니다.

선택 사항으로, 시작할 온디맨드 인스턴스의 기본 수를 지정할 수도 있습니다. 이러한 경우 Amazon EC2 Auto Scaling은 그룹 스케일 아웃 시 온디맨드 인스턴스의 기본 용량을 시작한 후에만 스팟 인스턴스를 시작합니다. 기본 용량을 초과하는 용량은 온디맨드 백분율을 사용하여 시작할 온디맨드 인스턴스 및 스팟 인스턴스 수를 결정합니다.

Amazon EC2 Auto Scaling은 백분율을 상응하는 인스턴스 수로 변환합니다. 변환 결과에 소수가 포함되면 온디맨드 인스턴스를 위해 다음 정수로 반올림합니다.

이 수가 증가하거나 감소할 경우 Auto Scaling 그룹은 다음 표와 같이 동작합니다.

예: 확장 동작

구매 옵션	구매 옵션 전체에서 실행되는 인스턴스 그룹의 크기와 수			
	10	20	30	40
예 1: 기본 10, 50/50% 온디맨드/스팟				
온디맨드 인스턴스 (기본 금액)	10	10	10	10
온디맨드 인스턴스	0	5	10	15
스팟 인스턴스	0	5	10	15

구매 옵션	구매 옵션 전체에서 실행되는 인스턴스 그룹의 크기와 수			
예 2: 기본 0, 0/100% 온디맨드/스팟				
온디맨드 인스턴스 (기본 금액)	0	0	0	0
온디맨드 인스턴스	0	0	0	0
스팟 인스턴스	10	20	30	40
예 3: 기본 0, 60/40% 온디맨드/스팟				
온디맨드 인스턴스 (기본 금액)	0	0	0	0
온디맨드 인스턴스	6	12	18	24
스팟 인스턴스	4	8	12	16
예 4: 기본 0, 100/0% 온디맨드/스팟				
온디맨드 인스턴스 (기본 금액)	0	0	0	0
온디맨드 인스턴스	10	20	30	40
스팟 인스턴스	0	0	0	0

구매 옵션	구매 옵션 전체에서 실행되는 인스턴스 그룹의 크기와 수			
예 5: 기본 12, 0/100% 온디맨드/스팟				
온디맨드 인스턴스 (기본 금액)	10	12	12	12
온디맨드 인스턴스	0	0	0	0
스팟 인스턴스	0	8	18	28

그룹 크기가 커지면 Amazon EC2 Auto Scaling이 지정된 가용 영역 전체에 용량을 고르게 유지하도록 조정합니다. 그런 다음, 지정한 할당 전략에 따라 인스턴스 유형을 시작합니다.

그룹 크기가 줄어들면 Amazon EC2 Auto Scaling이 먼저 두 가지 유형(스팟 또는 온디맨드) 중 종료해야 하는 유형을 식별합니다. 그런 다음, 지정된 가용 영역 전체에서 균형 잡힌 방식으로 인스턴스 종료를 시도합니다. 또한 할당 전략에 더 가깝게 정렬하는 방식으로 인스턴스를 종료하는 것이 좋습니다. 종료 정책에 대한 자세한 내용은 Amazon EC2 Auto Scaling에 대한 종료 정책을 구성합니다.(을)를 참조하세요.

인스턴스 유형의 지역별 가용성

EC2 인스턴스 유형의 가용성은 사용자에 따라 다릅니다. AWS 리전예를 들어, 최신 세대 인스턴스 유형을 지정된 리전에서 아직 사용하지 못할 수 있습니다. 리전마다 인스턴스 가용성이 다르기 때문에 재정의의 여러 인스턴스 유형을 해당 리전에서 사용할 수 없는 경우 프로그래밍 방식 요청을 할 때 문제가 발생할 수 있습니다. 해당 리전에서 사용할 수 없는 여러 인스턴스 유형을 사용하면 요청이 완전히 실패할 수 있습니다. 문제를 해결하려면 다양한 인스턴스 유형으로 요청을 재시도하고 해당 리전에서 각 인스턴스 유형을 사용할 수 있는지 확인합니다. 위치별로 제공되는 인스턴스 유형을 검색하려면 describe-instance-type-offerings명령을 사용합니다. 자세한 내용은 Amazon EC2 사용 설명서에서 Amazon EC2 인스턴스 유형 찾기를 참조하십시오.

관련 리소스

스팟 인스턴스에 대한 자세한 모범 사례는 Amazon EC2 사용 설명서의 EC2 스팟 모범 사례를 참조하십시오.

제한 사항

혼합 인스턴스 정책을 사용하여 Auto Scaling 그룹에 재정의를 추가한 후 `UpdateAutoScalingGroup` API 호출을 사용하여 재정의를 업데이트할 수는 있지만 삭제할 수는 없습니다. 오버라이드를 완전히 제거하려면 먼저 혼합 인스턴스 정책 대신 시작 템플릿 또는 시작 구성을 사용하도록 Auto Scaling 그룹을 전환해야 합니다. 그런 다음 재정의 없이 혼합 인스턴스 정책을 다시 추가할 수 있습니다.

여러 인스턴스 유형에 대한 할당 전략

여러 인스턴스 유형을 사용하는 경우 Amazon EC2 Auto Scaling이 가능한 인스턴스 유형으로 온디맨드 및 스팟 용량을 충족하는 방식을 관리할 수 있습니다. 이를 위해서는 할당 전략을 지정해야 합니다.

혼합 인스턴스 그룹의 모범 사례를 검토하려면 을 참조하십시오혼합 인스턴스 그룹 생성을 위한 설정 개요.

내용

- 스팟 인스턴스
- 온디맨드 인스턴스
- 가중치를 이용한 할당 전략 작동 방식

스팟 인스턴스

Amazon EC2 Auto Scaling은 스팟 인스턴스에 사용할 수 있는 다음과 같은 할당 전략을 제공합니다.

`price-capacity-optimized`(권장)

가격 및 용량 최적화 할당 전략은 가격과 용량을 모두 고려하여 중단될 가능성이 가장 낮으면서 가격이 가장 낮은 스팟 인스턴스 풀을 선택합니다.

시작할 때는 이 전략을 사용하는 것이 좋습니다. 자세한 내용은 AWS 블로그의 EC2 스팟 인스턴스 price-capacity-optimized 할당 전략 소개를 참조하십시오.

`capacity-optimized`

Amazon EC2 Auto Scaling이 시작하는 인스턴스의 수에 대한 최적 용량의 스팟 인스턴스 풀에서 스팟 인스턴스를 요청합니다.

스팟 인스턴스에서 요금은 시간이 지나면서 수요 및 공급의 장기 추세에 따라 서서히 변화합니다. 하지만, 용량은 실시간으로 변동됩니다. `capacity-optimized` 전략은 실시간 용량 데이터를 기준으로 가장 가용성이 높은 풀을 예측하여 자동으로 스팟 인스턴스를 가장 가용성이 높은 풀로 시

작합니다. 이를 통해 작업 재시작 및 체크포인트 수행과 연관된 중단 비용이 높을 수 있는 워크로드에 대한 중단 가능성을 최소화하는 데 도움이 됩니다. 특정 인스턴스 유형이 먼저 시작될 확률을 높이려면 `capacity-optimized-prioritized`를 사용합니다.

`capacity-optimized-prioritized`

시작 템플릿 재정의 목록의 인스턴스 유형 순서를 가장 높은 우선순위에서 가장 낮은 우선순위 순서로(목록의 첫 번째부터 마지막까지) 설정합니다. Amazon EC2 Auto Scaling은 최상의 노력으로 인스턴스 유형 우선순위를 준수하지만 먼저 용량을 최적화합니다. 이 옵션은 중단 가능성을 최소화해야 하지만 특정 인스턴스 유형에 대한 선호도가 중요한 워크로드에 적합합니다. 온디맨드 할당 전략을 `prioritized`로 설정하면 온디맨드 용량을 이행할 때 동일한 우선순위가 적용됩니다.

`lowest-price`

Amazon EC2 Auto Scaling은 최저가 풀 설정에 대해 지정한 N개의 스팟 풀 개수에 걸쳐 가용 영역 내에서 최저가인 풀을 사용하여 스팟 인스턴스를 요청합니다. 예를 들어, 인스턴스 유형 4개와 가용 영역 4개를 지정한 경우 Auto Scaling 그룹은 최대 16개의 스팟 풀에 액세스할 수 있습니다. (가용 영역당 4개) 할당 전략을 위해 스팟 풀 2개(N=2)를 지정한 경우 Auto Scaling 그룹이 가용 영역당 가장 저렴한 두 개의 풀에서 스팟 용량을 충족할 수 있습니다.

이 전략은 인스턴스 가격만 고려하고 용량 가용성은 고려하지 않기 때문에 중단률이 높아질 수 있습니다.

Amazon EC2 Auto Scaling은 지정된 N개의 풀에서 스팟 인스턴스를 끌어오려고 합니다. 하지만, 원하는 용량을 충족하기 전에 풀에 스팟 용량이 부족해질 경우 Amazon EC2 Auto Scaling은 다음으로 저렴한 풀에서 끌어와 요청을 계속 이행합니다. 원하는 용량을 충족하기 위해, 지정한 N개보다 많은 풀에서 스팟 인스턴스를 받게 될 수 있습니다. 마찬가지로 대부분의 풀에 스팟 용량이 없는 경우 지정한 N개보다 적은 풀에서 원하는 전체 용량을 받게 될 수 있습니다.

> **Note**
>
> AMD SEV-SNP가 켜진 상태에서 스팟 인스턴스를 시작하도록 구성하면 선택한 인스턴스 유형의 온디맨드 시간당 요금의 10%에 상응하는 시간당 사용 요금이 추가로 부과됩니다. 할당 전략에서 가격을 입력으로 사용하는 경우 Amazon EC2 Auto Scaling에는 이 추가 요금이 포함되지 않고 스팟 가격만 사용됩니다.

온디맨드 인스턴스

Amazon EC2 Auto Scaling은 온디맨드 인스턴스에 사용할 수 있는 다음 할당 전략을 제공합니다.

`lowest-price`

Amazon EC2 Auto Scaling은 현재 온디맨드 가격을 기준으로 각 가용 영역에 가장 저렴한 인스턴스 유형을 자동으로 배포합니다.

원하는 용량을 충족하기 위해서, 각 가용 영역에서 둘 이상의 인스턴스 유형의 온디맨드 인스턴스를 수신할 수 있습니다. 이는 요청한 용량에 따라 달라집니다.

`prioritized`

온디맨드 용량을 이행할 때, Amazon EC2 Auto Scaling은 시작 템플릿 재정의 목록에 있는 인스턴스 유형 순서를 기반으로 우선 사용할 인스턴스 유형을 결정합니다. 예를 들어, 세 가지 시작 템플릿 재정의를 `c5.large`, `c4.large`, `c3.large` 순으로 지정했다고 가정해 보겠습니다. 온디맨드 인스턴스가 시작되면 Auto Scaling 그룹이 `c5.large`, `c4.large`, `c3.large` 순으로 온디맨드 용량을 채웁니다.

온디맨드 인스턴스의 우선순위 순서를 관리할 때는 다음 사항을 고려하세요.

- 사용량에 대해 선결제하면 예약 인스턴스 또는 절감형 플랜을 사용하여 온디맨드 인스턴스에 대해 상당한 할인을 받을 수 있습니다. 자세한 정보는 Amazon EC2 요금 페이지를 참조하세요.
- 예약 인스턴스의 경우 Amazon EC2 Auto Scaling에서 일치하는 인스턴스 유형을 시작할 경우 일반 온디맨드 인스턴스 요금의 할인율이 적용됩니다. 따라서, `c4.large`에 사용되지 않은 예약 인스턴스가 있는 경우 인스턴스 유형 우선순위를 설정하여 예약 인스턴스의 `c4.large` 인스턴스 유형에 가장 높은 우선순위를 지정할 수 있습니다. `c4.large` 인스턴스가 시작되면 예약 인스턴스 요금을 받게 됩니다.
- 절감형 플랜을 사용하는 경우, Amazon EC2 인스턴스 절감형 플랜 또는 컴퓨팅 절감형 플랜을 사용할 때 일반 온디맨드 인스턴스 요금의 할인율이 적용됩니다. 절감형 플랜을 사용하면 인스턴스 유형의 우선순위를 유연하게 지정할 수 있습니다. 절감형 플랜이 적용되는 인스턴스 유형을 사용하는 한, 어떤 우선순위로든 인스턴스를 설정할 수 있습니다. 또한 가끔 절감형 플랜 할인 요금을 받으면서 인스턴스 유형의 전체 순서를 변경할 수 있습니다. 절감형 플랜에 대한 자세한 정보는 절감형 플랜 사용 설명서를 참조하세요.

가중치를 이용한 할당 전략 작동 방식

오버라이드 (`"DesiredCapacityType": "vcpu"`또는 `"DesiredCapacityType": "memory-mib"` 그룹 수준) 에서 `WeightedCapacity` 파라미터를 지정하면 할당 전략은 다른 Auto Scaling 그룹과 동일하게 작동합니다.

유일한 차이점은 `lowest-price` OR `price-capacity-optimized` 전략을 선택할 때 각 가용 영역에서 단위당 가격이 가장 낮은 인스턴스 풀에서 인스턴스를 생성한다는 것입니다. 자세한 정보는 인스턴스 가중치를 사용하도록 Auto Scaling 그룹을 구성합니다.을 참조하세요.

예를 들어, 다양한 양의 vCPU를 가진 여러 인스턴스 유형이 있는 Auto Scaling 그룹이 있다고 가정합니다. 스팟 및 온디맨드 할당 전략에 대해 `lowest-price`를 사용합니다. 각 인스턴스 유형의 vCPU 수를 기준으로 가중치를 할당하도록 선택하면 Amazon EC2 Auto Scaling은 이행 시점에 할당된 가중치 값당(예: vCPU당) 가격이 가장 저렴한 인스턴스 유형을 시작합니다. 스팟 인스턴스인 경우 이는 vCPU당 최저 스팟 가격을 의미합니다. 온디맨드 인스턴스인 경우 이는 vCPU당 최저 온디맨드 가격을 의미합니다.

속성 기반 인스턴스 유형 선택을 사용하여 혼합 인스턴스 그룹 생성

혼합 인스턴스 그룹을 위한 인스턴스 유형을 수작업으로 선택하는 대신 컴퓨팅 요건을 기술하는 인스턴스 속성 집합을 지정할 수 있습니다. Amazon EC2 Auto Scaling이 인스턴스를 시작할 때 Auto Scaling 그룹에서 사용하는 모든 인스턴스 유형은 필요한 인스턴스 속성과 일치해야 합니다. 이를 속성 기반 인스턴스 유형 선택이라고 합니다.

이 접근 방식은 컨테이너, 빅 데이터 및 CI/CD와 같이 사용하는 인스턴스 유형에 대해 유연한 워크로드 및 프레임워크에 이상적입니다.

속성 기반 인스턴스 유형을 선택하면 다음과 같은 이점이 있습니다.

- 스팟 인스턴스를 위한 최적의 유연성 — Amazon EC2 Auto Scaling은 다양한 인스턴스 유형 중에서 선택하여 스팟 인스턴스를 시작할 수 있습니다. 이는 Amazon EC2 스팟 서비스가 필요한 양의 컴퓨팅 용량을 찾고 할당할 수 있는 더 나은 기회를 제공하는 인스턴스 유형에 대한 유연성이라는 스팟 모범 사례를 충족합니다.
- 쉽게 올바른 인스턴스 유형 사용 - 사용 가능한 인스턴스 유형이 너무 많기 때문에 워크로드에 적합한 인스턴스 유형을 찾는 데 시간이 많이 걸릴 수 있습니다. 인스턴스 속성을 지정하면 인스턴스 유형에는 워크로드에 필요한 속성이 자동으로 포함됩니다.
- 새 인스턴스 유형 자동 사용 — Auto Scaling 그룹은 출시되는 대로 차세대 인스턴스 유형을 사용할 수 있습니다. 최신 세대 인스턴스 유형은 요건과 일치하고 Auto Scaling 그룹에 대해 선택한 할당 전략과 일치하면 자동으로 사용됩니다.

주제

- 속성 기반 인스턴스 유형 선택 작동 방법
- 가격 보호

- 사전 조건
- 속성 기반 인스턴스 유형 선택 기능을 사용하여 혼합 인스턴스 그룹 생성 (콘솔)
- 속성 기반 인스턴스 유형 선택 () 을 사용하여 혼합 인스턴스 그룹을 생성합니다.AWS CLI
- 구성의 예
- 인스턴스 유형 미리 보기
- 관련 리소스

속성 기반 인스턴스 유형 선택 작동 방법

속성 기반 인스턴스 유형 선택을 사용하면 특정 인스턴스 유형의 목록을 제공하는 대신 다음과 같이 인스턴스에 필요한 인스턴스 속성 목록을 제공할 수 있습니다.

- vCPU 수 – 인스턴스당 최소 및 최대 vCPU 수입니다.
- 메모리 — 인스턴스당 최소 및 최대 GiBs 메모리입니다.
- 로컬 스토리지 - 로컬 스토리지에 EBS를 사용할지 아니면 인스턴스 스토어 볼륨을 사용할지입니다.
- 성능 버스트 기능 – T4g, T3a, T3 및 T2 유형을 포함한 T 인스턴스 패밀리를 사용할지 여부입니다.

인스턴스 요구 사항을 정의하는 데 사용할 수 있는 다양한 옵션이 있습니다. 각 옵션과 기본값에 대한 설명은 Amazon EC2 Auto Scaling API 참조를 참조하십시오 InstanceRequirements.

Auto Scaling 그룹은 인스턴스를 시작해야 하는 경우 지정된 속성과 일치하고 해당 가용 영역에서 사용할 수 있는 인스턴스 유형을 검색합니다. 그러면 할당 전략에 따라 일치하는 인스턴스 유형 중 어느 것을 시작할지 결정합니다. 기본적으로 속성 기반 인스턴스 유형 선택에는 Auto Scaling 그룹이 예산 임계값을 초과하는 인스턴스 유형을 시작하지 못하도록 하는 가격 보호 기능이 활성화되어 있습니다.

기본적으로 Auto Scaling 그룹의 원하는 용량을 설정할 때는 인스턴스 수를 측정 단위로 사용합니다. 즉, 각 인스턴스가 하나의 단위로 계산됩니다.

또는 원하는 용량 값을 vCPU 수 또는 메모리 양으로 설정할 수 있습니다. 이렇게 하려면 또는 `UpdateAutoScalingGroup` API 작업의 AWS Management Console 또는 `DesiredCapacityType` 속성에 있는 원하는 용량 유형 드롭다운 필드를 사용하십시오. `CreateAutoScalingGroup` 그러면 Amazon EC2 Auto Scaling이 원하는 vCPU 또는 메모리 용량을 충족하는 데 필요한 수의 인스턴스를 시작합니다. 예를 들어 vCPU를 원하는 용량 유형으로 사용하고 각각 vCPU가 2개인 인스턴스를 사용하는 경우, 원하는 용량의 vCPU가 10개이면 인스턴스 5개가 시작됩니다. 이는 인스턴스 가중치에 대한 유용한 대안입니다.

가격 보호

가격 보호를 사용하면 Auto Scaling 그룹에서 시작한 EC2 인스턴스에 대해 지불할 최고 가격을 지정할 수 있습니다. 가격 보호는 사용자가 지정한 속성에 맞더라도 너무 비싸다고 생각되는 인스턴스 유형을 Auto Scaling 그룹에서 사용하지 못하도록 하는 기능입니다.

가격 보호는 기본적으로 활성화되며 온디맨드 인스턴스와 스팟 인스턴스에 대해 별도의 가격 임계값이 있습니다. Amazon EC2 Auto Scaling에서 새 인스턴스를 시작해야 하는 경우 관련 임계값을 초과하는 모든 인스턴스 유형은 시작되지 않습니다.

주제

- 온디맨드 가격 보호
- 스팟 가격 보호
- 가격 보호를 사용자 정의하십시오.

온디맨드 가격 보호

온디맨드 인스턴스의 경우 지불할 의사가 있는 최고 온디맨드 가격을 식별된 온디맨드 가격보다 높은 백분율로 정의합니다. 식별된 온디맨드 요금은 지정된 속성을 가진 현재 세대 C, M 또는 R 인스턴스 유형의 최저가입니다.

온디맨드 가격 보호 값이 명시적으로 정의되지 않은 경우 식별된 온디맨드 가격보다 20% 높은 기본 최고 온디맨드 가격이 사용됩니다.

스팟 가격 보호

기본적으로 Amazon EC2 Auto Scaling은 최적의 스팟 인스턴스 가격 보호를 자동으로 적용하여 다양한 인스턴스 유형 중에서 일관되게 선택할 수 있도록 합니다. 가격 보호를 직접 수동으로 설정할 수도 있습니다. 하지만 Amazon EC2 Auto Scaling이 대신 처리하도록 하면 스팟 용량이 충족될 가능성을 높일 수 있습니다.

다음 옵션 중 하나를 사용하여 가격 보호를 수동으로 지정할 수 있습니다. 가격 보호를 수동으로 설정하는 경우 첫 번째 옵션을 사용하는 것이 좋습니다.

- 식별된 온디맨드 요금의 백분율 — 식별된 온디맨드 가격은 지정된 속성을 가진 현재 세대 C, M 또는 R 인스턴스 유형 중 가장 저렴한 가격입니다.
- 식별된 스팟 가격보다 높은 비율 — 식별된 스팟 가격은 지정된 속성을 가진 현재 세대 C, M 또는 R 인스턴스 유형 중 가장 낮은 가격의 가격입니다. 스팟 가격이 변동될 수 있고 이에 따라 가격 보호 임계값도 변동될 수 있으므로 이 옵션은 사용하지 않는 것이 좋습니다.

가격 보호를 사용자 정의하십시오.

Amazon EC2 Auto Scaling 콘솔에서 또는 SDK를 사용하여 AWS CLI 가격 보호 임계값을 사용자 지정할 수 있습니다.

- 콘솔에서 추가 인스턴스 속성의 온디맨드 가격 보호 및 스팟 가격 보호 설정을 사용하십시오.
- InstanceRequirements구조에서 온디맨드 인스턴스 가격 보호 임계값을 지정하려면 속성을 사용하십시오. `OnDemandMaxPricePercentageOverLowestPrice` 스팟 인스턴스 가격 보호 임계값을 지정하려면 `MaxSpotPriceAsPercentageOfOptimalOnDemandPrice` 또는 `SpotMaxPricePercentageOverLowestPrice` 속성을 사용합니다.

원하는 용량 유형 (`DesiredCapacityType`) 을 vCPU 또는 Memory GiB로 설정하는 경우 가격 보호는 인스턴스당 가격 대신 vCPU당 또는 메모리당 가격을 기준으로 적용됩니다.

가격 보호를 끌 수도 있습니다. 가격 보호 기준이 없음을 나타내려면 999999와 같은 높은 백분율 값을 지정합니다.

> **Note**
>
> 현재 세대의 C, M 또는 R 인스턴스 유형이 지정된 속성과 일치하지 않는 경우에도 가격 보호가 적용됩니다. 일치하는 항목이 없는 경우, 해당 속성과 일치하는 가장 저렴한 현재 세대 인스턴스 유형 또는 그렇지 않은 경우 가장 저렴한 이전 세대 인스턴스 유형의 가격이 책정됩니다.

사전 조건

- 시작 템플릿을 생성합니다. 자세한 설명은 Auto Scaling 그룹에 대한 시작 템플릿 생성 섹션을 참조하세요.
- 시작 템플릿이 아직 스팟 인스턴스를 요청하지 않았는지 확인합니다.

속성 기반 인스턴스 유형 선택 기능을 사용하여 혼합 인스턴스 그룹 생성 (콘솔)

속성 기반 인스턴스 유형 선택을 사용하여 혼합 인스턴스 그룹을 만들려면 다음 절차를 따르세요. 단계를 효율적으로 진행하는 데 도움이 되도록 일부 선택 사항 섹션은 생략했습니다.

대부분의 범용 워크로드의 경우, 필요한 vCPU의 수와 메모리를 지정하는 것으로 충분합니다. 고급 사용 사례의 경우, 스토리지 유형, 네트워크 인터페이스, CPU 제조업체, 액셀러레이터 유형 등의 속성을 지정할 수 있습니다.

혼합 인스턴스 그룹의 모범 사례를 검토하려면 을 참조하십시오. 혼합 인스턴스 그룹 생성을 위한 설정 개요

혼합 인스턴스 그룹을 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. 화면 상단의 탐색 모음에서 시작 템플릿을 만들 때 사용한 지역과 동일한 AWS 리전 을 선택합니다.
3. Create an Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.
4. 시작 템플릿 또는 구성 선택 페이지에서 Auto Scaling group name(Auto Scaling 그룹 명칭)에 Auto Scaling 그룹의 이름을 입력합니다.
5. 시작 템플릿을 선택하려면 다음을 수행하십시오:
   a. 시작 템플릿에서 기존 시작 템플릿을 선택합니다.
   b. Launch template version(시작 템플릿 버전)에서 Auto Scaling 그룹이 스케일 아웃 시 시작 템플릿의 기본 버전을 사용할지, 최신 버전을 사용할지, 아니면 특정 버전을 사용할지를 선택합니다.
   c. 시작 템플릿이 사용하려는 모든 옵션을 지원하는지 확인한 후 다음을 선택합니다.
6. 인스턴스 시작 옵션 선택 페이지에서 다음을 선택합니다:
   a. 인스턴스 유형 요건(Instance type requirements)에서 시작 템플릿 재정의(Override launch template)를 선택합니다.

      > **Note**
      > vCPU 및 메모리와 같은 인스턴스 속성 집합이 이미 포함된 시작 템플릿을 선택한 경우, 인스턴스 속성이 표시됩니다. 이러한 속성은 Auto Scaling 그룹 속성에 추가되며, 언제든지 Amazon EC2 Auto Scaling 콘솔에서 업데이트할 수 있습니다.

   b. 인스턴스 속성 지정(Specify instance attributes)에서 vCPU 및 메모리 요건을 입력하여 시작합니다.
      - vCPU(vCPUs)에 원하는 최소 및 최대 vCPU 수를 입력합니다. 무한을 지정하려면 최소 없음(No minimum), 최대 없음(No maximum) 또는 둘 다 선택합니다.
      - 메모리(GiB)(Memory (GiB))에 원하는 최소 및 최대 메모리 양을 입력합니다. 무한을 지정하려면 최소 없음(No minimum), 최대 없음(No maximum) 또는 둘 다 선택합니다.

c. (선택 사항) 추가 인스턴스 속성(Additional instance attributes)에서 필요에 따라 하나 이상의 속성을 지정하여 컴퓨팅 요구 사항을 더 자세히 표현할 수 있습니다. 각 추가 속성은 요청에 제약 조건을 추가합니다.

d. 일치하는 인스턴스 유형 미리보기를 확장하여 지정된 속성을 가진 인스턴스 유형을 확인하십시오.

e. 인스턴스 구매 옵션에서 인스턴스 배포에 대해 각각 온디맨드 인스턴스로 및 스팟 인스턴스로 시작시킬 그룹의 백분율을 지정합니다. 귀하의 애플리케이션이 무상태이고 내결함성이 있으며 중단되는 인스턴스를 처리할 수 있는 경우, 귀하는 더 높은 백분율의 스팟 인스턴스를 지정할 수 있습니다.

f. (옵션) 스팟 인스턴스를 위한 백분율을 지정할 때 온디맨드 기본 용량 포함란을 선택한 다음 온디맨드 인스턴스가 충족해야 하는 Auto Scaling 그룹 초기 용량의 최소량을 지정하십시오. 기본 용량을 초과하는 용량은 인스턴스 배치(Instances distribution) 설정을 사용하여 시작할 온디맨드 인스턴스 및 스팟 인스턴스 수를 결정합니다.

g. 할당 전략(Allocation strategies) 아래에서 온디맨드 할당 전략(On-Demand allocation strategy)에 대해 최저가(Lowest price)가 자동으로 선택되며 변경할 수 없습니다.

h. 스팟 할당 전략(Spot allocation strategy)에서 할당 전략을 선택합니다. Price capacity optimized(가격 용량 최적화)가 기본적으로 선택됩니다. Lowest price(최저가)는 기본적으로 숨겨져 있으며 Show all strategies(모든 전략 보기)를 선택한 경우에만 표시됩니다. 최저가를 선택한 경우, 최저가 풀의 수를 입력하여 최저가 풀을 분산합니다.

i. 용량 재조정에서 용량 재조정을 활성화 또는 비활성화할지 선택합니다. 용량 재조정을 사용하여 스팟 중단으로 인해 스팟 인스턴스가 해지되려 할 때 자동화를 통해 대응하십시오. 자세한 설명은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리 섹션을 참조하세요.

j. 네트워크(Network)의 VPC에서 VPC를 선택합니다. Auto Scaling 그룹은 시작 템플릿에서 지정한 보안 그룹과 동일한 VPC에 생성되어야 합니다.

k. 가용 영역 및 서브넷(Availability Zones and subnets)에서 지정한 VPC에 있는 서브넷을 하나 이상 선택합니다. 여러 가용 영역의 서브넷을 사용하여 가용성을 높일 수 있습니다. 자세한 설명은 VPC 서브넷 선택 시 고려 사항 섹션을 참조하세요.

l. 다음, 다음을 선택합니다.

7. 그룹 크기 및 조정 정책 구성 단계에서는 다음을 수행하십시오:

a. 원하는 용량을 인스턴스 이외의 다른 단위로 측정하려면, 그룹 크기, 원하는 용량 유형에서 적절한 옵션을 선택합니다. Units(단위), vCPU, Memory GiB(메모리 GiB)가 지원됩니다. 기본적으로 Amazon EC2 Auto Scaling은 인스턴스 수로 변환되는 Units(단위)를 지정합니다.

b. 원하는 용량이란 Auto Scaling 그룹의 최초 크기입니다.

c. 스케일링 섹션의 스케일링 제한에서 희망 용량에 대한 새 값이 최소 희망 용량 및 최대 희망 용량보다 큰 경우, 최대 희망 용량은 자동으로 새 희망 용량 값으로 증가합니다. 필요에 따라 이러한 한도를 변경할 수 있습니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.

8. Skip to review(검토로 건너뛰기)를 선택합니다.
9. 검토(Review) 페이지에서 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.

속성 기반 인스턴스 유형 선택 () 을 사용하여 혼합 인스턴스 그룹을 생성합니다.AWS CLI

명령행을 사용하여 혼합 인스턴스 그룹을 생성하려면

다음 명령 중 하나를 사용합니다.

- create-auto-scaling-group (AWS CLI)
- New-as AutoScalingGroup ()AWS Tools for Windows PowerShell

구성의 예

를 사용하여 속성 기반 인스턴스 유형 선택이 포함된 Auto Scaling 그룹을 생성하려면 다음 create-auto-scaling-group명령을 사용합니다. AWS CLI

다음과 같은 인스턴스 속성이 지정됩니다.

- `VCpuCount` - 인스턴스 유형에는 최소 4개의 vCPU와 최대 8개의 vCPU가 있어야 합니다.
- `MemoryMiB` - 인스턴스 유형에 최소 16,384MiB의 메모리가 있어야 새 시작 템플릿 버전을 생성하지 않은 경우합니다.
- `CpuManufacturers` - 인스턴스 유형에는 인텔에서 제조한 CPU가 있어야 합니다.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

다음은 예 `config.json` 파일입니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredCapacityType": "units",
    "MixedInstancesPolicy": {
```

```
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [{
                "InstanceRequirements": {
                    "VCpuCount": {"Min": 4, "Max": 8},
                    "MemoryMiB": {"Min": 16384},
                    "CpuManufacturers": ["intel"]
                }
            }]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "price-capacity-optimized"
        }
    },
    "MinSize": 0,
    "MaxSize": 100,
    "DesiredCapacity": 4,
    "DesiredCapacityType": "units",
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

원하는 용량 값을 vCPU 수 또는 메모리 양으로 설정하려면 파일에서 `"DesiredCapacityType": "vcpu"` 또는 `"DesiredCapacityType": "memory-mib"`를 지정합니다. 기본 원하는 용량 유형은 원하는 용량 값을 인스턴스 수로 설정하는 `units`입니다.

YAML

또는 다음 create-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 생성할 수 있습니다. 이는 Auto Scaling 그룹의 유일한 파라미터로 YAML 파일을 참조합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

다음은 예 `config.yaml` 파일입니다.

```
---
AutoScalingGroupName: my-asg
DesiredCapacityType: units
MixedInstancesPolicy:
```

```
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceRequirements:
        VCpuCount:
          Min: 2
          Max: 4
        MemoryMiB:
          Min: 2048
        CpuManufacturers:
        - intel
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: price-capacity-optimized
MinSize: 0
MaxSize: 100
DesiredCapacity: 4
DesiredCapacityType: units
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

원하는 용량 값을 vCPU 수 또는 메모리 양으로 설정하려면 파일에서 `DesiredCapacityType: vcpu` 또는 `DesiredCapacityType: memory-mib`를 지정합니다. 기본 원하는 용량 유형은 원하는 용량 값을 인스턴스 수로 설정하는 `units`입니다.

인스턴스 유형 미리 보기

컴퓨팅 요건과 일치하는 인스턴스 유형을 시작하지 않고 미리 보고 필요한 경우, 요건을 조정할 수 있습니다. Amazon EC2 Auto Scaling 콘솔에서 Auto Scaling 그룹을 생성할 때 인스턴스 유형의 미리 보기가 인스턴스 시작 옵션 선택(Choose instance launch options) 페이지의 일치하는 인스턴스 유형 미리 보기(Preview matching instance types) 섹션에 나타납니다.

또는 SDK를 사용하여 Amazon EC2 GetInstanceTypesFromInstanceRequirementsAPI를 호출하여 인스턴스 유형을 미리 볼 수도 AWS CLI 있습니다. Auto Scaling 그룹을 생성하거나 업데이트하는 데 사용할 정확한 형식으로 `InstanceRequirements` 파라미터를 요청에 전달합니다. 자세한 내용은 Amazon EC2 사용 설명서에서 지정된 속성을 가진 인스턴스 유형 미리 보기를 참조하십시오.

관련 리소스

속성 기반 인스턴스 유형 선택에 대한 자세한 내용은 블로그에서 EC2 Auto Scaling 및 EC2 플릿을 위한 속성 기반 인스턴스 유형 선택을 참조하십시오. AWS

AWS CloudFormation을 사용하여 Auto Scaling 그룹을 생성할 때 속성 기반 인스턴스 유형 선택을 선언할 수 있습니다. 자세한 설명은 AWS CloudFormation 사용자 가이드의 Auto Scaling 템플릿 스니펫에서 예 스니펫을 참조하세요.

인스턴스 유형을 수동으로 선택하여 혼합 인스턴스 그룹 생성

이 항목에서는 인스턴스 유형을 수동으로 선택하여 단일 Auto Scaling 그룹에서 여러 인스턴스 유형을 시작하는 방법을 보여줍니다.

인스턴스 유형을 선택하는 기준으로 인스턴스 속성을 사용하려는 경우 속성 기반 인스턴스 유형 선택을 사용하여 혼합 인스턴스 그룹 생성(을)를 참조하세요.

내용

- 사전 조건
- 혼합 인스턴스 그룹 생성(콘솔)
- 혼합 인스턴스 그룹 생성 (AWS CLI)
- 구성의 예

사전 조건

- 시작 템플릿을 생성합니다. 자세한 설명은 Auto Scaling 그룹에 대한 시작 템플릿 생성 섹션을 참조하세요.
- 시작 템플릿이 아직 스팟 인스턴스를 요청하지 않았는지 확인합니다.

혼합 인스턴스 그룹 생성(콘솔)

다음 절차를 사용하여 그룹에서 시작할 수 있는 인스턴스 유형을 수동으로 선택하여 혼합 인스턴스 그룹을 생성합니다. 단계를 효율적으로 진행하는 데 도움이 되도록 일부 선택 사항 섹션은 생략했습니다.

혼합 인스턴스 그룹의 모범 사례를 검토하려면 을 참조하십시오혼합 인스턴스 그룹 생성을 위한 설정 개요.

혼합 인스턴스 그룹을 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. 화면 상단의 탐색 모음에서 시작 템플릿을 만들 때 사용한 지역과 동일한 AWS 리전 을 선택합니다.

3. Create an Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.
4. 시작 템플릿 또는 구성 선택 페이지에서 Auto Scaling group name(Auto Scaling 그룹 명칭)에 Auto Scaling 그룹의 이름을 입력합니다.
5. 시작 템플릿을 선택하려면 다음을 수행하십시오:
   a. 시작 템플릿에서 기존 시작 템플릿을 선택합니다.
   b. Launch template version(시작 템플릿 버전)에서 Auto Scaling 그룹이 스케일 아웃 시 시작 템플릿의 기본 버전을 사용할지, 최신 버전을 사용할지, 아니면 특정 버전을 사용할지를 선택합니다.
   c. 시작 템플릿이 사용하려는 모든 옵션을 지원하는지 확인한 후 Next(다음)를 선택합니다.
6. 인스턴스 시작 옵션 선택 페이지에서 다음을 수행합니다.
   a. Instance type requirements(인스턴스 유형 요구 사항)에서 Override launch template(시작 템플릿 재정의), Manually add instance types(수동으로 인스턴스 유형 추가)를 선택합니다.
   b. 인스턴스 유형을 선택합니다. 권장 사항을 시작점으로 사용할 수 있습니다. Family and generation flexible(유연한 패밀리 및 세대)이 기본적으로 선택됩니다.
      - 인스턴스 유형의 순서를 변경하려면 화살표를 사용합니다. 우선순위 지정을 지원하는 할당 전략을 선택하면 인스턴스 유형 순서에 따라 시작 우선순위가 설정됩니다.
      - 인스턴스 유형을 제거하려면 X를 선택합니다.
      - (선택 사항) 가중치 열의 상자에 대해 각 인스턴스 유형에 상대적 가중치를 할당합니다. 이렇게 하려면, 해당 유형의 인스턴스가 그룹의 원하는 용량에 포함되는 단위 수를 입력합니다. 인스턴스 유형마다 서로 다른 vCPU, 메모리, 스토리지 또는 네트워크 대역폭 기능을 제공하는 경우 이렇게 하는 것이 유용할 수 있습니다. 자세한 내용은 인스턴스 가중치를 사용하도록 Auto Scaling 그룹을 구성합니다.(을)를 참조하세요.

        유연한 크기 권장 사항 사용을 선택하는 경우 이 섹션에 포함된 모든 인스턴스 유형에는 자동으로 가중치 값이 지정됩니다. 가중치를 지정하지 않으려면 모든 인스턴스 유형의 Weight(가중치) 열에 있는 상자를 지웁니다.
   c. Instance purchase options(인스턴스 구매 옵션)에서 Instances distribution(인스턴스 배포)에 대해 각각 온디맨드 인스턴스 및 스팟 인스턴스로 시작할 그룹의 비율을 지정합니다. 애플리케이션이 무상태이고 내결함성이 있으며 중단되는 인스턴스를 처리할 수 있는 경우 더 높은 비율의 스팟 인스턴스를 지정할 수 있습니다.
   d. (옵션) 스팟 인스턴스를 위한 백분율을 지정할 때 온디맨드 기본 용량 포함란을 선택한 다음 온디맨드 인스턴스가 충족해야 하는 Auto Scaling 그룹 초기 용량의 최소량을 지정하십시오.

기본 용량을 초과하는 용량은 인스턴스 배치(Instances distribution) 설정을 사용하여 시작할 온디맨드 인스턴스 및 스팟 인스턴스 수를 결정합니다.

e. 할당 전략(Allocation strategies)에서 온디맨드 할당 전략(On-Demand allocation strategy)에 대해 할당 전략을 선택합니다. 인스턴스 유형을 수동으로 선택하면 기본적으로 Prioritized(우선순위 지정됨)가 선택됩니다.

f. 스팟 할당 전략(Spot allocation strategy)에서 할당 전략을 선택합니다. Price capacity optimized(가격 용량 최적화)가 기본적으로 선택됩니다. Lowest price(최저가)는 기본적으로 숨겨져 있으며 Show all strategies(모든 전략 보기)를 선택한 경우에만 표시됩니다.

- 최저가를 선택한 경우 최저가 풀의 수를 입력하여 최저가 풀을 분산합니다.
- 용량 최적화를 선택한 경우 필요에 따라 인스턴스 유형 우선순위 지정 확인란을 선택하여 Amazon EC2 Auto Scaling이 인스턴스 유형 나열 순서에 따라 먼저 시작할 인스턴스 유형을 선택할 수 있습니다.

g. 용량 재분배에서 용량 재분배를 활성화 또는 비활성화할지 선택합니다. 용량 재조정을 사용하여 스팟 중단으로 인해 스팟 인스턴스가 해지되려 할 때 자동화를 통해 대응하십시오. 자세한 설명은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리 섹션을 참조하세요.

h. 네트워크(Network)의 VPC에서 VPC를 선택합니다. Auto Scaling 그룹은 시작 템플릿에서 지정한 보안 그룹과 동일한 VPC에 생성되어야 합니다.

i. 가용 영역 및 서브넷(Availability Zones and subnets)에서 지정한 VPC에 있는 서브넷을 하나 이상 선택합니다. 여러 가용 영역의 서브넷을 사용하여 가용성을 높일 수 있습니다. 자세한 설명은 VPC 서브넷 선택 시 고려 사항 섹션을 참조하세요.

j. 다음, 다음을 선택합니다.

7. 그룹 크기 및 조정 정책 구성 단계에서는 다음을 수행하십시오:

a. 그룹 크기의 원하는 용량에서 시작할 초기 인스턴스 수를 입력합니다.

기본적으로 원하는 용량은 인스턴스 수로 표시됩니다. 인스턴스 유형에 가중치를 할당한 경우 이러한 값을 가중치를 할당할 때 사용되는 것과 동일한 측정 단위(예: vCPU 수)로 변환해야 합니다.

b. 조정 섹션의 조정 제한에서 원하는 용량의 새 값이 최소 희망 용량및 최대 희망 용량보다 큰 경우 원하는 최대 용량이 원하는 새 용량 값으로 자동 증가합니다. 필요에 따라 이러한 한도를 변경할 수 있습니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.

8. Skip to review(검토로 건너뛰기)를 선택합니다.

9. 검토(Review) 페이지에서 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.

혼합 인스턴스 그룹 생성 (AWS CLI)

명령행을 사용하여 혼합 인스턴스 그룹을 생성하려면

다음 명령 중 하나를 사용합니다.

- create-auto-scaling-group (AWS CLI)
- New-AS () AutoScalingGroupAWS Tools for Windows PowerShell

구성의 예

다음 예제 구성은 여러 스팟 할당 전략을 사용하여 혼합 인스턴스 그룹을 생성하는 방법을 보여 줍니다.

> **Note**
>
> 다음 예에서는 JSON 또는 YAML로 형식이 지정된 구성 파일을 사용하는 방법을 보여 줍니다. AWS CLI 버전 1을 사용하는 경우 JSON 형식의 구성 파일을 지정해야 합니다. AWS CLI 버전 2를 사용하는 경우 YAML 또는 JSON 형식의 구성 파일을 지정할 수 있습니다.

예

- 예제 1: capacity-optimized 할당 전략을 사용하여 스팟 인스턴스 시작
- 예제 2: capacity-optimized-prioritized 할당 전략을 사용하여 스팟 인스턴스 시작
- 예제 3: 두 개의 풀로 분산된 lowest-price 할당 전략을 사용한 스팟 인스턴스 시작
- 예 4: price-capacity-optimized 할당 전략을 사용하여 스팟 인스턴스 시작

예제 1: **`capacity-optimized`** 할당 전략을 사용하여 스팟 인스턴스 시작

다음 create-auto-scaling-group명령을 실행하면 다음을 지정하는 Auto Scaling 그룹이 생성됩니다.

- 온디맨드 인스턴스로 시작할 그룹의 백분율(`0`) 및 온디맨드 인스턴스의 처음 시작 기본 개수(`1`)
- 우선순위(`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.large`)에 따라 시작할 인스턴스 유형
- 인스턴스(`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`)를 시작하는 서브넷 각각 다른 가용 영역에 해당합니다.
- 시작 템플릿(`my-launch-template`) 및 시작 템플릿 버전(`$Default`)

Amazon EC2 Auto Scaling이 온디맨드 용량을 채우려고 시도하는 경우 `c5.large` 인스턴스 유형을 먼저 시작합니다. 스팟 인스턴스는 스팟 인스턴스 용량에 따라 각 가용 영역의 최적의 스팟 풀에서 나옵니다.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 파일에는 다음 내용이 포함되어 있습니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
```

```
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

또는 다음 create-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 생성할 수 있습니다. 이는 YAML 파일을 Auto Scaling 그룹의 유일한 파라미터로 참조합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

`config.yaml` 파일에는 다음 콘텐츠가 포함되어 있습니다.

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
```

```
    SpotAllocationStrategy: capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

예제 2: **`capacity-optimized-prioritized`** 할당 전략을 사용하여 스팟 인스턴스 시작

다음 create-auto-scaling-group명령을 실행하면 다음을 지정하는 Auto Scaling 그룹이 생성됩니다.

- 온디맨드 인스턴스로 시작할 그룹의 백분율(`0`) 및 온디맨드 인스턴스의 처음 시작 기본 개수(`1`)
- 우선순위(`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.large`)에 따라 시작할 인스턴스 유형
- 인스턴스(`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`)를 시작하는 서브넷 각각 다른 가용 영역에 해당합니다.
- 시작 템플릿(`my-launch-template`) 및 시작 템플릿 버전(`$Latest`)

Amazon EC2 Auto Scaling이 온디맨드 용량을 채우려고 시도하는 경우 `c5.large` 인스턴스 유형을 먼저 시작합니다. Amazon EC2 Auto Scaling은 스팟 용량을 충족하려고 할 때 최대한 인스턴스 유형 우선순위를 준수합니다. 그러나, 먼저 용량을 최적화합니다.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 파일에는 다음 내용이 포함되어 있습니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
```

```
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 1,
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized-prioritized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

또는 다음 create-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 생성할 수 있습니다. 이는 YAML 파일을 Auto Scaling 그룹의 유일한 파라미터로 참조합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

`config.yaml` 파일에는 다음 콘텐츠가 포함되어 있습니다.

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandBaseCapacity: 1
    OnDemandPercentageAboveBaseCapacity: 0
    SpotAllocationStrategy: capacity-optimized-prioritized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

예제 3: 두 개의 풀로 분산된 **`lowest-price`** 할당 전략을 사용한 스팟 인스턴스 시작

다음 create-auto-scaling-group명령을 실행하면 다음을 지정하는 Auto Scaling 그룹이 생성됩니다.

- 온디맨드 인스턴스로 시작할 그룹의 백분율(`50`) (이것은 시작할 온디맨드 인스턴스의 기본 수를 지정하지 않습니다.)
- 우선순위(`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.large`)에 따라 시작할 인스턴스 유형
- 인스턴스(`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`)를 시작하는 서브넷 각각 다른 가용 영역에 해당합니다.
- 시작 템플릿(`my-launch-template`) 및 시작 템플릿 버전(`$Latest`)

Amazon EC2 Auto Scaling이 온디맨드 용량을 채우려고 시도하는 경우 `c5.large` 인스턴스 유형을 먼저 시작합니다. 스팟 용량의 경우, Amazon EC2 Auto Scaling은 각 가용 영역 내 가장 낮은 가격의 풀 2개에 걸쳐 스팟 인스턴스를 균등하게 시작하려고 시도합니다.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 파일에는 다음 내용이 포함되어 있습니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
```

```
            "SpotAllocationStrategy": "lowest-price",
            "SpotInstancePools": 2
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

또는 다음 create-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 생성할 수 있습니다. 이는 YAML 파일을 Auto Scaling 그룹의 유일한 파라미터로 참조합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

`config.yaml` 파일에는 다음 콘텐츠가 포함되어 있습니다.

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 50
    SpotAllocationStrategy: lowest-price
    SpotInstancePools: 2
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
```

```
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

예 4: **`price-capacity-optimized`** 할당 전략을 사용하여 스팟 인스턴스 시작

다음 create-auto-scaling-group명령을 실행하면 다음을 지정하는 Auto Scaling 그룹이 생성됩니다.

- 온디맨드 인스턴스로 시작할 그룹의 백분율(`30`) (이것은 시작할 온디맨드 인스턴스의 기본 수를 지정하지 않습니다.)
- 우선순위(`c5.large`, `c5a.large`, `m5.large`, `m5a.large`, `c4.large`, `m4.large`, `c3.large`, `m3.large`)에 따라 시작할 인스턴스 유형
- 인스턴스(`subnet-5ea0c127`, `subnet-6194ea3b`, `subnet-c934b782`)를 시작하는 서브넷 각각 다른 가용 영역에 해당합니다.
- 시작 템플릿(`my-launch-template`) 및 시작 템플릿 버전(`$Latest`)

Amazon EC2 Auto Scaling이 온디맨드 용량을 채우려고 시도하는 경우 `c5.large` 인스턴스 유형을 먼저 시작합니다. 스팟 인스턴스의 경우, Amazon EC2 Auto Scaling은 가능한 한 낮은 가격뿐만 아니라 시작하는 인스턴스 수에 대한 최적의 용량으로 스팟 인스턴스 풀에서 스팟 인스턴스를 시작하려고 시도합니다.

JSON

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 파일에는 다음 내용이 포함되어 있습니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
```

```
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 30,
            "SpotAllocationStrategy": "price-capacity-optimized"
        }
    },
    "MinSize": 1,
    "MaxSize": 5,
    "DesiredCapacity": 3,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

또는 다음 create-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 생성할 수 있습니다. 이는 YAML 파일을 Auto Scaling 그룹의 유일한 파라미터로 참조합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

`config.yaml` 파일에는 다음 콘텐츠가 포함되어 있습니다.

```
---
AutoScalingGroupName: my-asg
MixedInstancesPolicy:
```

```
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
  InstancesDistribution:
    OnDemandPercentageAboveBaseCapacity: 30
    SpotAllocationStrategy: price-capacity-optimized
MinSize: 1
MaxSize: 5
DesiredCapacity: 3
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

인스턴스 가중치를 사용하도록 Auto Scaling 그룹을 구성합니다.

여러 인스턴스 유형을 사용하는 경우 각 인스턴스 유형에 연결할 단위 수를 지정한 다음 동일한 측정 단위를 사용하여 그룹의 용량을 지정할 수 있습니다. 이 용량 지정 옵션을 가중치라고 합니다.

예를 들어, 최소 8개의 vCPU 및 15GiB의 RAM에서 최적의 성능을 발휘하는 컴퓨팅 집약적 애플리케이션을 실행하는 경우, `c5.2xlarge`를 기본 유닛으로 사용할 때 다음 EC2 인스턴스 유형은 모두 애플리케이션 요구 사항을 충족합니다.

인스턴스 유형의 예

인스턴스 유형	vCPU	메모리(GiB)
`c5.2xlarge`	8	16
`c5.4xlarge`	16	32
`c5.12xlarge`	48	96
`c5.18xlarge`	72	144
`c5.24xlarge`	96	192

기본적으로 모든 인스턴스 유형은 크기에 관계없이 동일한 가중치를 갖습니다. Amazon EC2 Auto Scaling이 인스턴스를 시작하면 해당 인스턴스 유형이 크든 작든 각 인스턴스가 Auto Scaling 그룹의 원하는 용량에 동일하게 가산됩니다.

하지만 가중치를 사용하면 각 인스턴스 유형에 연결할 단위 수를 지정하는 숫자 값을 할당합니다. 예를 들어, 인스턴스 간의 크기가 서로 다른 경우 `c5.2xlarge` 인스턴스의 가중치는 2, `c5.4xlarge`(2배 더 큼)의 가중치는 4와 같은 식으로 지정할 수 있습니다. 그런 다음, Amazon EC2 Auto Scaling이 그룹을 조정할 때 가중치는 원하는 용량에 각 인스턴스가 가산되는 유닛 수로 변환됩니다.

가중치는 Amazon EC2 Auto Scaling에서 시작하려는 인스턴스 유형을 변경하지 않고, 대신 할당 전략이 변경합니다. 자세한 내용은 여러 인스턴스 유형에 대한 할당 전략(을)를 참조하세요.

> **Important**
>
> 각 인스턴스 유형의 vCPU 수 또는 메모리 양을 사용하여 원하는 용량을 채우도록 Auto Scaling 그룹을 구성하려면 속성 기반 인스턴스 유형 선택을 사용하는 것이 좋습니다. `DesiredCapacityType`매개변수를 설정하면 이 매개변수에 설정한 값을 기준으로 각 인스턴스 유형과 연결할 단위 수가 자동으로 지정됩니다. 자세한 내용은 속성 기반 인스턴스 유형 선택을 사용하여 혼합 인스턴스 그룹 생성 단원을 참조하십시오.

목차

- 고려 사항
- 인스턴스 가중치 동작
- 가중치를 사용하도록 Auto Scaling 그룹 구성
- 단위 시간당 스팟 가격의 예

고려 사항

이 섹션에서는 가중치를 효과적으로 구현하기 위한 주요 고려 사항에 대해 설명합니다.

- 애플리케이션의 성능 요구 사항에 맞는 몇 가지 인스턴스 유형을 선택하십시오. 기능에 따라 각 인스턴스 유형이 Auto Scaling 그룹의 원하는 용량에 포함되어야 하는 가중치를 결정합니다. 이 가중치는 현재 및 미래의 인스턴스에 적용됩니다.
- 가중치 사이의 큰 범위를 피하세요. 예를 들어 다음으로 큰 인스턴스 유형의 가중치가 200인 경우 인스턴스 유형에 가중치를 1로 지정하지 마십시오. 가장 작은 가중치와 가장 큰 가중치의 차이가 너무

커서도 안 됩니다. 가중치 차이가 심하면 비용 대비 성능 최적화에 부정적인 영향을 미칠 수 있습니다.

- 그룹의 원하는 용량을 인스턴스가 아닌 단위로 지정하십시오. 예를 들어 vCPU 기반 가중치를 사용하는 경우 원하는 코어 수와 최소 및 최대값을 설정합니다.
- 원하는 용량이 가장 큰 가중치보다 최소 2~3배 더 크도록 가중치와 원하는 용량을 설정합니다.

기존 그룹을 업데이트할 때는 다음 사항에 유의하세요.

- 기존 그룹에 가중치를 추가할 때는 현재 사용 중인 모든 인스턴스 유형에 대한 가중치를 포함하십시오.
- 가중치를 추가하거나 변경하면 Amazon EC2 Auto Scaling은 새 가중치 값을 기준으로 원하는 용량에 도달하도록 인스턴스를 시작하거나 종료합니다.
- 인스턴스 유형을 제거하면 해당 유형의 실행 중인 인스턴스는 더 이상 정의되지 않더라도 마지막 가중치를 유지합니다.

인스턴스 가중치 동작

인스턴스 가중치를 사용하는 경우 Amazon EC2 Auto Scaling은 다음과 같은 방식으로 작동합니다.

- 현재 용량이 원하는 용량 이상이 될 수 있습니다. 남은 원하는 용량 단위를 초과하는 인스턴스를 시작한 경우 현재 용량이 원하는 용량을 초과할 수 있습니다. 예를 들어, 두 인스턴스 유형 `c5.2xlarge`와 `c5.12xlarge`를 지정하고 `c5.2xlarge`에 인스턴스 가중치 2, `c5.12xlarge`에 인스턴스 가중치 12를 할당한 경우, 원하는 용량을 충족하기까지 5개 유닛이 남은 상태에서 Amazon EC2 Auto Scaling이 `c5.12xlarge`을 프로비저닝하면 원하는 용량이 7개 유닛만큼 초과됩니다.
- Amazon EC2 Auto Scaling은 인스턴스를 시작할 때 원하는 용량을 초과하는 것보다 가용 영역 전체에 용량을 분배하고 할당 전략을 준수하는 것을 우선시합니다.
- Amazon EC2 Auto Scaling은 사용자가 선호하는 할당 전략을 사용하여 최대 용량 한도를 초과하여 가용 영역 간의 균형을 유지할 수 있습니다. Amazon EC2 Auto Scaling에서 적용하는 엄격한 제한은 원하는 용량에 최대 중량을 더한 값입니다.

가중치를 사용하도록 Auto Scaling 그룹 구성

다음 AWS CLI 예와 같이 가중치를 사용하도록 Auto Scaling 그룹을 구성할 수 있습니다. 콘솔을 사용하는 방법은 인스턴스 유형을 수동으로 선택하여 혼합 인스턴스 그룹 생성(을)를 참조하세요.

가중치를 사용하도록 새 Auto Scaling 그룹을 구성하려면(AWS CLI)

create-auto-scaling-group 명령을 사용합니다. 예를 들어, 다음 명령은 새 Auto Scaling 그룹을 만들고 다음 내용을 지정하여 인스턴스 가중치를 할당합니다.

- 온디맨드 인스턴스로 시작할 그룹의 백분율(0)
- 각 가용 영역의 스팟 인스턴스에 대한 할당 전략(capacity-optimized)
- 우선순위(m4.16xlarge, m5.24xlarge)에 따라 시작할 인스턴스 유형
- 인스턴스 유형(16, 24) 간의 상대적 크기 차이(vCPU)에 해당하는 인스턴스 가중치
- 각각 다른 가용 영역에 해당하는 인스턴스를 시작하는 서브넷(subnet-5ea0c127, subnet-6194ea3b, subnet-c934b782)
- 시작 템플릿(my-launch-template) 및 시작 템플릿 버전($Latest)

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

config.json 파일에는 다음 내용이 포함되어 있습니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "m4.16xlarge",
                    "WeightedCapacity": "16"
                },
                {
                    "InstanceType": "m5.24xlarge",
                    "WeightedCapacity": "24"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 0,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    },
```

```
    "MinSize": 160,
    "MaxSize": 720,
    "DesiredCapacity": 480,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

가중치를 사용하도록 기존 Auto Scaling 그룹을 구성하려면(AWS CLI)

update-auto-scaling-group 명령을 사용합니다. 예를 들어, 다음을 지정하여 아래 명령을 실행하면 기존 Auto Scaling 그룹의 인스턴스 유형에 가중치가 할당됩니다.

- 우선순위(`c5.18xlarge`, `c5.24xlarge`, `c5.2xlarge`, `c5.4xlarge`)에 따라 시작할 인스턴스 유형
- 인스턴스 유형(18, 24, 2, 4) 간의 상대적 크기 차이(vCPU)에 해당하는 인스턴스 가중치
- 가장 큰 가중치보다 크게 증가된 새 원하는 용량

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 파일에는 다음 내용이 포함되어 있습니다.

```
{
    "AutoScalingGroupName": "my-existing-asg",
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "Overrides": [
                {
                    "InstanceType": "c5.18xlarge",
                    "WeightedCapacity": "18"
                },
                {
                    "InstanceType": "c5.24xlarge",
                    "WeightedCapacity": "24"
                },
                {
                    "InstanceType": "c5.2xlarge",
                    "WeightedCapacity": "2"
                },
                {
                    "InstanceType": "c5.4xlarge",
```

```
                    "WeightedCapacity": "4"
                }
            ]
        }
    },
    "MinSize": 0,
    "MaxSize": 100,
    "DesiredCapacity": 100
}
```

명령줄을 사용하여 가중치를 확인하려면

다음 명령 중 하나를 사용합니다.

- describe-auto-scaling-groups (AWS CLI)
- Get-AS () AutoScalingGroupAWS Tools for Windows PowerShell

단위 시간당 스팟 가격의 예

다음 표는 미국 동부(버지니아 북부)의 여러 가용 영역에 있는 스팟 인스턴스의 시간당 가격과 동일한 리전의 온디맨드 인스턴스 가격을 비교한 것입니다. 표시된 가격은 현재 가격이 아니라 예를 든 것이며, 인스턴스 시간당 가격입니다.

예: 인스턴스 시간당 스팟 가격

인스턴스 유형	us-east-1a	us-east-1b	us-east-1c	온디맨드 가격
`c5.2xlarge`	0.180 USD	0.191 USD	0.170 USD	0.34 USD
`c5.4xlarge`	0.341 USD	0.361 USD	0.318 USD	0.68 USD
`c5.12xlarge`	0.779 USD	0.777 USD	0.777 USD	2.04 USD
`c5.18xlarge`	1.207 USD	1.475 USD	1.357 USD	3.06 USD

인스턴스 유형	us-east-1a	us-east-1b	us-east-1c	온디맨드 가격
c5.24xlarge	1.555 USD	1.555 USD	1.555 USD	4.08 USD

인스턴스 가중치를 사용하면 단위 시간당 사용량을 기준으로 비용을 평가할 수 있습니다. 단위 시간당 가격은 인스턴스 유형에 따른 가격을 인스턴스가 나타내는 유닛 수로 나누어 계산합니다. 온디맨드 인스턴스의 경우, 하나의 인스턴스 유형을 배포하는 경우 동일한 인스턴스 유형의 다른 크기를 배포하더라도 단위 시간당 가격이 동일합니다. 그러나, 단위 시간당 스팟 가격은 스팟 풀에 따라 다릅니다.

다음 예제는 인스턴스 가중치를 기준으로 단위 시간당 스팟 가격을 계산하는 방법을 보여줍니다. 간단한 계산을 위해 us-east-1a에서만 스팟 인스턴스를 시작한다고 가정합니다. 단위 시간당 가격은 다음 표에 수집되어 있습니다.

예: 단위 시간당 스팟 가격

인스턴스 유형	us-east-1a	인스턴스 가중치	단위 시간당 가격
c5.2xlarge	0.180 USD	2	0.090 USD
c5.4xlarge	0.341 USD	4	0.085 USD
c5.12xlarge	0.779 USD	12	0.065 USD
c5.18xlarge	1.207 USD	18	0.067 USD
c5.24xlarge	1.555 USD	24	0.065 USD

인스턴스 유형에 대해 서로 다른 시작 템플릿 사용

여러 인스턴스 유형을 사용하는 것 외에도 여러 시작 템플릿을 사용할 수 있습니다.

예를 들어, 컴퓨팅 집약적인 애플리케이션을 위한 Auto Scaling 그룹을 구성하고 C5, C5a 및 C6g 인스턴스 유형을 혼합해 포함하려 한다고 가정해 보세요. 하지만 C6g 인스턴스는 64비트 Arm 아키텍처 기반의 AWS 그라비톤 프로세서를 사용하는 반면, C5 및 C5a 인스턴스는 64비트 인텔 x86 프로세서에서 실행됩니다. C5 및 C5a 인스턴스용 AMI는 두 인스턴스에서 작동하지만 C6g 인스턴스에서는 작동하지 않습니다. 이 문제를 해결하려면 다른 C6g 인스턴스용 시작 템플릿을 사용합니다. C5 및 C5a 인스턴스에 동일한 시작 템플릿을 계속 사용할 수 있습니다.

이 섹션에는 를 사용하여 여러 시작 템플릿 사용과 관련된 작업을 수행하는 절차가 포함되어 있습니다. AWS CLI 현재 이 기능은 AWS CLI 또는 SDK를 사용하는 경우에만 사용할 수 있으며 콘솔에서는 사용할 수 없습니다.

내용

- 여러 시작 템플릿을 사용하도록 Auto Scaling 그룹 구성
- 관련 리소스

여러 시작 템플릿을 사용하도록 Auto Scaling 그룹 구성

다음 예와 같이 여러 시작 템플릿을 사용하도록 Auto Scaling 그룹을 구성할 수 있습니다.

여러 시작 템플릿을 사용하도록 새 Auto Scaling 그룹을 구성하려면(AWS CLI)

create-auto-scaling-group 명령을 사용합니다. 예를 들어, 다음 명령은 새 Auto Scaling 그룹을 생성합니다. 이 파일은 `c5.large`, `c5a.large`, 및 `c6g.large` 인스턴스 유형을 지정하고, `c6g.large` 인스턴스 유형에 대한 새 시작 템플릿을 정의하여 Arm 인스턴스를 시작하는 데 적절한 AMI가 사용되는지 확인합니다. Amazon EC2 Auto Scaling은 인스턴스 유형 순서를 사용하여 온디맨드 용량을 채울 때 우선 사용할 인스턴스 유형을 결정합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 파일에는 다음 내용이 포함되어 있습니다.

```
{
  "AutoScalingGroupName":"my-asg",
  "MixedInstancesPolicy":{
    "LaunchTemplate":{
      "LaunchTemplateSpecification":{
        "LaunchTemplateName":"my-launch-template-for-x86",
        "Version":"$Latest"
      },
      "Overrides":[
        {
          "InstanceType":"c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
        },
```

```
        {
          "InstanceType":"c5.large"
        },
        {
          "InstanceType":"c5a.large"
        }
      ]
    },
    "InstancesDistribution":{
      "OnDemandBaseCapacity": 1,
      "OnDemandPercentageAboveBaseCapacity": 50,
      "SpotAllocationStrategy": "capacity-optimized"
    }
  },
  "MinSize":1,
  "MaxSize":5,
  "DesiredCapacity":3,
  "VPCZoneIdentifier":"subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
  "Tags":[ ]
}
```

여러 시작 템플릿을 사용하도록 기존 Auto Scaling 그룹을 구성하려면(AWS CLI)

update-auto-scaling-group 명령을 사용합니다. 예를 들어, 다음 명령은 이름이 *my-launch-template-for-arm*로 지정된 시작 템플릿을 이름이 *my-asg*인 Auto Scaling 그룹의 *c6g.large* 인스턴스 유형에 할당합니다.

```
aws autoscaling update-auto-scaling-group --cli-input-json file://~/config.json
```

`config.json` 파일에는 다음 콘텐츠가 포함되어 있습니다.

```
{
  "AutoScalingGroupName":"my-asg",
  "MixedInstancesPolicy":{
    "LaunchTemplate":{
      "Overrides":[
        {
          "InstanceType":"c6g.large",
          "LaunchTemplateSpecification": {
            "LaunchTemplateName": "my-launch-template-for-arm",
            "Version": "$Latest"
          }
```

```
        },
        {
          "InstanceType":"c5.large"
        },
        {
          "InstanceType":"c5a.large"
        }
      ]
    }
  }
}
```

Auto Scaling 그룹의 시작 템플릿 확인

다음 명령 중 하나를 사용합니다.

- describe-auto-scaling-groups (AWS CLI)
- Get-as () AutoScalingGroupAWS Tools for Windows PowerShell

관련 리소스

re:Post의 템플릿에서 속성 기반 인스턴스 유형 선택을 사용하여 여러 시작 템플릿을 지정하는 예를 찾을 수 있습니다 AWS CloudFormation .AWS

시작 구성을 사용하여 Auto Scaling 그룹 생성

> **Important**
>
> 2022년 12월 31일 이후에 릴리스된 `CreateLaunchConfiguration` 새 Amazon EC2 인스턴스 유형을 사용하여 를 호출할 수 없습니다. 또한 2023년 6월 1일 이후에 생성된 새 계정에는 이 콘솔을 통해 새 시작 구성을 만들 수 있는 옵션이 없습니다. 2024년 10월 1일부터 새 계정은 콘솔, , API CLI및 를 사용하여 새 시작 구성을 생성할 수 없습니다 CloudFormation. 를 시작 템플릿으로 마이그레이션하여 현재 또는 향후에 새 시작 구성을 생성할 필요가 없도록 합니다. Auto Scaling 그룹을 위한 시작 템플릿 생성에 대한 자세한 설명은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작을 참조하세요.

시작 구성 또는 EC2 인스턴스를 생성한 경우 시작 구성을 EC2 인스턴스의 구성 템플릿으로 사용하는 Auto Scaling 그룹을 생성할 수 있습니다. 시작 구성은 인스턴스의 AMI ID, 인스턴스 유형, 키 페어, 보

안 그룹 및 블록 디바이스 매핑과 같은 정보를 지정합니다. 시작 구성을 생성하는 자세한 설명은 시작 구성 생성 섹션을 참조하십시오.

Auto Scaling 그룹을 만들려면 충분한 권한이 있어야 합니다. 또한 Amazon EC2 Auto Scaling이 아직 존재하지 않는 경우 사용자를 대신하여 작업을 수행하는 데 사용하는 서비스 연결 역할을 생성할 수 있는 충분한 권한이 있어야 합니다. 관리자가 권한을 부여하기 위한 참조로 사용할 수 있는 IAM 정책의 예는 섹션을 참조하세요자격 증명 기반 정책 예시.

내용

- 시작 구성을 사용하여 Auto Scaling 그룹 생성
- 기존 인스턴스의 파라미터를 사용하여 Auto Scaling 그룹 생성

시작 구성을 사용하여 Auto Scaling 그룹 생성

> **Important**
>
> 시작 구성은 시작 구성에서 시작 템플릿으로 아직 마이그레이션하지 않은 고객을 위해 제공하고 있습니다. Auto Scaling 그룹을 위한 시작 템플릿 생성에 대한 자세한 설명은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작을 참조하세요.

Auto Scaling 그룹을 생성할 때는 Amazon EC2 인스턴스, 인스턴스의 가용 영역 및 VPC 서브넷, 원하는 용량, 최소 및 최대 용량 제한을 구성하는 데 필요한 정보를 지정해야 합니다.

다음 절차는 시작 구성을 사용하여 Auto Scaling 그룹을 생성하는 방법을 보여줍니다. 시작 구성을 생성한 후에는 수정할 수가 없지만 Auto Scaling 그룹에 대한 시작 구성을 교체할 수 있습니다. 자세한 설명은 Auto Scaling 그룹에 대한 시작 구성 변경 섹션을 참조하세요.

사전 조건

- 시작 구성을 생성해야 합니다. 자세한 설명은 시작 구성 생성 섹션을 참조하세요.

시작 구성을 사용하여 Auto Scaling 그룹을 생성하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 화면 상단의 탐색 표시줄에서 시작 구성을 생성할 때 사용한 것과 동일한 AWS 리전 것을 선택합니다.

3. Auto Scaling 그룹 생성을 선택합니다.
4. 시작 템플릿 또는 구성 선택 페이지에서 Auto Scaling 그룹 명칭에 Auto Scaling 그룹의 이름을 입력합니다.
5. 시작 구성을 선택하려면 다음을 수행하십시오:
   a. 시작 템플릿(Launch template)에서 시작 구성으로 전환(Switch to launch configuration)을 선택합니다.
   b. 시작 구성에 대해 기존 시작 구성을 선택합니다.
   c. 시작 구성이 사용하려는 모든 옵션을 지원하는지 확인한 후 다음을 선택합니다.
6. 인스턴스 시작 옵션 구성 페이지의 네트워크에서 원하는 VPC항목을 선택합니다VPC. 시작 구성에서 지정한 보안 VPC 그룹과 동일한 위치에 Auto Scaling 그룹을 생성해야 합니다.
7. 가용 영역 및 서브넷의 경우 지정된 서브넷에서 하나 이상의 서브넷을 선택합니다. VPC 여러 가용 영역의 서브넷을 사용하여 가용성을 높일 수 있습니다. 자세한 설명은 VPC 서브넷 선택 시 고려 사항 섹션을 참조하세요.
8. Next(다음)를 선택합니다.

   또는 나머지 기본값을 그대로 두고, 검토로 이동(Skip to review)을 선택할 수 있습니다.
9. (옵션) 고급 옵션 구성(Configure advanced options) 페이지에서 다음 옵션을 구성하고 다음(Next)을 선택합니다.
   a. (선택 사항) 상태 점검, 추가 상태 확인 유형의 경우 Amazon EBS 상태 확인 활성화를 선택합니다. 자세한 내용은 상태 확인을 사용하여 Amazon EBS 볼륨이 손상된 Auto Scaling 인스턴스를 모니터링합니다. 단원을 참조하십시오.
   b. (옵션) 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. 이 시간은 Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.
   c. 추가 설정, 모니터링에서 CloudWatch 그룹 지표 수집을 활성화할지 여부를 선택합니다. 이러한 지표는 해지 인스턴스 수 또는 보류 중인 인스턴스 수와 같은 잠재적 문제의 지표가 될 수 있는 측정값을 제공합니다. 자세한 내용은 Auto Scaling 그룹 및 인스턴스의 CloudWatch 메트릭을 모니터링합니다. 단원을 참조하십시오.
   d. 기본 인스턴스 웜업 활성화에서 이 옵션을 선택하고 애플리케이션의 웜업 시간을 선택합니다. 조정 정책이 있는 Auto Scaling 그룹을 생성하는 경우 기본 인스턴스 웜업 기능은 동적 조정에 사용되는 Amazon CloudWatch 지표를 개선합니다. 자세한 내용은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 단원을 참조하십시오.
10. (옵션) 그룹 크기 및 조정 정책 구성 페이지에서 다음 옵션을 구성하고 다음을 선택합니다.

a. 그룹 사이즈에서 원하는 용량에 시작시킬 초기 인스턴스 수를 입력합니다.

b. 스케일링 섹션의 스케일링 제한에서 희망 용량에 대한 새 값이 최소 희망 용량 및 최대 희망 용량보다 큰 경우, 최대 희망 용량은 자동으로 새 희망 용량 값으로 증가합니다. 필요에 따라 이러한 한도를 변경할 수 있습니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.

c. 자동 조정의 경우, 대상 추적 조정 정책을 생성할지 여부를 선택합니다. Auto Scaling 그룹을 생성한 후에 이 정책을 생성할 수도 있습니다.

   대상 추적 조정 정책을 선택하는 경우, 대상 추적 조정 정책 생성의 지침에 따라 정책을 생성하십시오.

d. 인스턴스 유지 관리 정책에서는 인스턴스 유지 관리 정책을 만들지 여부를 선택합니다. Auto Scaling 그룹을 생성한 후에 이 정책을 생성할 수도 있습니다. 인스턴스 유지 관리 정책 설정의 지침에 따라 정책을 만듭니다.

e. 인스턴스 스케일 인 보호(Instance scale-in protection)에서 인스턴스 스케일 인 보호를 활성화할지를 선택합니다. 자세한 설명은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 섹션을 참조하세요.

11. (옵션) 알림을 받으려면 알림 추가(Add notification)에 알림을 구성하고 다음(Next)을 선택합니다. 자세한 설명은 아마존 EC2 오토 스케일링에 대한 아마존 SNS 알림 옵션 섹션을 참조하세요.

12. (옵션) 태그를 추가하려면 태그 추가(Add tag)를 선택하고 각 태그에 태그 키와 값을 제공한 후 다음(Next)을 선택합니다. 자세한 설명은 Auto Scaling 그룹 및 인스턴스에 태그 지정 섹션을 참조하세요.

13. 검토(Review) 페이지에서 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.

명령행을 사용하여 Auto Scaling 그룹을 생성하려면

다음 명령 중 하나를 사용할 수 있습니다:

- create-auto-scaling-group (AWS CLI)
- New- ASAutoScalingGroup ()AWS Tools for Windows PowerShell

기존 인스턴스의 파라미터를 사용하여 Auto Scaling 그룹 생성

> **Important**
>
> 시작 구성은 시작 구성에서 시작 템플릿으로 아직 마이그레이션하지 않은 고객을 위해 제공하고 있습니다. Auto Scaling 그룹을 시작 템플릿으로 마이그레이션하는 데 대한 자세한 설명은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작을 참조하세요.

Auto Scaling 그룹을 처음 생성하는 경우, 콘솔을 사용하여 기존 EC2 인스턴스에서 시작 템플릿을 생성하는 것이 좋습니다. 그런 다음 시작 템플릿을 사용하여 새 Auto Scaling 그룹을 생성합니다. 이 절차는 Amazon EC2 시작 마법사를 사용하여 Auto Scaling 그룹 생성 섹션을 참조하세요.

다음 절차에서는 다른 인스턴스를 시작하기 위한 기반으로 사용할 기존 인스턴스를 지정하여 Auto Scaling 그룹을 생성하는 방법을 보여줍니다. EC2 인스턴스를 생성하려면 Amazon Machine Image(AMI) ID, 인스턴스 유형, 키 페어 및 보안 그룹 등 여러 파라미터가 필요합니다. 이러한 모든 정보는 크기 조정이 필요할 때 사용자를 대신하여 Amazon EC2 Auto Scaling에서 인스턴스를 시작하는 데 사용됩니다. 이 정보는 시작 템플릿 또는 시작 구성에 저장됩니다.

기존 인스턴스를 사용하면 Amazon EC2 Auto Scaling이 동시에 생성된 시작 구성에 근거하여 인스턴스를 시작하는 Auto Scaling 그룹을 생성합니다. 새 시작 구성은 Auto Scaling 그룹과 이름이 동일하며 식별된 인스턴스의 특정 구성 세부 정보를 포함합니다.

다음 구성 세부 정보가 식별된 인스턴스에서 시작 구성으로 복사됩니다.

- AMI ID
- 인스턴스 유형
- 키 페어
- 보안 그룹
- IP 주소 유형(퍼블릭 또는 프라이빗)
- IAM 인스턴스 프로파일(해당하는 경우)
- 모니터링(true 또는 false)
- EBS 최적화(true 또는 false)
- 테넌시 설정(VPC에서 시작하는 경우, 공유 또는 전용)
- 커널 ID 및 RAM 디스크 ID(해당되는 경우)
- 사용자 데이터(지정된 경우)

- 스팟 (최고) 가격

VPC 서브넷 및 가용성 영역이 식별된 인스턴스에서 Auto Scaling 그룹의 자체 리소스 정의로 복사됩니다.

식별된 인스턴스가 배치 그룹에 있는 경우, 새 Auto Scaling 그룹은 식별된 인스턴스와 동일한 배치 그룹으로 인스턴스를 시작합니다. 시작 구성 설정에서는 배치 그룹을 지정할 수 없으므로 배치 그룹이 새 Auto Scaling 그룹의 `PlacementGroup` 속성에 복사됩니다.

다음 구성 세부 정보는 식별된 인스턴트에서 복사되지 않습니다.

- 스토리지: 블록 디바이스(EBS 볼륨 및 인스턴스 스토어 볼륨)는 식별된 인스턴스에서 복사되지 않습니다. 대신 AMI 생성의 일부로 생성된 블록 디바이스 매핑에 따라 사용되는 디바이스가 결정됩니다.
- 네트워크 인터페이스 수: 식별된 인스턴스에서 네트워크 인터페이스가 복사되지 않습니다. 대신 Amazon EC2 Auto Scaling은 기본 설정을 사용하여 기본 네트워크 인터페이스(eth0)인 하나의 네트워크 인터페이스를 생성합니다.
- 인스턴스 메타데이터 옵션: 액세스 가능한 메타데이터, 메타데이터 버전 및 토큰 응답 홉 제한 설정은 식별된 인스턴스에서 복사되지 않습니다. 대신 Amazon EC2 Auto Scaling은 기본 설정을 사용합니다. 자세한 설명은 인스턴스 메타데이터 옵션 구성 섹션을 참조하세요.
- 로드 밸런서: 식별된 인스턴스가 하나 이상의 로드 밸런서에 등록되어 있으면 로드 밸런서에 대한 정보가 새 Auto Scaling 그룹의 로드 밸런서 또는 대상 그룹 속성에 복사되지 않습니다.
- 태그: 식별된 인스턴스에 태그가 있는 경우, 해당 태그가 새 Auto Scaling 그룹의 `Tags` 속성에 복사되지 않습니다.

사전 조건

EC2 인스턴스는 다음 기준을 충족해야 합니다.

- 인스턴스가 다른 오토 스케일링의 구성원이 아닙니다.
- 인스턴스는 `running` 상태에 있습니다.
- 인스턴스를 시작할 때 사용된 AMI가 항상 있어야 합니다.

EC2 인스턴스()에서 Auto Scaling 그룹 생성

EC2 인스턴스에서 Auto Scaling 그룹을 생성하려면

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

2. 탐색 창의 인스턴스(Instances)에서 인스턴스(Instances)를 선택한 다음 인스턴스를 선택합니다.
3. 작업(Actions), 인스턴스 설정(Instance settings), Auto Scaling 그룹에 연결(Attach to Auto Scaling Group)을 차례로 선택합니다.
4. Auto Scaling 그룹에 연결(Attach to Auto Scaling group) 페이지에서 Auto Scaling 그룹(Auto Scaling Group)에 해당 그룹의 이름을 입력한 다음 연결(Attach)을 선택합니다.

   인스턴스가 연결되면 Auto Scaling 그룹의 일부로 간주됩니다. Auto Scaling 그룹에 대해 지정한 것과 같은 이름과 함께 새 시작 구성을 사용하여 새 Auto Scaling 그룹이 생성됩니다. Auto Scaling 그룹에 원하는 용량이 있고 최대 크기는 1입니다.
5. (옵션) Auto Scaling 그룹의 설정을 편집하려면 탐색 창의 Auto Scaling 아래에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다. 새 Auto Scaling 그룹 옆의 확인란을 선택하고 그룹 목록 위에 있는 편집(Edit) 버튼을 선택하고 필요에 따라 설정을 변경한 다음 업데이트(Update)를 선택합니다.

EC2 인스턴스(AWS CLI)에서 Auto Scaling 그룹 생성

다음 절차에서는 CLI 명령을 사용하여 EC2 인스턴스에서 Auto Scaling 그룹을 생성하는 방법을 보여줍니다.

이 절차에서는 인스턴스를 Auto Scaling 그룹에 추가하지 않습니다. 연결할 인스턴스의 경우, Auto Scaling 그룹이 생성된 후 attach-instances 명령을 실행해야 합니다.

시작하기 전에 Amazon EC2 콘솔 또는 describe-instances 명령을 사용하여 EC2 인스턴스의 ID를 확인합니다.

현재 인스턴스를 템플릿으로 사용하는 방법

- 다음 create-auto-scaling-group명령을 사용하여 EC2 `i-0e69cc3f05f825f4f` 인스턴스에서 Auto Scaling 그룹을 생성합니다. `my-asg-from-instance`

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg-from-instance \
  --instance-id i-0e69cc3f05f825f4f --min-size 1 --max-size 2 --desired-capacity 2
```

Auto Scaling 그룹에서 인스턴스를 시작했는지 확인하려면

- 다음 describe-auto-scaling-groups명령을 사용하여 Auto Scaling 그룹이 성공적으로 생성되었는지 확인합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg-from-instance
```

다음 예의 응답은 해당 그룹의 원하는 용량이 2이고 그룹에 실행 중인 인스턴스가 2개이며 시작 구성의 이름이 `my-asg-from-instance`임을 보여 줍니다.

```
{
  "AutoScalingGroups":[
    {
      "AutoScalingGroupName":"my-asg-from-instance",
      "AutoScalingGroupARN":"arn",
      "LaunchConfigurationName":"my-asg-from-instance",
      "MinSize":1,
      "MaxSize":2,
      "DesiredCapacity":2,
      "DefaultCooldown":300,
      "AvailabilityZones":[
        "us-west-2a"
      ],
      "LoadBalancerNames":[],
      "TargetGroupARNs":[],
      "HealthCheckType":"EC2",
      "HealthCheckGracePeriod":0,
      "Instances":[
        {
          "InstanceId":"i-06905f55584de02da",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        },
        {
          "InstanceId":"i-087b42219468eacde",
          "InstanceType":"t2.micro",
          "AvailabilityZone":"us-west-2a",
          "LifecycleState":"InService",
          "HealthStatus":"Healthy",
          "LaunchConfigurationName":"my-asg-from-instance",
          "ProtectedFromScaleIn":false
        }
```

```
      ],
      "CreatedTime":"2020-10-28T02:39:22.152Z",
      "SuspendedProcesses":[ ],
      "VPCZoneIdentifier":"subnet-6bea5f06",
      "EnabledMetrics":[ ],
      "Tags":[ ],
      "TerminationPolicies":[
        "Default"
      ],
      "NewInstancesProtectedFromScaleIn":false,
      "ServiceLinkedRoleARN":"arn",
      "TrafficSources":[]
    }
  ]
}
```

시작 구성을 보려면

- 다음 describe-launch-configurations명령을 사용하여 시작 구성의 세부 정보를 볼 수 있습니다.

```
aws autoscaling describe-launch-configurations --launch-configuration-names my-asg-
from-instance
```

다음은 예 출력입니다.

```
{
  "LaunchConfigurations":[
    {
      "LaunchConfigurationName":"my-asg-from-instance",
      "LaunchConfigurationARN":"arn",
      "ImageId":"ami-0528a5175983e7f28",
      "KeyName":"my-key-pair-uswest2",
      "SecurityGroups":[
        "sg-05eaec502fcdadc2e"
      ],
      "ClassicLinkVPCSecurityGroups":[ ],
      "UserData":"",
      "InstanceType":"t2.micro",
      "KernelId":"",
      "RamdiskId":"",
      "BlockDeviceMappings":[ ],
      "InstanceMonitoring":{
```

```
            "Enabled":true
          },
          "CreatedTime":"2020-10-28T02:39:22.321Z",
          "EbsOptimized":false,
          "AssociatePublicIpAddress":true
        }
      ]
    }
```

인스턴스를 해지하려면

- 인스턴스가 더 이상 필요하지 않은 경우, 해지할 수 있습니다. terminate-instances 명령을 사용하여 인스턴스 `i-0e69cc3f05f825f4f`를 해지합니다.

  ```
  aws ec2 terminate-instances --instance-ids i-0e69cc3f05f825f4f
  ```

  Amazon EC2 인스턴스를 해지한 후에는 인스턴스를 다시 시작할 수 없습니다. 해지하면 데이터가 사라지므로 해당 볼륨을 인스턴스에 연결할 수 없습니다. 인스턴스 종료에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하십시오.

Auto Scaling 그룹 업데이트

Auto Scaling 그룹의 세부 정보 대부분을 업데이트할 수 있습니다. Auto Scaling 그룹의 이름을 업데이트하거나 를 변경할 수 없습니다 AWS 리전.

Auto Scaling 그룹 업데이트(콘솔)

1. 에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 https://console.aws.amazon.com/ec2/선택합니다.
2. Auto Scaling 그룹을 선택하면 세부 정보, 활동, Auto Scaling, 인스턴스 관리, 모니터링 및 인스턴스 새로 고침 탭과 함께 그룹에 대한 정보가 표시됩니다.
3. 관심 있는 구성 영역의 탭을 선택하고 필요에 따라 설정을 업데이트하십시오. 편집하는 각 설정에 대해 업데이트를 선택하여 Auto Scaling 그룹의 구성에 변경 사항을 저장합니다.

   - 세부 정보 탭

     다음은 Auto Scaling 그룹에 대한 일반적인 설정입니다. Auto Scaling 그룹을 생성할 때와 동일한 방식으로 편집하고 관리할 수 있습니다.

고급 구성 섹션에는 해지 정책, 쿨다운, 일시 중단된 프로세스 및 최대 인스턴스 수명과 같이 그룹을 만들 때 사용할 수 없는 몇 가지 옵션이 있습니다. 또한 Auto Scaling 그룹의 배치 그룹 및 서비스 연결 역할은 볼 수 있지만 편집할 수는 없습니다.

그룹이 Elastic Load Balancing 리소스와 연결되어 있는 경우, 가용 영역을 변경하기 전에 가용 영역 추가을 참조하세요. 로드 밸런서의 일부 제한 사항으로 인해 그룹의 가용 영역 변경 사항을 로드 밸런서의 가용 영역에 적용하지 못할 수도 있습니다.

- 활동 탭
  - 활동 알림 - Amazon SNS 알림
- 자동 규모 조정 탭
  - 동적 조정 정책 - 동적 조정 정책
  - 예측 조정 정책 - 예측 조정 정책
  - 예약된 작업 - 예약된 작업
- 인스턴스 관리 탭
  - 수명 주기 후크 - 수명 주기 후크
  - 웜 풀 - 웜 풀
- 모니터링 탭
  - 이 탭에는 CloudWatch 그룹 지표 수집을 활성화하거나 비활성화할 수 있는 단일 옵션만 있습니다.

명령행을 사용하여 Auto Scaling 그룹을 업데이트하려면

다음 명령 중 하나를 사용할 수 있습니다:

- update-auto-scaling-group (AWS CLI)
- 업데이트-ASAutoScalingGroup (AWS Tools for Windows PowerShell)

Auto Scaling 인스턴스 업데이트

새 시작 템플릿 또는 시작 구성을 Auto Scaling 그룹과 연결하면 모든 새 인스턴스에 업데이트된 구성이 적용됩니다. 기존 인스턴스는 원래 시작된 구성으로 계속 실행됩니다. 기존 인스턴스에 변경 사항을 적용하려면 다음과 같은 옵션이 있습니다.

- 인스턴스 새로 고침을 시작하여 이전 인스턴스를 교체합니다. 자세한 내용은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다. 단원을 참조하십시오.
- 해지 정책에 근거하여 스케일링 활동이 이전 인스턴스를 새 인스턴스로 점진적으로 교체하도록 기다리십시오.
- Auto Scaling 그룹으로 교체되도록 수동으로 해지합니다.

> **Note**
>
> 다음 인스턴스 속성을 시작 템플릿 또는 시작 구성의 일부로 지정하여 변경할 수 있습니다.
>
> - Amazon Machine 이미지(AMI)
> - 블록 디바이스
> - 키 페어
> - 인스턴스 유형
> - 보안 그룹
> - 사용자 데이터
> - 모니터링
> - IAM 인스턴스 프로파일
> - 배치 테넌시
> - kernel
> - ramdisk
> - 인스턴스에 공용 IP 주소가 있는지 여부

Auto Scaling 그룹 및 인스턴스에 태그 지정

태그는 사용자가 할당하거나 AWS 리소스에 AWS 할당하는 사용자 지정 속성 레이블입니다. 각 태그에는 다음 두 가지 부분이 있습니다.

- 태그 키(예: `costcenter`, `environment` 또는 `project`)
- 태그 값(예: `111122223333` 또는 `production`)으로 알려진 선택적 필드

태그는 다음을 지원합니다.

- AWS 비용을 추적하세요. AWS Billing and Cost Management 대시보드에서 이러한 태그를 활성화합니다. AWS 는 태그를 사용하여 비용을 분류하고 월별 비용 할당 보고서를 제공합니다. 자세한 설명은 AWS Billing 사용자 가이드에서 비용 할당 태그 사용을 참조하세요.
- 태그에 근거하여 Auto Scaling 그룹에 대한 액세스를 제어합니다. IAM 정책의 조건을 사용하여 해당 그룹의 태그를 기반으로 Auto Scaling 그룹에 대한 액세스를 제어할 수 있습니다. 자세한 내용은 보안을 위한 태그 단원을 참조하십시오.
- 추가하는 태그를 기준으로 Auto Scaling 그룹을 필터링하고 검색합니다. 자세한 내용은 태그를 사용하여 Auto Scaling 그룹 필터링 단원을 참조하십시오.
- AWS 리소스를 식별하고 구성합니다. 많은 AWS 서비스 지원 태깅이 있으므로 서로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 연결되어 있음을 나타낼 수 있습니다.

신규 또는 기존 Auto Scaling 그룹에 태그를 지정할 수 있습니다. Auto Scaling 그룹의 태그를 시작하는 EC2 인스턴스로 전파할 수도 있습니다.

태그는 Amazon EBS 볼륨에 전파되지 않습니다. Amazon EBS 볼륨에 태그를 추가하려면 시작 템플릿에서 태그를 지정합니다. 자세한 내용은 Auto Scaling 그룹에 대한 시작 템플릿 생성 단원을 참조하십시오.

AWS Management Console AWS CLI, 또는 를 통해 태그를 생성하고 관리할 수 있습니다SDKs.

내용

- 태그 이름 지정 및 사용 제한
- EC2 인스턴스 태그 지정 수명 주기
- Auto Scaling 그룹에 태그 지정
- 태그 삭제
- 보안을 위한 태그
- 태그에 대한 액세스 통제
- 태그를 사용하여 Auto Scaling 그룹 필터링

태그 이름 지정 및 사용 제한

태그에 적용되는 기본 제한은 다음과 같습니다.

- 리소스당 최대 태그 수는 50개입니다.
- 단일 호출을 사용하여 추가하거나 제거할 수 있는 최대 태그 수는 25입니다.

- 최대 키 길이는 유니코드 문자 128자입니다.
- 최대 값 길이는 유니코드 문자 256자입니다.
- 태그 키와 값은 대/소문자를 구분합니다. 태그를 대문자로 사용하는 전략을 세우고 이러한 전략을 모든 리소스 유형에 대해 일관되게 구현하는 것이 가장 좋습니다.
- 태그 이름 또는 값에 `aws:` 접두사를 사용하지 마세요. 이 접두사는 AWS 사용을 위해 예약되어 있기 때문입니다. 이 접두사를 가진 태그 이름이나 값은 편집 또는 삭제할 수 없으며 리소스 할당량당 태그로도 계수되지 않습니다.

EC2 인스턴스 태그 지정 수명 주기

EC2 인스턴스에 태그를 전파하기로 선택한 경우 태그는 다음과 같이 관리됩니다.

- Auto Scaling 그룹에서 인스턴스를 시작하면 리소스를 생성한 후가 아니라 리소스를 생성하는 동안 인스턴스에 태그를 추가합니다.
- Auto Scaling 그룹은 `aws:autoscaling:groupName` 키와 Auto Scaling 그룹 명칭 값을 사용하여 인스턴스에 태그를 자동으로 추가합니다.
- 시작 템플릿에 인스턴스 태그를 지정하고 그룹의 태그를 해당 인스턴스로 전파하도록 선택한 경우, 모든 태그가 병합됩니다. 시작 템플릿의 태그와 Auto Scaling 그룹의 태그에 대해 동일한 태그 키가 지정된 경우, 그룹의 태그 값이 우선합니다.
- 기존 인스턴스를 연결하면 Auto Scaling 그룹이 인스턴스에 태그를 추가하여 동일한 태그 키를 가진 기존 태그를 덮어씁니다. 또한 `aws:autoscaling:groupName` 키와 Auto Scaling 그룹 명칭 값을 사용하여 태그를 추가합니다.
- Auto Scaling 그룹에서 인스턴스를 분리하면 `aws:autoscaling:groupName` 태그만 제거됩니다.

Auto Scaling 그룹에 태그 지정

Auto Scaling 그룹에 태그를 추가할 때 Auto Scaling 그룹에서 시작되는 인스턴스에 추가할지를 지정할 수 있습니다. 태그를 수정하면 변경 후 태그의 업데이트 버전이 Auto Scaling 그룹에서 시작되는 인스턴스에 추가됩니다. Auto Scaling 그룹의 태그를 생성하거나 수정하는 경우, 이러한 변경 사항은 Auto Scaling 그룹에서 이미 실행 중인 인스턴스에는 적용되지 않습니다.

내용

- 태그 추가 또는 수정(콘솔)
- 태그 추가 또는 수정(AWS CLI)

태그 추가 또는 수정(콘솔)

생성 시 Auto Scaling 그룹에 태그를 지정하려면

Amazon EC2 콘솔을 사용하여 Auto Scaling 그룹을 생성할 때 Auto Scaling 그룹 생성 마법사의 태그 추가 페이지에서 태그 키와 값을 지정할 수 있습니다. Auto Scaling 그룹에서 시작된 인스턴스에 태그를 전파하려면 선택된 태그에 대한 새 인스턴스에 태그 지정(Tag new instances) 옵션을 유지해야 합니다. 그렇지 않은 경우, 선택을 취소할 수 있습니다.

기존 Auto Scaling 그룹의 태그를 추가하거나 수정하려면

1. 에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 https://console.aws.amazon.com/ec2/선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹(Auto Scaling groups) 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 태그, 편집을 선택합니다.
4. 기존의 태그를 수정하려면 Key 및 Value를 수정합니다.
5. 새 태그를 추가하려면 Add tag를 선택하고 Key 및 Value를 편집합니다. Auto Scaling 그룹에서 시작된 인스턴스에 태그를 자동으로 추가하려면 새 인스턴스에 태그 지정(Tag new instances)을 선택한 채로 두고 그러지 않으려는 경우, 이 옵션을 선택 취소할 수 있습니다.
6. 태그 추가를 마쳤으면 저장을 선택합니다.

태그 추가 또는 수정(AWS CLI)

다음 예제에서는 Auto Scaling 그룹을 생성할 때 AWS CLI 를 사용하여 태그를 추가하고 기존 Auto Scaling 그룹에 대한 태그를 추가하거나 수정하는 방법을 보여줍니다.

생성 시 Auto Scaling 그룹에 태그를 지정하려면

create-auto-scaling-group 명령을 사용하여 새 Auto Scaling 그룹을 생성하고 **`environment=production`** Auto Scaling 그룹에 와 같은 태그를 추가합니다. 태그는 Auto Scaling 그룹에서 시작되는 모든 인스턴스에도 추가됩니다.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-configuration-name my-launch-config --min-size 1 --max-size 3 \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --tags Key=environment,Value=production,PropagateAtLaunch=true
```

기존 Auto Scaling 그룹의 태그를 추가하거나 수정하려면

create-or-update-tags 명령을 사용하여 태그를 생성하거나 수정합니다. 예컨대, 다음 명령은 **Name=my-asg** 및 **costcenter=cc123** 태그를 추가합니다. 태그는 이 변경 후에 Auto Scaling 그룹에서 시작되는 모든 인스턴스에도 추가됩니다. 이 키를 가진 태그가 이미 있으면 기존 태그가 교체됩니다. Amazon EC2 콘솔은 각 인스턴스의 표시 이름을 Name 키에 지정된 이름(대소문자 구분)과 연결합니다.

```
aws autoscaling create-or-update-tags \
  --tags ResourceId=my-asg,ResourceType=auto-scaling-group,Key=Name,Value=my-
asg,PropagateAtLaunch=true \
  ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=costcenter,Value=cc123,PropagateAtLaunch=true
```

Auto Scaling 그룹의 태그 설명(AWS CLI)

특정 Auto Scaling 그룹에 적용된 태그를 보려는 경우, 다음 명령 중 하나를 사용할 수 있습니다:

- describe-tags - Auto Scaling 그룹 이름을 제공하여 지정된 그룹의 태그 목록을 볼 수 있습니다.

  ```
  aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
  ```

  다음은 응답의 예입니다.

  ```
  {
      "Tags": [
          {
              "ResourceType": "auto-scaling-group",
              "ResourceId": "my-asg",
              "PropagateAtLaunch": true,
              "Value": "production",
              "Key": "environment"
          }
      ]
  }
  ```

- describe-auto-scaling-groups - 태그를 포함하여 지정된 그룹의 속성을 볼 수 있도록 Auto Scaling 그룹 이름을 제공합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

다음은 응답의 예입니다.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 1,
            ...
            "Tags": [
                {
                    "ResourceType": "auto-scaling-group",
                    "ResourceId": "my-asg",
                    "PropagateAtLaunch": true,
                    "Value": "production",
                    "Key": "environment"
                }
            ],
            ...
        }
    ]
}
```

태그 삭제

Auto Scaling 그룹과 연결된 태그는 언제든지 삭제할 수 있습니다.

내용

- 태그 삭제(콘솔)
- 태그 삭제(AWS CLI)

태그 삭제(콘솔)

태그를 삭제하려면

1. 에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 https://console.aws.amazon.com/ec2/선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹(Auto Scaling groups) 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 태그, 편집을 선택합니다.
4. 태그 옆에 있는 제거를 선택합니다.
5. 업데이트를 선택합니다.

태그 삭제(AWS CLI)

delete-tags 명령을 사용하여 태그를 삭제합니다. 예컨대, 다음 명령은 키가 **environment**인 태그를 삭제합니다.

```
aws autoscaling delete-tags --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=environment"
```

태그 키는 지정해야 하지만 값은 지정하지 않아도 됩니다. 값을 지정했는데 해당 값이 잘못된 경우, 태그가 삭제되지 않습니다.

보안을 위한 태그

태그를 사용하여 요청자(예: IAM 사용자 또는 역할)가 특정 Auto Scaling 그룹을 생성, 수정 또는 삭제할 권한이 있는지 확인합니다. 다음 조건 키 중 하나 이상을 사용하여 IAM 정책의 조건 요소에 태그 정보를 제공합니다.

- 특정 태그가 있는 Auto Scaling 그룹에 대한 사용자 작업을 허용(또는 거부)하려면 `autoscaling:ResourceTag/tag-key: tag-value`를 사용합니다.
- 요청에 특정 태그가 존재하도록 (또는 존재하지 않도록) 요구하려면 `aws:RequestTag/tag-key: tag-value`를 사용합니다.
- 요청에 특정 태그 키가 존재하도록 (또는 존재하지 않도록) 요구하려면 `aws:TagKeys [tag-key, ...]`를 사용합니다.

예컨대, 다음 예에 표시된 것처럼 키가 **environment**이고 값이 **production**인 태그가 포함된 모든 Auto Scaling 그룹에 대한 액세스를 거부할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"autoscaling:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

조건 키를 사용하여 Auto Scaling 그룹에 대한 액세스를 제어하는 방법에 대한 자세한 설명은 Amazon EC2 Auto Scaling과 함께 작동하는 방식 IAM을 참조하세요.

태그에 대한 액세스 통제

태그를 사용하여 요청자(예: IAM 사용자 또는 역할)가 Auto Scaling 그룹에 대한 태그를 추가, 수정 또는 삭제할 권한이 있는지 확인합니다.

다음 예제 IAM 정책은 보안 주체에게 Auto Scaling 그룹에서 **temporary** 키가 있는 태그만 제거할 수 있는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "autoscaling:DeleteTags",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": { "aws:TagKeys": ["temporary"] }
            }
```

```
        }
    ]
}
```

Auto Scaling 그룹에 지정된 태그에 제약 조건을 적용하는 IAM 정책의 자세한 예는 섹션을 참조하세요사용할 수 있는 태그 키 및 태그 값 제어.

> **Note**
>
> 사용자가 Auto Scaling 그룹에서 태그 지정 또는 태그 해제 작업을 수행하지 못하도록 제한하는 정책을 설정해도 인스턴스를 시작한 후에 수동으로 인스턴스의 태그를 변경하는 것을 막을 수 없습니다. EC2 인스턴스의 태그에 대한 액세스를 제어하는 예제는 Amazon EC2 사용 설명서의 예제: 리소스 태그 지정을 참조하세요.

태그를 사용하여 Auto Scaling 그룹 필터링

다음 예제에서는 describe-auto-scaling-groups 명령과 함께 필터를 사용하여 특정 태그가 있는 Auto Scaling 그룹을 설명하는 방법을 보여줍니다. 태그별 필터링은 AWS CLI 또는 로 제한SDK되며 콘솔에서는 사용할 수 없습니다.

필터링 고려 사항

- 단일 요청에서 여러 필터와 여러 필터 값을 지정할 수 있습니다.
- 또한 필터 값과 함께 와일드카드를 사용할 수 없습니다.
- 필터 값은 대/소문자를 구분합니다.

예: 특정 태그 키 및 값 페어로 Auto Scaling 그룹 설명

다음 명령은 **environment=production**의 태그 키 및 값 페어가 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-value,Values=production
```

다음은 응답의 예입니다.

```
{
    "AutoScalingGroups": [
```

```
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-0b97f1e282EXAMPLE",
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Latest"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 1,
            ...
            "Tags": [
                {
                    "ResourceType": "auto-scaling-group",
                    "ResourceId": "my-asg",
                    "PropagateAtLaunch": true,
                    "Value": "production",
                    "Key": "environment"
                }
            ],
            ...
        },

    ... additional groups ...

    ]
}
```

또는 `tag:<key>` 필터를 사용하여 태그를 지정할 수 있습니다. 예컨대, 다음 명령은 **environment=production**의 태그 키 및 값 페어가 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다. 이 필터의 형식은 `Name=tag:<key>,Values=<value>`이며 **<key>** 및 **<value>**는 태그 키 및 값 페어를 나타냅니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production
```

`--query` 옵션을 사용하여 AWS CLI 출력을 필터링할 수도 있습니다. 다음 예제에서는 이전 명령의 AWS CLI 출력을 그룹 이름, 최소 크기, 최대 크기 및 원하는 용량 속성으로 제한하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
```

```
  --filters Name=tag:environment,Values=production \
  --query "AutoScalingGroups[].{AutoScalingGroupName: AutoScalingGroupName, MinSize:
 MinSize, MaxSize: MaxSize, DesiredCapacity: DesiredCapacity}"
```

다음은 응답의 예입니다.

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "MinSize": 0,
        "MaxSize": 10,
        "DesiredCapacity": 1
    },

    ... additional groups ...

]
```

필터링에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 출력 필터링을 참조하세요.

예: 지정된 태그 키와 일치하는 태그가 있는 Auto Scaling 그룹 설명

다음 명령은 태그 값에 관계없이 **environment** 태그가 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment
```

예: 지정된 태그 키 집합과 일치하는 태그가 있는 Auto Scaling 그룹 설명

다음 명령은 태그 값에 관계없이 **environment** 및 **project**에 대한 태그가 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment Name=tag-key,Values=project
```

예: 지정된 태그 키 중 하나 이상과 일치하는 태그가 있는 Auto Scaling 그룹 설명

다음 명령은 태그 값에 관계없이 **environment** 또는 **project**에 대한 태그가 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-key,Values=environment,project
```

예: 지정된 태그 값이 있는 Auto Scaling 그룹 설명

다음 명령은 태그 키에 관계없이 **production**의 태그 값이 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production
```

예: 지정된 태그 값 집합이 있는 Auto Scaling 그룹 설명

다음 명령은 태그 키에 관계없이 태그 값 **production** 및 **development**가 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production Name=tag-value,Values=development
```

예: 지정된 태그 값 중 하나 이상과 일치하는 태그가 있는 Auto Scaling 그룹 설명

다음 명령은 태그 키에 관계없이 **production** 또는 **development**의 태그 값이 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag-value,Values=production,development
```

예: 여러 태그 키 및 값과 일치하는 태그가 있는 Auto Scaling 그룹 설명

필터를 결합하여 사용자 지정을 생성할 수도 있습니다.AND 그리고 OR 더 복잡한 필터링을 수행하는 로직입니다.

다음 명령은 특정 태그 집합이 있는 Auto Scaling 그룹만 표시하도록 결과를 필터링하는 방법을 보여줍니다. 태그 키 1개는 **environment** AND 태그 값은 (**production** OR **development**) AND 다른 태그 키는 **costcenter** AND 태그 값은 입니다**cc123**.

```
aws autoscaling describe-auto-scaling-groups \
  --filters Name=tag:environment,Values=production,development
 Name=tag:costcenter,Values=cc123
```

인스턴스 유지 관리 정책

인스턴스 새로 고침 또는 상태 확인 프로세스와 같이 인스턴스 교체를 유발하는 이벤트 발생 중에 특정 용량 요구 사항을 충족하도록 Auto Scaling 그룹에 대한 인스턴스 유지 관리 정책을 구성할 수 있습니다.

예를 들어, 적은 수의 인스턴스가 있는 Auto Scaling 그룹이 있다고 가정합니다. 상태 점검 결과 손상된 인스턴스가 발견될 경우 인스턴스가 종료되고 교체되어 발생할 수 있는 중단을 방지하는 것이 좋습니다. 인스턴스 유지 관리 정책을 사용하면 Amazon EC2 Auto Scaling에서 먼저 새 인스턴스를 시작한 다음 완전히 준비될 때까지 기다렸다가 비정상 인스턴스를 종료하도록 할 수 있습니다.

또한 인스턴스 유지 관리 정책은 여러 인스턴스가 동시에 교체되는 경우 발생할 수 있는 중단을 최소화하는 데도 도움이 됩니다. 정책의 최소 및 최대 정상 백분율 파라미터를 설정하면 Auto Scaling 그룹에서 인스턴스를 교체할 때 해당 최소-최대 범위 내에서만 용량을 늘리거나 줄일 수 있습니다. 범위가 클수록 동시에 교체될 수 있는 인스턴스 수가 늘어납니다.

내용

- Auto Scaling 그룹의 인스턴스 유지 관리 정책
- Auto Scaling 그룹에서 인스턴스 유지 관리 정책 설정

Auto Scaling 그룹의 인스턴스 유지 관리 정책

이 항목에서는 사용 가능한 옵션에 대한 개요와 인스턴스 유지 관리 정책을 생성할 때 고려해야 할 사항을 설명합니다.

내용

- 개요
- 핵심 개념
- 인스턴스 워밍업
- 건전성 체크 유예 기간
- Auto Scaling 그룹 조정
- 예제 시나리오

개요

Auto Scaling 그룹에 대한 인스턴스 유지 관리 정책을 생성하면 정책이 인스턴스가 교체되는 Amazon EC2 Auto Scaling 이벤트에 영향을 줍니다. 그 결과 동일한 Auto Scaling 그룹 내에서 보다 일관된 교체 동작이 가능합니다. 또한 필요에 따라 그룹의 가용성 또는 비용을 최적화할 수 있습니다.

콘솔에서는 다음과 같은 구성 옵션을 사용할 수 있습니다.

- 종료 전 시작 - 기존 인스턴스를 종료하기 전에 새 인스턴스를 프로비저닝해야 합니다. 이 접근 방식은 비용 절감보다 가용성을 선호하는 애플리케이션에 적합합니다.
- 종료 및 시작 - 기존 인스턴스가 종료되는 동시에 새 인스턴스가 프로비저닝됩니다. 이 접근 방식은 가용성보다 비용 절감을 선호하는 애플리케이션에 적합합니다. 또한 인스턴스를 교체하더라도 현재 사용 가능한 용량보다 많은 용량을 실행하지 않아야 하는 애플리케이션에도 적합합니다.
- 사용자 지정 정책 - 이 옵션을 사용하면 인스턴스 교체 시 사용할 수 있는 용량에 대한 사용자 지정 최소 및 최대 범위로 정책을 설정할 수 있습니다. 이 접근 방식은 비용과 가용성의 균형을 적절하게 조정하는 데 도움이 될 수 있습니다.

Auto Scaling 그룹의 기본값은 인스턴스 유지 관리 정책을 사용하지 않는 것인데, 이렇게 하면 기본 동작으로 인스턴스 유지 관리 이벤트에 응답합니다. 기본 동작은 다음 표에 설명되어 있습니다.

인스턴스 유지 관리 이벤트 기본 동작

이벤트	설명	기본 동작
상태 확인 실패	인스턴스가 상태 확인에 실패할 때 자동으로 발생합니다. Amazon EC2 Auto Scaling은 상태 확인에 실패한 인스턴스를 교체합니다. 상태 확인 실패의 원인을 이해하려면 Auto Scaling 그룹의 인스턴스에 대한 상태 확인(을)를 참조하세요.	종료 및 시작
인스턴스 새로 고침	인스턴스 새로 고침을 시작할 때 발생합니다. 구성에 따라 인스턴스 새로 고침은 인스턴스를 한 번에 하나씩, 한 번에 여	종료 및 시작

이벤트	설명	기본 동작
	러 개 또는 한 번에 모두 교체할 수 있습니다. 자세한 내용은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다.(을)를 참조하세요.	
최대 인스턴스 수명	인스턴스가 Auto Scaling 그룹에 지정한 최대 인스턴스 수명에 도달할 때 자동으로 발생합니다. Amazon EC2 Auto Scaling은 최대 인스턴스 수명에 도달한 인스턴스를 교체합니다. 자세한 내용은 최대 인스턴스 수명을 기준으로 Auto Scaling 인스턴스 교체(을)를 참조하세요.	종료 및 시작

이벤트	설명	기본 동작
재분배	그룹의 불균형을 초래하는 근본적인 변화가 있는 경우 자동으로 발생합니다. Amazon EC2 Auto Scaling은 다음과 같은 상황에서 그룹의 균형을 재조정합니다. • 이전에 용량이 부족했던 가용 영역이 복구되거나 그룹에서 가용 영역을 추가 또는 삭제합니다. 이 경우 Auto Scaling 그룹은 가용 영역 간에 균형을 유지하려고 합니다. 자세한 내용은 재조정 활동(을)를 참조하세요. • Auto Scaling 그룹에서 용량 재분배를 활성화하면 스팟 인스턴스의 가용성이 변경되어 기존 스팟 인스턴스가 중단되기 전에 새 스팟 인스턴스를 시작하려고 합니다. 자세한 내용은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리(을)를 참조하세요. • Auto Scaling 그룹을 업데이트하면 혼합 인스턴스 정책을 업데이트할 때 선택한 새 구매 옵션에 맞게 인스턴스가 점진적으로 교체됩니다. 자세한 내용은 Auto Scaling 그룹 업데이트(을)를 참조하세요.	종료 전 시작 Amazon EC2 Auto Scaling은 그룹의 크기 제한을 최대 용량의 10%까지 초과할 수 있습니다. 하지만, 용량 재분배를 사용하는 경우 원하는 용량의 최대 10%까지만 이러한 제한을 초과할 수 있습니다.

Amazon EC2 Auto Scaling은 다음과 같은 상황에서 계속해서 기본적으로 종료되고 시작됩니다. 따라서, 이러한 상황 중 하나가 발생하면 그룹 용량이 인스턴스 유지 관리 정책의 하한 임계값보다 적을 수 있습니다.

- 예를 들어, 사람의 행동으로 인해 인스턴스가 예기치 않게 종료되는 경우는 다음과 같습니다. Amazon EC2 Auto Scaling에서 더 이상 실행되지 않는 인스턴스를 즉시 교체합니다. 자세한 내용은 아마존 EC2 헬스 체크(을)를 참조하세요.
- Amazon EC2 Auto Scaling에서 대체 인스턴스를 시작하기 전에 예정된 이벤트의 일환으로 Amazon EC2가 인스턴스를 재부팅, 중지 또는 폐기하는 경우. 이러한 이벤트에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 예약 이벤트를 참조하십시오.
- Amazon EC2 스팟 서비스가 스팟 인스턴스 중단을 시작하고 스팟 인스턴스가 강제로 종료되는 경우는 다음과 같습니다.

스팟 인스턴스와 관련하여 Auto Scaling 그룹에서 용량 재분배를 활성화한 경우 스팟 중단을 시작하기 전에 시작한 다른 스팟 풀의 보류 중인 인스턴스가 인스턴스에 이미 있을 수 있습니다. 용량 재분배 방법에 대한 자세한 내용은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리(을)를 참조하세요.

그러나, 스팟 인스턴스는 사용 가능한 상태로 유지될 수 없으며 2분 스팟 인스턴스 중단 알림과 함께 종료될 수 있으므로 새 인스턴스가 시작되기 전에 인스턴스가 중단되면 인스턴스 유지 관리 정책의 하한 임계값이 초과될 수 있습니다.

핵심 개념

시작하기 전에 다음과 같은 핵심 개념 및 용어를 익힙니다.

원하는 용량

원하는 용량은 생성 시 Auto Scaling 그룹의 용량입니다. 또한 그룹에 연결된 조정 조건이 없을 때 그룹이 유지하려고 시도하는 용량이기도 합니다.

인스턴스 유지 관리 정책

인스턴스 유지 관리 정책은 기존 인스턴스가 종료되기 전에 인스턴스 유지 관리 이벤트를 위해 인스턴스를 먼저 프로비저닝할지 여부를 제어합니다. 또한 Auto Scaling 그룹이 동시에 여러 인스턴스를 교체할 때 원하는 용량보다 낮거나 초과할 수 있는 범위도 결정합니다.

최대 정상 백분율

최대 정상 백분율은 Auto Scaling 그룹이 인스턴스를 교체할 때 늘릴 수 있는 원하는 용량의 백분율입니다. 이는 워크로드를 지원하기 위해 서비스 중이고 정상 상태이거나 보류 중일 수 있는 그룹의

최대 비율을 나타냅니다. 콘솔에서는 종료 전 시작 옵션 또는 사용자 지정 정책 옵션을 사용할 때 최대 정상 비율을 설정할 수 있습니다. 유효한 값은 100~200%입니다.

최소 건전 백분율

최소 정상 백분율은 인스턴스를 교체할 때 서비스 상태를 유지하고 워크로드 지원에 사용할 준비가 된 상태로 유지하기 위해 필요한 용량의 비율입니다. 첫 번째 상태 점검을 성공적으로 완료하고 지정된 워밍업 시간이 경과하면 인스턴스가 정상으로 간주되어 사용할 준비가 된 것으로 간주됩니다. 콘솔에서는 종료 및 시작 옵션 또는 사용자 지정 정책 옵션을 사용할 때 최소 정상 비율을 설정할 수 있습니다. 유효한 값은 0~100%입니다.

> **Note**
>
> 인스턴스를 더 빨리 교체하려면 최소 정상 비율을 낮게 지정할 수 있습니다. 하지만, 실행 중인 정상 인스턴스가 충분하지 않으면 가용성이 저하될 수 있습니다. 여러 인스턴스가 교체되는 상황에서 가용성을 유지하기 위해 적절한 값을 선택하는 것이 좋습니다.

인스턴스 워밍업

`InService` 상태가 된 후 인스턴스를 초기화하는 데 시간이 필요한 경우 Auto Scaling 그룹의 기본 인스턴스 워밍업을 활성화합니다. 기본 인스턴스 워밍업을 사용하면 인스턴스가 준비되기 전에 최소 정상 백분율에 포함되는 것을 방지할 수 있습니다. 따라서, Amazon EC2 Auto Scaling은 기존 인스턴스를 종료하기 전에 워크로드를 지원할 충분한 용량을 확보하는 데 걸리는 시간을 고려할 수 있습니다.

추가 혜택으로, 기본 인스턴스 워밍업을 활성화하면 동적 조정에 사용되는 Amazon CloudWatch 지표를 개선할 수 있습니다. Auto Scaling 그룹에 조정 정책이 있는 경우 그룹이 축소될 때 동일한 기본 준비 기간을 사용하여 초기화가 완료되기 전에 인스턴스가 CloudWatch 측정치에 포함되는 것을 방지합니다.

자세한 정보는 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정을 참조하세요.

건전성 체크 유예 기간

Amazon EC2 Auto Scaling은 Auto Scaling 그룹이 사용하는 건전성 체크의 상태에 따라 인스턴스가 정상인지 여부를 결정합니다. 자세한 설명은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 섹션을 참조하세요.

건전성 체크가 최대한 빨리 시작되게 하려면 그룹의 건전성 체크 유예 기간을 너무 길게 설정하지 마세요. 하지만 Elastic Load Balancing 건전성 체크에서 대상이 요청을 처리할 수 있는지 확인할 수 있을 만큼 길게 설정하세요. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정(을)를 참조하세요.

Auto Scaling 그룹 조정

인스턴스 유지 관리 정책은 인스턴스 유지 관리 이벤트에만 적용되며 그룹을 수동 또는 자동으로 조정하는 것을 방지하지 못합니다.

Auto Scaling 그룹에 연결된 조정 정책 또는 예약된 작업이 있으면 인스턴스 유지 관리 이벤트가 발생하는 동안 병렬로 실행할 수 있습니다. 이 경우 사용자가 정의한 조정 한도 내에서만 그룹의 원하는 용량을 늘리거나 줄일 수 있습니다. 한도에 대한 자세한 내용은 Auto Scaling 그룹에 대한 스케일링 제한 설정(을)를 참조하세요.

예제 시나리오

일반적인 시나리오에서 인스턴스 유지 관리 정책과 원하는 용량은 다음과 같을 수 있습니다.

- 최소 정상 백분율 = 90%
- 최대 정상 백분율 = 120%
- 원하는 용량 = 100

인스턴스 유지 관리 이벤트 중에 Auto Scaling 그룹에는 적게는 90개에서 많게는 120개의 인스턴스가 있을 수 있습니다. 이벤트가 끝나면 그룹은 다시 100개의 인스턴스를 보유하게 됩니다.

웜 풀이 있는 Auto Scaling 그룹에서 인스턴스 유지 관리 정책을 사용하는 경우 최소 및 최대 정상 비율이 Auto Scaling 그룹과 웜 풀에 각각 적용됩니다.

예를 들어, 다음과 같은 구성을 가정해 보겠습니다.

- 최소 정상 백분율 = 90%
- 최대 정상 백분율 = 120%
- 원하는 용량 = 100
- 웜 풀 크기 = 10

그룹의 인스턴스를 재활용하기 위해 인스턴스 새로 고침을 시작하는 경우 Amazon EC2 Auto Scaling은 Auto Scaling 그룹의 인스턴스를 먼저 교체한 다음 웜 풀의 인스턴스를 교체합니다. Amazon EC2 Auto Scaling은 아직 Auto Scaling 그룹의 인스턴스를 교체하는 작업을 진행 중이지만 그룹에 적게는 90개에서 많게는 120개의 인스턴스가 있을 수 있습니다. Amazon EC2 Auto Scaling은 그룹 작업을 완료한 후 웜 풀의 인스턴스를 교체하는 작업을 수행할 수 있습니다. 이 경우 웜 풀에는 최소 9개에서 최대 12개의 인스턴스가 있을 수 있습니다.

Auto Scaling 그룹에서 인스턴스 유지 관리 정책 설정

Auto Scaling 그룹을 만들 때 인스턴스 유지 관리 정책을 생성할 수 있습니다. 기존 그룹에서도 생성할 수 있습니다.

Auto Scaling 그룹에서 인스턴스 유지 관리 정책을 설정하면 인스턴스 유지 관리 정책을 재정의하지 않는 한 인스턴스 새로 고침 기능에 대한 최소 및 최대 정상 백분율 파라미터 값을 더 이상 지정할 필요가 없습니다.

콘솔에서는 Amazon EC2 Auto Scaling에서 제공하는 옵션이 시작하는 데 도움이 됩니다.

내용

- 인스턴스 유지 관리 정책 설정
- 인스턴스 유지 관리 정책 제거

인스턴스 유지 관리 정책 설정

Auto Scaling 그룹에서 인스턴스 유지 관리 정책을 설정하려면 다음 방법 중 하나를 사용합니다.

Console

새 그룹에 인스턴스 유지 관리 정책을 설정하려면(콘솔)

1. 시작 템플릿을 사용하여 Auto Scaling 그룹 생성의 지침에 따라 절차의 각 단계(최대 11단계)를 완료합니다.
2. 그룹 크기 및 조정 정책 구성의 원하는 용량에 시작할 인스턴스의 초기 수를 입력합니다.
3. 조정 섹션의 조정 제한에서 원하는 용량의 새 값이 최소 희망 용량및 최대 희망 용량보다 큰 경우 원하는 최대 용량이 원하는 새 용량 값으로 자동 증가합니다. 필요에 따라 이러한 제한을 변경할 수 있습니다.
4. 자동 조정의 경우, 대상 추적 조정 정책을 생성할지 여부를 선택합니다. Auto Scaling 그룹을 생성한 후에 이 정책을 생성할 수도 있습니다.

   대상 추적 조정 정책을 선택하는 경우 대상 추적 조정 정책 생성의 지침에 따라 정책을 생성합니다.
5. 인스턴스 유지 관리 정책 섹션에서 사용 가능한 옵션 중 하나를 선택합니다.

   - 종료 전 시작: 기존 인스턴스를 종료하기 전에 새 인스턴스를 프로비저닝해야 합니다. 이 방식은 비용 절감보다 가용성을 선호하는 경우에 적합합니다.

- 해지 후 시작: 기존 인스턴스가 해지되는 것과 동시에 새 인스턴스가 시작됩니다. 이 방식은 가용성보다 비용 절감을 선호하는 경우에 적합합니다. 또한 현재 사용 가능한 용량보다 많은 용량을 실행하지 않아야 하는 애플리케이션에도 적합합니다.
- 사용자 지정 정책: 이 옵션을 사용하면 인스턴스 교체 시 사용할 수 있는 용량에 대한 사용자 지정 최소 및 최대 범위로 정책을 설정할 수 있습니다. 이는 비용과 가용성의 균형을 적절하게 조정하는 데 도움이 될 수 있습니다.

6. 정상 백분율 설정의 경우, 다음 필드 중 하나 또는 둘 다에 값을 입력합니다. 활성화된 필드는 이전 단계에서 선택한 옵션에 따라 달라집니다.
   - 최소: 인스턴스 교체를 진행하는 데 필요한 최소 정상 비율을 설정합니다.
   - 최대: 인스턴스 교체 시 가능한 최대 정상 비율을 설정합니다.
7. 원하는 용량을 기준으로 교체 중 용량 보기 섹션을 확장하여 최소 및 최대 값이 그룹에 어떻게 적용되는지 확인하세요. 사용되는 정확한 값은 원하는 용량 값에 따라 달라지며, 이 값은 그룹이 조정되면 변경됩니다.
8. 시작 템플릿을 사용하여 Auto Scaling 그룹 생성의 단계를 계속합니다.

AWS CLI

새 그룹에 인스턴스 유지 관리 정책을 설정하려면(AWS CLI)

명령에 `--instance-maintenance-policy` 옵션을 추가합니다. create-auto-scaling-group 다음 예제에서는 이름이 *my-asg*로 지정된 새 Auto Scaling 그룹에서 인스턴스 유지 관리 정책을 설정합니다.

```
aws autoscaling create-auto-scaling-group \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --auto-scaling-group-name my-asg \
  --min-size 1 \
  --max-size 10 \
  --desired-capacity 5 \
  --default-instance-warmup 20 \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": 90,
      "MaxHealthyPercentage": 120
    }' \
  --vpc-zone-identifier "subnet-5e6example,subnet-613example,subnet-c93example"
```

Console

기존 그룹에 인스턴스 유지 관리 정책을 설정하려면(콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
2. 화면 상단의 탐색 모음에서 Auto Scaling 그룹을 생성한 AWS 리전 을 선택합니다.
3. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
4. 세부 정보 탭에서 인스턴스 유지 관리 정책, 편집을 선택합니다.
5. 그룹에서 인스턴스 유지 관리 정책을 설정하려면 사용 가능한 옵션 중 하나를 선택합니다.

   - 종료 전 시작: 기존 인스턴스를 종료하기 전에 새 인스턴스를 프로비저닝해야 합니다. 이 방식은 비용 절감보다 가용성을 선호하는 경우에 적합합니다.
   - 해지 후 시작: 기존 인스턴스가 해지되는 것과 동시에 새 인스턴스가 시작됩니다. 이 방식은 가용성보다 비용 절감을 선호하는 경우에 적합합니다. 또한 현재 사용 가능한 용량보다 많은 용량을 실행하지 않아야 하는 애플리케이션에도 적합합니다.
   - 사용자 지정 정책: 이 옵션을 사용하면 인스턴스 교체 시 사용할 수 있는 용량에 대한 사용자 지정 최소 및 최대 범위로 정책을 설정할 수 있습니다. 이는 비용과 가용성의 균형을 적절하게 조정하는 데 도움이 될 수 있습니다.
6. 정상 백분율 설정의 경우, 다음 필드 중 하나 또는 둘 다에 값을 입력합니다. 활성화된 필드는 이전 단계에서 선택한 옵션에 따라 달라집니다.

   - 최소: 인스턴스 교체를 진행하는 데 필요한 최소 정상 비율을 설정합니다.
   - 최대: 인스턴스 교체 시 가능한 최대 정상 비율을 설정합니다.
7. 원하는 용량을 기준으로 교체 중 용량 보기 섹션을 확장하여 최소 및 최대 값이 그룹에 어떻게 적용되는지 확인하세요. 사용되는 정확한 값은 원하는 용량 값에 따라 달라지며, 이 값은 그룹이 조정되면 변경됩니다.
8. 업데이트를 선택합니다.

AWS CLI

기존 그룹에 인스턴스 유지 관리 정책을 설정하려면(AWS CLI)

update-auto-scaling-group명령에 `--instance-maintenance-policy` 옵션을 추가합니다. 다음 예제에서는 지정된 Auto Scaling 그룹에서 인스턴스 유지 관리 정책을 설정합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": 90,
      "MaxHealthyPercentage": 120
    }'
```

인스턴스 유지 관리 정책 제거

Auto Scaling 그룹에서 인스턴스 유지 관리 정책 사용을 중지하려면 해당 정책을 제거하면 됩니다.

Console

인스턴스 유지 관리 정책을 제거하려면(콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
2. 화면 상단의 탐색 모음에서 Auto Scaling 그룹을 생성한 AWS 리전 을 선택합니다.
3. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
4. 세부 정보 탭에서 인스턴스 유지 관리 정책, 편집을 선택합니다.
5. 인스턴스 유지 관리 정책 없음을 선택합니다.
6. 업데이트를 선택합니다.

AWS CLI

인스턴스 유지 관리 정책을 제거하려면(AWS CLI)

update-auto-scaling-group명령에 `--instance-maintenance-policy` 옵션을 추가합니다. 다음 예제에서는 지정된 Auto Scaling 그룹에서 인스턴스 유지 관리 정책을 제거합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --instance-maintenance-policy '{
      "MinHealthyPercentage": -1,
      "MaxHealthyPercentage": -1
    }'
```

Amazon EC2 Auto Scaling 수명 주기 후크

Amazon EC2 Auto Scaling은 Auto Scaling 그룹에 수명 주기 후크를 추가할 수 있는 기능을 제공합니다. 이러한 후크를 사용하면 Auto Scaling 인스턴스 수명 주기의 이벤트를 인식하는 솔루션을 생성한 다음 해당 수명 주기 이벤트가 발생할 때 인스턴스에 대한 사용자 정의 작업을 수행할 수 있습니다. 수명 주기 후크는 인스턴스가 다음 상태로 전환되기 전에 작업이 완료될 때까지 대기할 지정된 시간(기본적으로 1시간)을 제공합니다.

Auto Scaling 인스턴스에서 수명 주기 후크를 사용하는 예는 다음과 같습니다.

- 스케일 아웃 이벤트가 발생하면 새로 시작된 인스턴스는 시작 시퀀스를 완료하고 대기 상태로 전환합니다. 인스턴스가 대기 상태에 있는 동안 그룹은 스크립트를 실행하여 애플리케이션에 필요한 소프트웨어 패키지를 다운로드해 설치하여 트래픽 수신을 시작하기 전에 인스턴스가 준비를 마치도록 합니다. 스크립트가 소프트웨어 설치를 마치면 complete-lifecycle-action 명령을 전송하여 계속 진행합니다.
- 확장 이벤트가 발생하면 수명 주기 후크가 인스턴스를 일시 중지한 후 종료하고 Amazon을 사용하여 알림을 보냅니다. EventBridge 인스턴스가 대기 상태인 동안에는 인스턴스가 완전히 종료되기 전에 AWS Lambda 함수를 호출하거나 인스턴스에 연결하여 로그 또는 기타 데이터를 다운로드할 수 있습니다.

수명 주기 후크의 일반적인 용도는 인스턴스가 Elastic Load Balancing에 등록되는 시점을 제어하는 것입니다. Auto Scaling 그룹에 시작 수명 주기 후크를 추가하면 부트스트랩 스크립트가 성공적으로 완료되었고 수명 주기 후크 해지 시 로드 밸런서에 등록되기 전에 인스턴스의 애플리케이션이 트래픽을 수락할 준비가 되도록 할 수 있습니다.

내용

- 수명 주기 후크 가용성
- 수명 주기 후크에 대한 고려 사항 및 제한 사항
- 관련 리소스
- Auto Scaling 그룹에서 라이프사이클 후크가 작동하는 방식
- Auto Scaling 그룹에 대한 수명 주기 후크 추가 준비
- 인스턴스 메타데이터를 통해 대상 수명 주기 상태 검색
- Auto Scaling 그룹에 라이프사이클 후크 추가
- Auto Scaling 그룹에서 라이프사이클 작업 완료
- 자습서: 데이터 스크립트와 인스턴스 메타데이터를 사용하여 수명 주기 상태 검색

- 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성

수명 주기 후크 가용성

다음 표에는 다양한 시나리오에 사용할 수 있는 수명 주기 후크가 나와 있습니다.

이벤트	인스턴스 시작 또는 해지[1]	최대 인스턴스 수명 주기: 교체 인스턴스	인스턴스 새로 고침: 교체 인스턴스	용량 재조정: 교체 인스턴스	웜 풀: 웜 풀에 들어오고 웜 풀에서 나가는 인스턴스
인스턴스 시작	✓	✓	✓	✓	✓
인스턴스 해지	✓	✓	✓	✓	✓

`SetDesiredCapacity` 또는 `TerminateInstanceInAutoScalingGroup` 작업을 호출할 때와 같이 자동으로 시작하든 수동으로 시작하든 모든 시작 및 해지에 적용됩니다. 강제 삭제 옵션을 사용하여 인스턴스를 연결 또는 분리하거나, 인스턴스를 대기 모드로 전환하거나 대기 모드에서 해제하거나, 그룹을 삭제할 때는 적용되지 않습니다.

수명 주기 후크에 대한 고려 사항 및 제한 사항

수명 주기 후크를 사용하는 경우, 다음 고려 사항 및 제한 사항에 유의하세요:

- Amazon EC2 Auto Scaling은 Auto Scaling 그룹의 배치를 돕기 위해 자체 수명 주기를 제공합니다. 이 수명 주기는 다른 EC2 인스턴스의 수명 주기와 다릅니다. 자세한 설명은 아마존 EC2 Auto Scaling 인스턴스 라이프사이클 섹션을 참조하세요. 웜 풀의 인스턴스에 대한 수명 주기 상태 전환에 설명된 바와 같이, 웜 풀의 인스턴스에도 자체 수명 주기가 있습니다.
- 수명 주기 호크를 스팟 인스턴스와 함께 사용할 수 있지만 수명 주기 후크는 용량을 더 이상 사용할 수 없는 경우, 인스턴스 해지를 방지하지 않습니다. 이 상황은 2분 중단 알림과 함께 언제든지 발생할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 스팟 인스턴스 중단을 참조하세요. 그러나 Amazon EC2 스팟 서비스로부터 스팟 인스턴스의 중단 위험이 높아질 때 전송되는 신호인 리밸런싱 권고를 수신하는 스팟 인스턴스를 사전 교체하도록 용량 리밸런싱을 활성화할 수 있습니다. 자세한 설명은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리 섹션을 참조하세요.

- 인스턴스는 한정된 시간 동안 대기 상태를 유지할 수 있습니다. 수명 주기 후크의 기본 제한 시간은 1시간(하트비트 제한 시간)입니다. 인스턴스를 대기 상태로 유지할 수 있는 최대 시간을 지정하는 전역 제한 시간도 있습니다. 전역 제한 시간은 48시간 또는 하트비트 제한 시간의 100배 중 적은 쪽입니다.
- 수명 주기 후크의 결과는 포기 또는 계속입니다. 인스턴스를 시작하는 중인 경우, 계속은 작업에 성공했고 Amazon EC2 Auto Scaling에서 인스턴스를 서비스 상태로 전환할 수 있음을 나타냅니다. 그에 반해 포기는 사용자 지정 작업이 실패했으며, 인스턴스를 해지하고 교체할 수 있음을 표시합니다. 인스턴스가 해지 중인 경우에는 포기와 계속 둘 다 인스턴스 해지를 허용합니다. 그러나 중단은 수명 주기 후크와 같은 남아 있는 모든 작업을 중지하는 반면, 계속은 다른 모든 수명 주기 후크를 완료합니다.
- Amazon EC2 Auto Scaling은 수명 주기 후크가 일관되게 실패하는 경우, 인스턴스를 시작할 수 있도록 허용하는 속도를 제한하므로 수명 주기 작업의 영구적 오류를 테스트하고 수정해야 합니다.
- AWS CLI AWS CloudFormation, 또는 SDK를 사용하여 수명 주기 후크를 만들고 업데이트하면 에서 수명 주기 후크를 만들 때 사용할 수 없는 옵션이 제공됩니다. AWS Management Console예를 들어 Amazon EC2 Auto Scaling에서 이미 Amazon으로 이벤트를 전송하기 때문에 SNS 주제 또는 SQS 대기열의 ARN을 지정하는 필드는 콘솔에 표시되지 않습니다. EventBridge 이러한 이벤트는 필요에 따라 필터링하여 Lambda, Amazon SNS 및 Amazon SQS와 같은 AWS 서비스로 리디렉션할 수 있습니다.
- Auto Scaling 그룹을 생성하는 동안 AWS CLI AWS CloudFormation, 또는 SDK를 사용하여 CreateAutoScalingGroupAPI를 호출하여 수명 주기 후크를 여러 개 추가할 수 있습니다. 그러나 지정된 경우, 각 후크에는 동일한 알림 대상 및 IAM 역할이 있어야 합니다. 알림 대상과 역할이 서로 다른 라이프사이클 후크를 생성하려면 PutLifecycleHookAPI에 대한 개별 호출을 통해 라이프사이클 후크를 한 번에 하나씩 생성하십시오.
- 인스턴스 시작을 위한 수명 주기 후크를 추가하면 인스턴스가 상태에 도달하는 즉시 `InService` 건전성 체크 유예 기간이 시작됩니다. 자세한 설명은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 섹션을 참조하세요.

스케일 아웃 고려 사항

- 동적 규모 조정 정책은 여러 인스턴스에 걸쳐 집계되는 CPU 및 네트워크 I/O와 같은 CloudWatch 지표 데이터에 따라 확장 및 축소됩니다. 스케일 아웃 시, Amazon EC2 Auto Scaling은 Auto Scaling 그룹의 집계된 인스턴스 지표에 새 인스턴스를 즉시 계산하지 않습니다. 인스턴스가 `InService` 상태에 도달하고 인스턴스 워밍업이 완료될 때까지 기다립니다. 자세한 설명은 기본 인스턴스 워밍업 주제에 대한 스케일링 수행 고려 사항 섹션을 참조하세요.

- 스케일 인 시 집계된 인스턴스 지표에 해지 인스턴스 제거가 즉시 반영되지 않을 수 있습니다. Amazon EC2 Auto Scaling 해지 워크플로우가 시작된 직후 해지 인스턴스는 그룹의 집계된 인스턴스 지표에 대한 계산을 중지합니다.
- 대부분의 경우에 수명 주기 후크가 호출되면 수명 주기 작업이 완료되고 냉각 기간이 만료될 때까지 단순 크기 조정 정책으로 인한 크기 조정 활동이 일시 중지됩니다. 냉각 기간에 긴 간격을 설정하면 크기 조정을 재개하는 데 더 많은 시간이 걸립니다. 자세한 설명은 냉각 주제에서 수명 주기 후크는 추가적 지연을 야기할 수 있습니다. 섹션을 참조하세요. 일반적으로 단계 스케일링 또는 대상 추적 스케일링 정책을 대신 사용할 수 있는 경우, 단순 스케일링 정책을 사용하지 않는 것이 좋습니다.

관련 리소스

소개 동영상은 AWS re:Invent 2018: Amazon EC2 Auto Scaling이 켜진 상태에서 제공되는 간편한 용량 관리를 참조하십시오. YouTube

스택 템플릿에서 수명 주기 후크를 선언하는 방법을 이해하는 데 사용할 수 있는 몇 가지 JSON 및 YAML 템플릿 스니펫을 제공합니다. AWS CloudFormation 자세한 내용은 사용 설명서의 AWS::AutoScaling::LifecycleHook참조를 참조하십시오.AWS CloudFormation

GitHub저장소를 방문하여 라이프사이클 후크에 대한 예제 템플릿과 사용자 데이터 스크립트를 다운로드할 수도 있습니다.

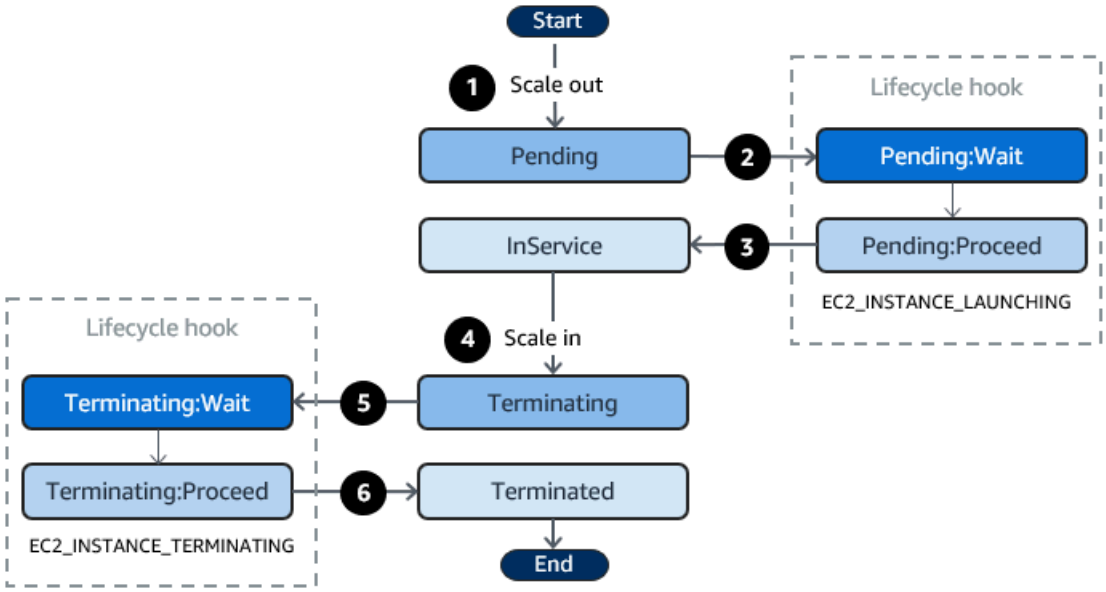
수명 주기 후크의 사용 예는 다음 블로그 게시물을 참조하세요.

- Lambda 및 Amazon EC2 실행 명령을 사용하여 스케일 아웃된 인스턴스를 위한 백업 시스템 구축
- EC2 Auto Scaling 인스턴스를 해지하기 전에 코드를 실행합니다.

Auto Scaling 그룹에서 라이프사이클 후크가 작동하는 방식

Amazon EC2 인스턴스는 시작한 순간부터 해지될 때까지 여러 상태로 전환됩니다. 수명 주기 후크로 인해 인스턴스가 대기 상태로 전환될 때 Auto Scaling 그룹이 행동하도록 사용자 지정 조치를 만들 수 있습니다.

다음 그림은 확장 및 축소에 수명 주기 후크를 사용할 때 Auto Scaling 인스턴스 상태 간의 전환을 보여줍니다.



앞의 다이어그램에 표시된 것처럼 다음과 같습니다.

1. Auto Scaling 그룹은 스케일 아웃 이벤트에 응답하고 인스턴스 실행을 시작합니다.
2. 수명 주기 후크는 인스턴스를 대기 상태(`Pending:Wait`)로 전환한 다음 사용자 지정 작업을 수행합니다.

   인스턴스는 수명 주기 작업을 완료하거나 제한 시간이 끝날 때까지 대기 상태로 유지됩니다. 기본적으로 인스턴스는 한 시간 동안 대기 상태로 유지된 후 Auto Scaling 그룹에서 시작 프로세스를 진행합니다(`Pending:Proceed`). 시간이 더 필요한 경우, 하트비트를 기록하여 제한 시간을 다시 시작할 수 있습니다. 사용자 지정 작업이 완료됐으나 제한 기간이 만료되지 않았을 때 수명 주기 작업을 완료하면 제한 기간이 해지되고 Auto Scaling 그룹이 시작 프로세스를 계속합니다.
3. 인스턴스가 `InService` 상태로 전환되고 건전성 체크 유예 기간이 시작됩니다. 그러나 인스턴스가 `InService` 상태에 도달하기 전에, Auto Scaling 그룹이 Elastic Load Balancing 로드 밸런서와 연결된 경우, 인스턴스가 로드 밸런서에 등록되고 로드 밸런서가 해당 상태를 확인하기 시작합니다. 건전성 체크 유예 기간이 끝나면 Amazon EC2 Auto Scaling이 인스턴스의 건전성 체크를 시작합니다.
4. Auto Scaling 그룹은 축소 이벤트에 응답하고 인스턴스 해지를 시작합니다. Auto Scaling 그룹이 Elastic Load Balancing과 함께 사용되는 경우, 해지 인스턴스가 먼저 로드 밸런서에서 등록 취소됩니다. 로드 밸런서에 대해 Connection Draining이 활성화된 경우, 인스턴스는 새 연결 수락을 중지하고 기존 연결이 드레이닝될 때까지 기다린 다음 등록 취소 프로세스를 완료합니다.
5. 수명 주기 후크는 인스턴스를 대기 상태(`Terminating:Wait`)로 전환한 다음 사용자 지정 작업을 수행합니다.

수명 주기 작업을 완료할 때까지 혹은 제한 시간(기본 1시간)이 끝날 때까지 인스턴스는 대기 상태로 유지됩니다. 수명 주기 후크를 완료하거나 시간 초과 기간이 만료되면 인스턴스가 다음 상태(`Terminating:Proceed`)로 전환됩니다.

6. 인스턴스가 해지됩니다.

> **Important**
>
> 웜 풀의 인스턴스에 대한 수명 주기 상태 전환에 설명된 바와 같이, 웜 풀의 인스턴스에도 해당 대기 상태와 자체 수명 주기가 있습니다.

Auto Scaling 그룹에 대한 수명 주기 후크 추가 준비

Auto Scaling 그룹에 수명 주기 후크를 추가하기 전에 사용자 데이터 스크립트 또는 알림 대상이 올바르게 설정되어 있는지 확인하세요.

- 인스턴스가 시작될 때 사용자 데이터 스크립트를 사용하여 사용자 지정 작업을 수행하려는 경우, 알림 대상을 구성할 필요가 없습니다. 그러나 사용자 데이터 스크립트를 지정하는 시작 템플릿 또는 시작 구성을 사전에 생성하고 이를 Auto Scaling 그룹과 연결해야 합니다. 사용자 데이터 스크립트에 대한 자세한 내용은 Amazon EC2 사용 설명서의 시작 시 Linux 인스턴스에서 명령 실행을 참조하십시오.
- 수명 주기 작업이 완료될 때 Amazon EC2 Auto Scaling에 신호를 보내려면 스크립트에 CompleteLifecycleActionAPI 호출을 추가하고, Auto Scaling 인스턴스가 이 API를 호출하도록 허용하는 정책을 사용하여 IAM 역할을 수동으로 생성해야 합니다. 시작 템플릿 또는 시작 구성은 시작 시 Amazon EC2 인스턴스에 연결되는 IAM 인스턴스 프로필을 사용하여 이 역할을 지정해야 합니다. 자세한 내용은 Auto Scaling 그룹에서 라이프사이클 작업 완료 및 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 섹션을 참조하세요.
- Lambda와 같은 서비스를 사용하여 사용자 지정 작업을 수행하려면 이미 규칙을 생성하고 EventBridge Lambda 함수를 대상으로 지정해야 합니다. 자세한 정보는 수명 주기 알림에 대한 알림 대상 구성을 참조하세요.
- 라이프사이클 작업이 완료될 때 Lambda가 Amazon EC2 Auto Scaling에 신호를 보낼 수 있도록 하려면 함수 코드에 API 호출을 CompleteLifecycleAction추가해야 합니다. 또한 Lambda에 수명 주기 작업을 완료할 수 있는 권한을 부여하는 함수의 집행 역할에 IAM 정책을 연결해야 합니다. 자세한 설명은 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성 섹션을 참조하세요.

- Amazon SNS 또는 Amazon SQS와 같은 서비스를 사용하여 사용자 지정 작업을 수행하려면 사전에 SNS 주제 또는 SQS 대기열을 생성하고 Amazon 리소스 이름(ARN)을 준비해야 합니다. 또한 사전에 Amazon EC2 Auto Scaling에 SNS 주제 또는 SQS 대상에 대한 액세스 권한을 부여하는 IAM 역할을 생성하고 ARN을 준비해야 합니다. 자세한 정보는 수명 주기 알림에 대한 알림 대상 구성을 참조하세요.

> **Note**
>
> 기본적으로 콘솔에서 수명 주기 후크를 추가하면 Amazon EC2 Auto Scaling에서 Amazon에 수명 주기 이벤트 알림을 보냅니다. EventBridge EventBridge 또는 사용자 데이터 스크립트를 사용하는 것이 가장 좋습니다. Amazon SNS 또는 Amazon SQS로 알림을 직접 보내는 수명 주기 후크를 생성하려면 AWS CLI AWS CloudFormation, 또는 SDK를 사용하여 수명 주기 후크를 추가하십시오.

수명 주기 알림에 대한 알림 대상 구성

Auto Scaling 그룹에 수명 주기 후크를 추가하여 인스턴스가 대기 상태로 전환될 때 사용자 정의 작업을 수행할 수 있습니다. 선호하는 개발 방식에 따라 다양한 Amazon Web Services를 사용하여 이러한 작업을 수행하도록 알림 대상을 구성할 수 있습니다.

첫 번째 접근 방식은 EventBridge Amazon을 사용하여 원하는 작업을 수행하는 Lambda 함수를 호출하는 것입니다. 두 번째 방법은 알림을 게시할 Amazon Simple Notification Service(Amazon SNS) 주제를 생성하는 것입니다. 클라이언트는 SNS 주제를 구독하고 지원되는 프로토콜을 사용하여 게시된 메시지를 수신할 수 있습니다. 마지막 접근 방식은 분산 애플리케이션에서 폴링 모델을 통해 메시지를 교환하는 데 사용되는 메시징 시스템인 Amazon Simple Queue Service(Amazon SQS)를 사용하는 것입니다.

모범 사례로 사용하는 것이 좋습니다. EventBridge Amazon SNS와 Amazon SQS로 전송된 알림에는 Amazon EC2 Auto Scaling에서 보내는 알림과 동일한 정보가 포함되어 있습니다. EventBridge 이전에는 EventBridge SNS나 SQS에 알림을 보내고 다른 서비스를 SNS 또는 SQS와 통합하여 프로그래밍 작업을 수행하는 것이 표준 관행이었습니다. 이제 타겟팅할 수 있는 더 많은 옵션이 EventBridge 제공되고 서버리스 아키텍처를 사용하여 이벤트를 더 쉽게 처리할 수 있습니다.

다음 절차에서는 알림 대상을 설정하는 방법에 대해 설명합니다.

시작 템플릿 또는 시작 구성에 인스턴스가 시작될 때 인스턴스를 구성하는 사용자 지정 스크립트가 있는 경우, 인스턴스에 사용자 지정 작업을 수행하기 위해 알림을 받을 필요가 없습니다.

내용

- 를 사용하여 Lambda로 알림을 라우팅합니다. EventBridge
- Amazon SNS를 사용하여 알림 수신
- Amazon SQS 사용하여 알림 수신
- Amazon SNS 및 Amazon SQS에 대한 알림 메시지의 예

> **Important**
> 라이프사이클 후크와 함께 사용하는 EventBridge 규칙, Lambda 함수, Amazon SNS 주제 및 Amazon SQS 대기열은 항상 Auto Scaling 그룹을 생성한 지역과 동일한 지역에 있어야 합니다.

를 사용하여 Lambda로 알림을 라우팅합니다. EventBridge

인스턴스가 대기 상태에 들어갈 때 Lambda 함수를 호출하도록 EventBridge 규칙을 구성할 수 있습니다. Amazon EC2 Auto Scaling은 시작 또는 종료 중인 인스턴스와 수명 주기 작업을 제어하는 데 사용할 수 있는 EventBridge 토큰에 대한 수명 주기 이벤트 알림을 내보냅니다. 이러한 이벤트의 예는 Amazon EC2 Auto Scaling 이벤트 참조 섹션을 참조하세요.

> **Note**
> 를 사용하여 이벤트 규칙을 생성하면 콘솔은 Lambda 함수를 호출할 권한을 부여하는 EventBridge 데 필요한 IAM 권한을 자동으로 추가합니다. AWS Management Console AWS CLI를 사용하여 이벤트 규칙을 만드는 경우, 이 권한을 명시적으로 부여해야 합니다. EventBridge 콘솔에서 이벤트 규칙을 생성하는 방법에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 이벤트에 반응하는 Amazon EventBridge 규칙 생성을 참조하십시오.
> 또는
> 콘솔 사용자를 대상으로 하는 입문자용 자습서는 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성 섹션을 참조하세요. 이 자습서에서는 시작 이벤트를 수신하고 로그 로그에 기록하는 간단한 Lambda 함수를 생성하는 방법을 보여줍니다. CloudWatch

Lambda 함수를 호출하는 EventBridge 규칙을 생성하려면

1. Lambda 콘솔을 사용하여 Lambda 함수를 생성하고 Amazon 리소스 이름(ARN)을 적어둡니다. 예를 들어 `arn:aws:lambda:region:123456789012:function:my-function`입니다. 대상을 만들려면 ARN이 필요합니다. EventBridge 자세한 설명은 AWS Lambda 개발자 가이드에서 Lambda 시작하기를 참조하세요.
2. 인스턴스 시작에 대한 이벤트와 일치하는 규칙을 생성하려면 다음 put-rule 명령을 사용합니다.

```
aws events put-rule --name my-rule --event-pattern file://pattern.json --state
 ENABLED
```

다음 예에서는 인스턴스 시작 수명 주기 작업을 위한 `pattern.json`을 보여 줍니다. `####` 텍스트를 Auto Scaling 그룹의 이름으로 바꿉니다.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "AutoScalingGroupName": [ "my-asg" ]
   }
}
```

명령이 성공적으로 실행되면 규칙의 ARN으로 EventBridge 응답합니다. ARN을 적어 두세요. 4단계에서 이 정보가 필요합니다.

다른 이벤트와 일치하는 규칙을 생성하려면 이벤트 패턴을 수정합니다. 자세한 설명은 Auto Scaling 이벤트를 처리하는 EventBridge 데 사용합니다. 섹션을 참조하세요.

3. 규칙의 대상으로 사용할 Lambda 함수를 지정하려면 다음 put-targets 명령을 사용합니다.

```
aws events put-targets --rule my-rule --targets
 Id=1,Arn=arn:aws:lambda:region:123456789012:function:my-function
```

앞의 명령에서 *my-rule*은 2단계에서 규칙에 지정한 이름이고, `Arn` 파라미터 값은 1단계에서 생성한 함수의 ARN입니다.

4. 규칙이 Lambda 함수를 호출하도록 허용하는 권한을 추가하려면 다음 Lambda add-permission 명령을 사용합니다. 이 명령은 EventBridge 서비스 주체 (`events.amazonaws.com`) 를 신뢰하고 권한 범위를 지정된 규칙으로 지정합니다.

```
aws lambda add-permission --function-name my-function --statement-id my-unique-id \
  --action 'lambda:InvokeFunction' --principal events.amazonaws.com --source-arn
 arn:aws:events:region:123456789012:rule/my-rule
```

앞의 명령에서:

- *my-function*은 규칙에서 대상으로 사용하도록 할 Lambda 함수의 이름입니다.
- *my-unique-id*Lambda 함수 정책에서 명령문을 설명하기 위해 정의하는 고유 식별자입니다.
- `source-arn`규칙의 ARN입니다. EventBridge

이 명령이 성공적으로 실행되면 다음과 비슷한 출력이 표시됩니다.

```
{
  "Statement": "{\"Sid\":\"my-unique-id\",
    \"Effect\":\"Allow\",
    \"Principal\":{\"Service\":\"events.amazonaws.com\"},
    \"Action\":\"lambda:InvokeFunction\",
    \"Resource\":\"arn:aws:lambda:us-west-2:123456789012:function:my-function\",
    \"Condition\":
      {\"ArnLike\":
        {\"AWS:SourceArn\":
         \"arn:aws:events:us-west-2:123456789012:rule/my-rule\"}}}"
}
```

`Statement` 값은 Lambda 함수 정책에 추가된 문의 JSON 문자열 버전입니다.

5. 이 지침을 따른 후에는 다음 단계 Auto Scaling 그룹에 라이프사이클 후크 추가로 계속 진행합니다.

Amazon SNS를 사용하여 알림 수신

Amazon SNS를 사용하여 수명 주기 작업 발생 시 알림을 수신하도록 Amazon SNS 알림 대상(SNS 주제)을 설정할 수 있습니다. 그러면 Amazon SNS에서는 구독한 수신자에게 알림을 보냅니다. 구독이 확인되기 전에는 주제에 게시된 알림이 수신자에게 전송되지 않습니다.

Amazon SNS를 사용하여 알림 설정

1. Amazon SNS 콘솔 또는 다음 create-topic 명령을 사용하여 Amazon SNS 주제를 생성합니다. 사용 중인 Auto Scaling 그룹과 동일한 지역에 주제가 있는지 확인합니다. 자세한 설명은 Amazon Simple Notification Service 개발자 안내서의 Amazon SNS 시작하기를 참조하세요.

```
aws sns create-topic --name my-sns-topic
```

2. 주제 Amazon 리소스 이름(ARN)을 기록합니다(예: `arn:aws:sns:region:123456789012:my-sns-topic`). 이 이름은 수명 주기 후크를 생성하는데 필요합니다.
3. Amazon SNS 알림 대상에게 Amazon EC2 Auto Scaling 액세스 권한을 부여할 IAM 서비스 역할을 생성합니다.

   Amazon EC2 Auto Scaling에 SNS 주제에 대한 액세스 권한 부여

   a. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
   b. 왼쪽의 탐색 창에서 역할을 선택합니다.
   c. 역할 생성을 선택합니다.
   d. 신뢰할 수 있는 엔터티 선택(Select trusted entity)에서 AWS 서비스(service)를 선택합니다.
   e. 사용 사례를 보려면 다른 AWS 서비스의 사용 사례(Use cases for other services)에서 EC2 Auto Scaling을 선택한 다음 EC2 Auto Scaling 알림 액세스(EC2 Auto Scaling Notification Access)를 선택합니다.
   f. 다음(Next)을 두 번 선택하여 이름 지정, 검토 및 생성(Name, review, and create) 페이지로 이동합니다.
   g. 역할 이름(Role name)에 역할의 이름(예: **`my-notification-role`**)을 입력하고 역할 생성(Create role)을 선택합니다.
   h. 역할 페이지에서 방금 만든 역할을 선택하여 요약 페이지를 엽니다. 역할 ARN을 기록해 둡니다. 예를 들어 `arn:aws:iam::123456789012:role/my-notification-role`입니다. 이 이름은 수명 주기 후크를 생성하는데 필요합니다.
4. 이 지침을 따른 후에는 다음 단계 수명 주기 후크 추가(AWS CLI)로 계속 진행합니다.

Amazon SQS 사용하여 알림 수신

Amazon SQS를 사용하여 수명 주기 작업이 발생할 때 메시지를 수신할 알림 대상을 설정할 수 있습니다. 그런 다음 대기열 소비자는 이러한 알림에 따라 작업을 수행하기 위해 SQS 대기열을 폴링해야 합니다.

> **Important**
>
> FIFO 대기열은 수명 주기 후크와 호환되지 않습니다.

Amazon SQS를 사용하여 알림 설정

1. Amazon SQS 콘솔을 사용하여 SQS 대기열을 생성합니다. 사용 중인 Auto Scaling 그룹과 동일한 지역에 대기열이 있는지 확인합니다. 자세한 설명은 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 시작하기를 참조하세요.
2. 대기열 ARN을 적어 둡니다(예: `arn:aws:sqs:us-west-2:123456789012:my-sqs-queue`). 이 이름은 수명 주기 후크를 생성하는데 필요합니다.
3. Amazon SQS 알림 대상에게 Amazon EC2 Auto Scaling 액세스 권한을 부여할 IAM 서비스 역할을 생성합니다.

   Amazon EC2 Auto Scaling에 SQS 대기열에 대한 액세스 권한 부여

   a. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
   b. 왼쪽의 탐색 창에서 역할을 선택합니다.
   c. 역할 생성을 선택합니다.
   d. 신뢰할 수 있는 엔터티 선택(Select trusted entity)에서 AWS 서비스(service)를 선택합니다.
   e. 사용 사례를 보려면 다른 AWS 서비스의 사용 사례(Use cases for other services)에서 EC2 Auto Scaling을 선택한 다음 EC2 Auto Scaling 알림 액세스(EC2 Auto Scaling Notification Access)를 선택합니다.
   f. 다음(Next)을 두 번 선택하여 이름 지정, 검토 및 생성(Name, review, and create) 페이지로 이동합니다.
   g. 역할 이름(Role name)에 역할의 이름(예: **`my-notification-role`**)을 입력하고 역할 생성(Create role)을 선택합니다.
   h. 역할 페이지에서 방금 만든 역할을 선택하여 요약 페이지를 엽니다. 역할 ARN을 기록해 둡니다. 예를 들어 `arn:aws:iam::123456789012:role/my-notification-role`입니다. 이 이름은 수명 주기 후크를 생성하는데 필요합니다.

4. 이 지침을 따른 후에는 다음 단계 수명 주기 후크 추가(AWS CLI)로 계속 진행합니다.

Amazon SNS 및 Amazon SQS에 대한 알림 메시지의 예

인스턴스가 대기 상태에 있는 동안 Amazon SNS 또는 Amazon SQS 알림 대상에게 메시지가 전송됩니다. 메시지에 포함되는 정보는 다음과 같습니다.

- `LifecycleActionToken` - 수명 주기 작업 토큰
- `AccountId`— AWS 계정 ID.
- `AutoScalingGroupName` — Auto Scaling 그룹의 이름입니다.
- `LifecycleHookName` - 수명 주기 후크 이름
- `EC2InstanceId` — EC2 인스턴스의 ID입니다.
- `LifecycleTransition` - 수명 주기 후크 유형
- `NotificationMetadata` - 알림 메타데이터

다음은 알림 메시지의 예입니다.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:36:26.533Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
LifecycleActionToken: 71514b9d-6a40-4b26-8523-05e7ee35fa40
AccountId: 123456789012
AutoScalingGroupName: my-asg
LifecycleHookName: my-hook
EC2InstanceId: i-0598c7d356eba48d7
LifecycleTransition: autoscaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata: hook message metadata
```

테스트 알림 메시지의 예

수명 주기 후크를 처음 추가하면 알림 대상에게 테스트 알림 메시지가 전송됩니다. 다음은 테스트 알림 메시지의 예입니다.

```
Service: AWS Auto Scaling
Time: 2021-01-19T00:35:52.359Z
RequestId: 18b2ec17-3e9b-4c15-8024-ff2e8ce8786a
Event: autoscaling:TEST_NOTIFICATION
AccountId: 123456789012
AutoScalingGroupName: my-asg
```

```
AutoScalingGroupARN: arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:042cba90-
ad2f-431c-9b4d-6d9055bcc9fb:autoScalingGroupName/my-asg
```

> **Note**
>
> Amazon EC2 Auto Scaling에서 전송된 이벤트의 예는 을 EventBridge 참조하십시오. Amazon EC2 Auto Scaling 이벤트 참조

인스턴스 메타데이터를 통해 대상 수명 주기 상태 검색

사용자가 시작하는 각 Auto Scaling 인스턴스는 여러 수명 주기 상태를 거칩니다. 특정 수명 주기 상태 전환 시에 행동하는 인스턴스 내로부터 사용자 지정 조치를 호출하려면 인스턴스 메타데이터를 통해 대상 수명 주기 상태를 검색해야 합니다.

예컨대, 인스턴스가 해지되기 전에 인스턴스에서 일부 코드를 실행하기 위해 인스턴스 내부에서 인스턴스 해지를 감지하는 메커니즘이 필요할 수 있습니다. 인스턴스에서 직접 인스턴스의 수명 주기 상태를 폴링하는 코드를 작성하여 이 작업을 수행할 수 있습니다. 그런 다음 Auto Scaling 그룹에 수명 주기 후크를 추가하여 코드에서 계속하라는 complete-lifecycle-action 명령을 보낼 때까지 인스턴스를 계속 실행할 수 있습니다.

Auto Scaling 인스턴스 수명 주기에는 `InService`와 `Terminated`의 두 가지 기본 건전 상태 및 `Detached`와 `Standby`의 두 가지 부 건전 상태가 있습니다. 웜 풀을 사용하는 경우, 수명 주기에는 `Warmed:Hibernated`, `Warmed:Running`, `Warmed:Stopped` 및 `Warmed:Terminated`의 네 가지 추가 건전 상태가 있습니다.

인스턴스가 이전 건전 상태 중 하나로 전환될 준비가 되면 Amazon EC2 Auto Scaling은 `autoscaling/target-lifecycle-state` 인스턴스 메타데이터 항목의 값을 업데이트합니다. 인스턴스 내에서 대상 수명 주기 상태를 가져오려면 인스턴스 메타데이터 서비스를 사용하여 인스턴스 메타데이터에서 해당 상태를 검색해야 합니다.

> **Note**
>
> 인스턴스 메타데이터는 애플리케이션에서 인스턴스 정보를 쿼리하는 데 사용할 수 있는 Amazon EC2 인스턴스 관련 데이터입니다. 인스턴스 메타데이터 서비스는 지역 코드가 인스턴스 메타데이터에 액세스하기 위해 사용하는 온인스턴스 구성 요소입니다. 지역 코드에는 인스턴스에서 실행 중인 사용자 데이터 스크립트 또는 애플리케이션이 포함될 수 있습니다.

지역 코드는 인스턴스 메타데이터 서비스 버전 1(IMDSv1) 또는 인스턴스 메타데이터 서비스 버전 2(IMDSv2)의 두 가지 방법 중 하나를 사용하여 실행 중인 인스턴스에서 인스턴스 메타데이터에 액세스할 수 있습니다. IMDSv2는 세션 지향 요청을 사용하며 인스턴스 메타데이터에 액세스하기 위해 사용될 수 있는 여러 유형의 취약성을 완화합니다. 이 두 가지 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 IMDSv2 사용을 참조하십시오.

IMDSv2

```
[ec2-user ~]$ TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"` \
&& curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

IMDSv1

```
[ec2-user ~]$ curl http://169.254.169.254/latest/meta-data/autoscaling/target-lifecycle-state
```

출력의 예제는 다음과 같습니다.

```
InService
```

대상 수명 주기 상태는 인스턴스가 전환 중인 대상 상태입니다. 현재 수명 주기 상태는 인스턴스의 현재 상태입니다. 이런 상태는 수명 주기 작업이 완료되고 인스턴스가 대상 수명 주기 상태로 전환을 끝마친 후에도 동일할 수 있습니다. 인스턴스 메타데이터에서 인스턴스의 현재 수명 주기 상태를 검색할 수 없습니다.

Amazon EC2 Auto Scaling은 2022년 3월 10일에 대상 수명 주기 상태를 생성하기 시작했습니다. 인스턴스가 이 날짜 이후에 대상 수명 주기 상태 중 하나로 전환되면 인스턴스 메타데이터에 대상 수명 주기 상태 항목이 있습니다. 그렇지 않은 경우, 해당 항목이 없고 HTTP 404 오류가 발생합니다.

인스턴스 메타데이터 검색에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 검색을 참조하십시오.

대상 수명 주기 상태를 사용하는 사용자 데이터 스크립트에서 사용자 지정 작업을 통해 수명 주기 후크를 만드는 방법을 설명하는 자습서는 자습서: 데이터 스크립트와 인스턴스 메타데이터를 사용하여 수명 주기 상태 검색 섹션을 참조하세요.

> **Important**
>
> 가능한 한 빨리 사용자 지정 작업을 호출할 수 있도록 로컬 코드에서 IMDS를 자주 폴링하고 오류가 발생하면 다시 시도해야 합니다.

Auto Scaling 그룹에 라이프사이클 후크 추가

Auto Scaling 인스턴스를 대기 상태로 전환하고 해당 인스턴스에 대해 사용자 지정 작업을 수행하려는 경우, Auto Scaling 그룹에 수명 주기 후크를 추가할 수 있습니다. 사용자 지정 작업은 인스턴스가 시작될 때 또는 해지되기 전에 수행됩니다. 수명 주기 작업을 완료할 때까지 혹은 제한 시간이 끝날 때까지 인스턴스는 대기 상태로 유지됩니다.

에서 Auto Scaling 그룹을 생성한 후 AWS Management Console, 그룹에 수명 주기 후크를 하나 이상 추가할 수 있으며, 총 50개의 수명 주기 후크를 추가할 수 있습니다. AWS CLI AWS CloudFormation, 또는 SDK를 사용하여 Auto Scaling 그룹을 생성할 때 수명 주기 후크를 추가할 수도 있습니다.

기본적으로 콘솔에서 수명 주기 후크를 추가하면 Amazon EC2 Auto Scaling에서 Amazon에 수명 주기 이벤트 알림을 보냅니다. EventBridge EventBridge 또는 사용자 데이터 스크립트를 사용하는 것이 가장 좋습니다. Amazon SNS 또는 Amazon SQS로 알림을 직접 보내는 수명 주기 후크를 생성하려면 이 항목의 예제에 표시된 대로 put-lifecycle-hook명령을 사용할 수 있습니다.

내용

- 수명 주기 후크 추가(콘솔)
- 수명 주기 후크 추가(AWS CLI)

수명 주기 후크 추가(콘솔)

Auto Scaling 그룹에 수명 주기 후크를 추가하려면 다음 단계를 따르십시오. 스케일 아웃(인스턴스 시작) 및 스케일 인(인스턴스 해지)를 위한 수명 주기 후크를 생성하려면 두 개의 개별 후크를 생성해야 합니다.

시작하기 전에 필요한 대로 Auto Scaling 그룹에 대한 수명 주기 후크 추가 준비의 설명에 따라 사용자 지정 작업을 설정하십시오.

스케일 아웃을 위한 수명 주기 후크를 추가

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.

2. Auto Scaling 그룹 옆의 확인란을 선택합니다. 페이지 하단에 분할 창이 열립니다.
3. 인스턴스 관리 탭의 Lifecycle hooks(수명 주기 후크)에서 Create lifecycle hook(수명 주기 후크 생성)을 선택합니다.
4. 스케일 아웃(인스턴스 실행)을 위한 수명 주기 후크를 정의하려면 다음을 수행하세요.
   a. Lifecycle hook name(수명 주기 후크 이름)에 수명 주기 후크의 이름을 지정합니다.
   b. Lifecycle transition(수명 주기 전환)에서 Instance launch(인스턴스 시작)를 선택합니다.
   c. 하트비트 제한 시간에서는 스케일 아웃할 때 훅이 시간 초과되기 전에 인스턴스가 대기 상태로 유지되는 시간을 지정하십시오. 범위는 30 ~ 7200초입니다. 제한 시간을 길게 설정하면 사용자 지정 작업을 완료하기 위한 시간을 더 많이 제공합니다. 그런 다음 제한 시간이 끝나기 전에 작업을 마치면 complete-lifecycle-action명령을 전송하여 인스턴스가 다음 상태로 진행되도록 하십시오.
   d. 기본 결과(Default result)에서 수명 주기 후크 제한 시간이 경과하거나 예기치 못한 오류가 발생할 때 수행할 작업을 지정합니다. 계속 또는 포기를 선택할 수 있습니다.
      - 계속을 선택하면 Auto Scaling 그룹이 다른 수명 주기 후크를 진행한 다음 인스턴스를 서비스할 수 있습니다.
      - 포기를 선택하면 Auto Scaling 그룹이 잔여 작업을 중단하고 인스턴스를 즉각 해지합니다.
   e. (옵션) 알림 메타데이터에 Amazon EC2 Auto Scaling이 알림 대상에 메시지를 전송할 때 포함할 다른 정보를 지정합니다.
5. 생성을 선택합니다.

스케일 인을 위한 수명 주기 후크 추가

1. 스케일 아웃을 위한 수명 주기 후크를 생성한 후 중단한 부분부터 계속하려면 수명 주기 후크 만들기를 선택합니다.
2. 스케일 인(인스턴스가 해지되거나 웜 풀로 돌아가는 것)을 위한 수명 주기 후크를 정의하려면 다음과 같이 하세요.
   a. Lifecycle hook name(수명 주기 후크 이름)에 수명 주기 후크의 이름을 지정합니다.
   b. Lifecycle transition(수명 주기 전환)에서 Instance terminate(인스턴스 해지)를 선택합니다.
   c. 하트비트 제한 시간에서는 스케일 아웃할 때 훅이 시간 초과되기 전에 인스턴스가 대기 상태로 유지되는 시간을 지정하십시오. 최종 작업 (예: EC2 로그 가져오기) 30 을 수행하는 데 필요한 시간에 따라 120 몇 초에서 몇 초까지의 짧은 제한 시간을 사용하는 것이 좋습니다. CloudWatch

d. Default result(기본 결과)에 제한 시간이 경과하거나 예기치 못한 오류가 발생할 때 Auto Scaling 그룹에서 수행하는 작업을 지정합니다. ABANDON(중단) 및 CONTINUE(계속) 둘 다 인스턴스를 해지할 수 있습니다.

   - CONTINUE(계속)을 선택하면 해지 전에 Auto Scaling 그룹이 남아 있는 모든 작업(예: 다른 수명 주기 후크)을 진행할 수 있습니다.
   - 포기를 선택하면 Auto Scaling 그룹이 인스턴스를 즉각 해지합니다.

e. (옵션) 알림 메타데이터에 Amazon EC2 Auto Scaling이 알림 대상에 메시지를 전송할 때 포함할 추가 정보를 지정합니다.

3. 생성을 선택합니다.

수명 주기 후크 추가(AWS CLI)

명령을 사용하여 라이프사이클 후크를 생성하고 업데이트하십시오. put-lifecycle-hook

스케일 아웃 시 작업을 수행하려면 다음 명령을 사용합니다.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-launch-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING
```

축소 시 작업을 수행하려면 대신 다음 명령을 사용합니다.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING
```

Amazon SNS 또는 Amazon SQS를 사용하여 알림을 수신하려면 `--notification-target-arn`과 `--role-arn` 옵션을 추가합니다.

다음 예에서는 *my-sns-topic*이라는 SNS 주제를 알림 대상으로 지정하는 수명 주기 후크를 생성합니다.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook  \
  --auto-scaling-group-name my-asg \
  --lifecycle-transition autoscaling:EC2_INSTANCE_TERMINATING \
  --notification-target-arn arn:aws:sns:region:123456789012:my-sns-topic \
  --role-arn arn:aws:iam::123456789012:role/my-notification-role
```

이 주제는 다음 키-값 쌍에 해당하는 테스트 알림을 보냅니다.

```
"Event": "autoscaling:TEST_NOTIFICATION"
```

기본적으로 이 put-lifecycle-hook명령은 하트비트 제한 시간이 3600 초 (1시간) 인 수명 주기 후크를 생성합니다.

기존 수명 주기 후크에 대한 하트비트 제한 시간을 변경하려면 다음 예에 따라 --heartbeat-timeout 옵션을 추가합니다.

```
aws autoscaling put-lifecycle-hook --lifecycle-hook-name my-termination-hook \
  --auto-scaling-group-name my-asg --heartbeat-timeout 120
```

인스턴스가 이미 대기 상태인 경우 record-lifecycle-action-heartbeatCLI 명령을 사용하여 하트비트를 기록하여 수명 주기 후크가 시간 초과되는 것을 방지할 수 있습니다. 그러면 수명 주기 후크를 만들 때 지정한 제한 시간 값을 기준으로 제한 시간이 늘어납니다. 제한 시간이 끝나기 전에 작업을 완료하면 complete-lifecycle-actionCLI 명령을 전송하여 인스턴스가 다음 상태로 진행되도록 할 수 있습니다. 자세한 정보와 지침은 Auto Scaling 그룹에서 라이프사이클 작업 완료 섹션을 참조하세요.

Auto Scaling 그룹에서 라이프사이클 작업 완료

Auto Scaling 그룹에서 수명 주기 이벤트에 응답할 때 인스턴스를 대기 상태로 설정하고 이벤트 알림을 전송합니다. 인스턴스가 대기 상태에 있는 동안 사용자 지정 작업을 수행할 수 있습니다.

타임아웃 기간이 만료되기 전에 완료하면 결과가 CONTINUE인 수명 주기 작업을 완료하는 데 도움이 됩니다. 수명 주기 작업을 완료하지 않으면 시간 초과 기간이 해지된 후 수명 주기 후크가 기본 결과에 지정된 상태로 전환됩니다.

내용

- 수명 주기 작업 완료(수동)
- 수명 주기 작업 완료(자동)

수명 주기 작업 완료(수동)

다음 절차는 명령행 인터페이스용이며 콘솔에서는 지원되지 않습니다. 인스턴스 ID 또는 Auto Scaling 그룹의 이름과 같이 교체해야 하는 정보는 기울임꼴으로 표시됩니다.

수명 주기 작업 완료(AWS CLI)

1. 사용자 지정 작업을 완료하는 데 시간이 더 필요한 경우 record-lifecycle-action-heartbeat명령을 사용하여 제한 시간을 다시 시작하고 인스턴스를 대기 상태로 유지합니다. 예컨대, 제한 시간이 1시간이고 이 명령을 30분 후에 호출한 경우, 인스턴스는 추가 1시간 동안 대기 상태로 유지됩니다(총 90분).

   다음 명령과 같이 알림에서 받은 수명 주기 작업 토큰을 지정할 수 있습니다.

   ```
   aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \
     --auto-scaling-group-name my-asg --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
   ```

   또는 다음 명령과 같이 알림과 함께 받은 인스턴스의 ID를 지정할 수 있습니다.

   ```
   aws autoscaling record-lifecycle-action-heartbeat --lifecycle-hook-name my-launch-hook \
     --auto-scaling-group-name my-asg --instance-id i-1a2b3c4d
   ```

2. 제한 시간이 끝나기 전에 사용자 지정 작업을 완료한 경우 Auto Scaling 그룹이 인스턴스를 계속 시작하거나 종료할 수 있도록 complete-lifecycle-action명령을 사용하십시오. 다음 명령과 같이 수명 주기 작업 토큰을 지정할 수 있습니다.

   ```
   aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \
     --lifecycle-hook-name my-launch-hook --auto-scaling-group-name my-asg \
     --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
   ```

   또는 다음 명령과 같이 인스턴스의 ID를 지정할 수도 있습니다.

   ```
   aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINUE \
     --instance-id i-1a2b3c4d --lifecycle-hook-name my-launch-hook \
     --auto-scaling-group-name my-asg
   ```

수명 주기 작업 완료(자동)

인스턴스를 시작 후에 구성하는 사용자 데이터 스크립트가 있는 경우, 수명 주기 작업을 수동으로 완료할 필요가 없습니다. 스크립트에 complete-lifecycle-action명령을 추가할 수 있습니다. 스크립트는 인스

턴스 메타데이터에서 인스턴스 ID를 검색하고 부트스트랩 스크립트가 성공적으로 완료되면 Amazon EC2 Auto Scaling에 알릴 수 있습니다.

아직 없는 경우, 인스턴스 메타데이터에서 인스턴스의 인스턴스 ID를 검색하도록 스크립트를 업데이트합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 검색을 참조하십시오.

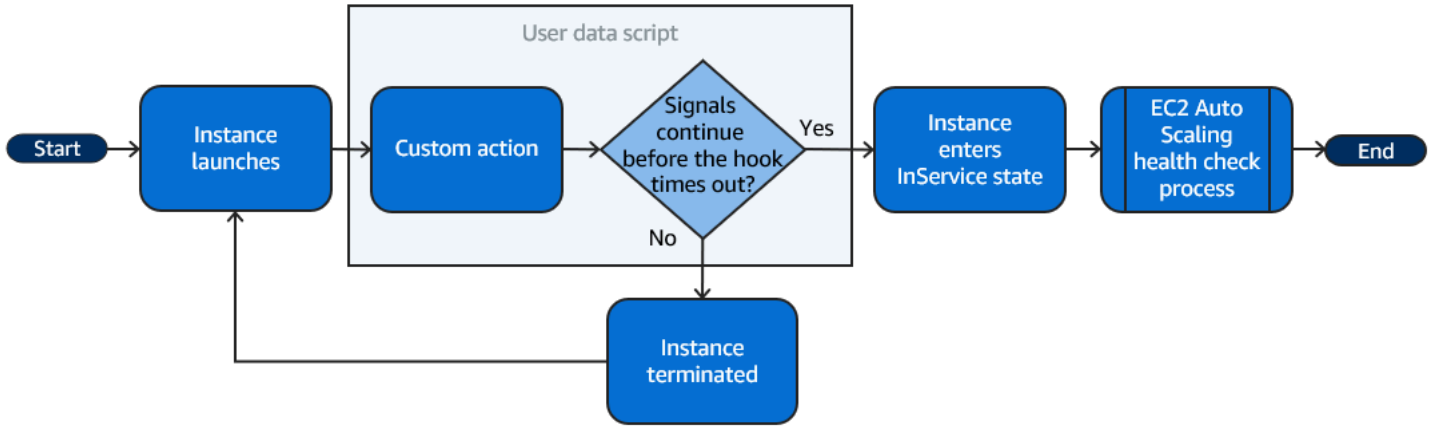
Lambda를 사용하는 경우, 함수 코드에 대한 콜백을 설정하여 사용자 지정 작업이 성공할 경우, 인스턴스의 수명 주기가 진행되도록 할 수도 있습니다. 자세한 정보는 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성을 참조하세요.

자습서: 데이터 스크립트와 인스턴스 메타데이터를 사용하여 수명 주기 상태 검색

수명 주기 후크에 대한 사용자 지정 작업을 생성하는 일반적인 방법은 Amazon EC2 Auto Scaling에서 Amazon 같은 다른 서비스에 보내는 알림을 사용하는 것입니다. EventBridge 그러나 대신 사용자 데이터 스크립트를 사용하여 인스턴스를 구성하고 수명 주기 작업을 완료하는 코드를 인스턴스 자체로 이동하면 추가 인프라를 생성하지 않아도 됩니다.

다음 자습서에서는 사용자 데이터 스크립트 및 인스턴스 메타데이터 사용을 시작하는 방법을 보여줍니다. 그룹에서 인스턴스의 대상 수명 주기 상태를 읽고 시작 프로세스를 계속하기 위해 인스턴스 수명 주기의 특정 단계에서 콜백 작업을 수행하는 사용자 데이터 스크립트를 사용하여 기본 Auto Scaling 그룹 구성을 생성합니다.

다음 그림은 사용자 데이터 스크립트를 사용하여 사용자 지정 작업을 수행할 때 발생하는 확장 이벤트의 흐름을 요약한 것입니다. 인스턴스가 시작된 후에는 제한 시간이 초과되거나 Amazon EC2 Auto Scaling에서 계속하라는 신호를 수신하여 수명 주기 후크가 완료될 때까지 인스턴스의 수명 주기가 일시 중지됩니다.



내용

- 1단계: 수명 주기 작업을 완료할 권한이 있는 IAM 역할 생성
- 2단계: 시작 템플릿 생성 및 IAM 역할과 사용자 데이터 스크립트 추가
- 3단계: Auto Scaling 그룹 생성
- 4단계: 수명 주기 후크 추가
- 5단계: 기능 테스트 및 확인
- 6단계: 정리
- 관련 리소스

1단계: 수명 주기 작업을 완료할 권한이 있는 IAM 역할 생성

AWS CLI 또는 AWS SDK를 사용하여 콜백을 전송하여 라이프사이클 작업을 완료하는 경우 라이프사이클 작업을 완료할 권한이 있는 IAM 역할을 사용해야 합니다.

정책 생성

1. IAM 콘솔에서 정책(Policies) 페이지를 열고 Create policy(정책 생성)를 선택합니다.
2. JSON 탭을 선택합니다.
3. 정책 문서(Policy Document) 상자에서 다음 정책 문서를 복사하여 상자에 붙여넣습니다. *## ###(sample text)*를 생성하려는 Auto Scaling 그룹의 계정 번호 및 이름(**TestAutoScalingEvent-group**)으로 교체합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
 "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/
TestAutoScalingEvent-group"
    }
  ]
}
```

4. 다음을 선택합니다.

5. 정책 이름(Policy name)에 **`TestAutoScalingEvent-policy`**을 입력합니다. Create policy(정책 생성)를 선택합니다.

정책 생성을 완료하면 정책을 사용하는 역할을 생성할 수 있습니다.

역할 생성

1. 왼쪽의 탐색 창에서 역할을 선택합니다.
2. 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 선택(Select trusted entity)에서 AWS 서비스(service)를 선택합니다.
4. 사용 사례에 대해 EC2를 선택하고 다음(Next)을 선택합니다.
5. 권한 추가에서 생성한 정책 (TestAutoScalingEvent-policy) 을 선택합니다. 그리고 다음을 선택합니다.
6. 이름 지정, 검토 및 생성(Name, review, and create) 페이지에서 역할 이름(Role name)에 **`TestAutoScalingEvent-role`**을 입력하고 역할 생성(Create role)을 선택합니다.

2단계: 시작 템플릿 생성 및 IAM 역할과 사용자 데이터 스크립트 추가

Auto Scaling 그룹에서 사용할 시작 템플릿을 생성합니다. 생성한 IAM 역할과 제공된 샘플 사용자 데이터 스크립트를 추가합니다.

시작 템플릿 생성

1. Amazon EC2 콘솔의 시작 템플릿 페이지를 엽니다.
2. Create launch template(시작 템플릿 생성)을 선택합니다.
3. Launch template name(시작 템플릿 이름)에 **`TestAutoScalingEvent-template`**을 입력합니다.
4. Auto Scaling guidance(Auto Scaling 지침)에서 확인란을 선택합니다.
5. 애플리케이션 및 OS 이미지(Amazon Machine Image)(Application and OS Images (Amazon Machine Image))에서 빠른 시작(Quick Start) 목록의 Amazon Linux 2(HVM), SSD 볼륨 유형, 64 비트(x86)를 선택합니다.
6. 인스턴스 유형(Instance type)에서 Amazon EC2 인스턴스 유형을 선택합니다(예: “t2.micro”).
7. Advanced details(고급 세부 정보)에서 섹션을 스케일 아웃하여 필드를 표시합니다.

8. IAM 인스턴스 프로필의 경우 IAM 역할 (-role) 의 IAM 인스턴스 프로필 이름을 선택합니다. TestAutoScalingEvent 인스턴스 프로파일은 인스턴스가 시작될 때 Amazon EC2가 인스턴스에 IAM 역할을 전달하도록 허용하는 IAM 역할을 위한 컨테이너입니다.

   IAM 콘솔을 사용하여 IAM 역할을 생성한 경우, 콘솔에서 해당 역할과 동일한 이름으로 인스턴스 프로파일을 자동으로 생성합니다.

9. 사용자 데이터(User data)에서 필드에 다음 사용자 데이터 스크립트를 붙여넣습니다. 의 샘플 텍스트를 생성하려는 Auto Scaling 그룹의 이름과 Auto Scaling 그룹에서 사용할 Auto Scaling 그룹의 이름으로 바꾸십시오. `group_name` `region` AWS 리전

```
#!/bin/bash

function get_target_state {
    echo $(curl -s http://169.254.169.254/latest/meta-data/autoscaling/target-
lifecycle-state)
}

function get_instance_id {
    echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
}

function complete_lifecycle_action {
    instance_id=$(get_instance_id)
    group_name='TestAutoScalingEvent-group'
    region='us-west-2'

    echo $instance_id
    echo $region
    echo $(aws autoscaling complete-lifecycle-action \
      --lifecycle-hook-name TestAutoScalingEvent-hook \
      --auto-scaling-group-name $group_name \
      --lifecycle-action-result CONTINUE \
      --instance-id $instance_id \
      --region $region)
}

function main {
    while true
    do
        target_state=$(get_target_state)
        if [ \"$target_state\" = \"InService\" ]; then
            # Change hostname
```

```
            export new_hostname="${group_name}-$instance_id"
            hostname $new_hostname
            # Send callback
            complete_lifecycle_action
            break
        fi
        echo $target_state
        sleep 5
    done
}

main
```

이 간단한 사용자 데이터 스크립트는 다음을 수행하십시오:

- 인스턴스 메타데이터를 호출하여 인스턴스 메타데이터에서 대상 수명 주기 상태 및 인스턴스 ID를 검색합니다.
- 대상 수명 주기 상태가 `InService`로 변경될 때까지 해당 상태를 반복적으로 검색합니다.
- 대상 수명 주기 상태가 `InService`인 경우, 인스턴스의 호스트 이름을 Auto Scaling 그룹 명칭이 앞에 추가된 인스턴스 ID로 변경합니다.
- complete-lifecycle-action CLI 명령을 호출하여 콜백을 보내 EC2 시작 프로세스를 `CONTINUE`하도록 Amazon EC2 Auto Scaling에 신호를 보냅니다.

10. Create launch template(시작 템플릿 생성)을 선택합니다.
11. 확인 페이지에서 Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

> **Note**
>
> 사용자 데이터 스크립트 개발을 위한 참조로 사용할 수 있는 다른 예제는 Amazon EC2 Auto Scaling용 GitHub 리포지토리를 참조하십시오.

3단계: Auto Scaling 그룹 생성

시작 템플릿을 생성한 후 Auto Scaling 그룹을 생성합니다.

Auto Scaling 그룹 생성

1. 시작 템플릿 또는 구성 선택 페이지에서 Auto Scaling 그룹 명칭(Auto Scaling group name)에 Auto Scaling 그룹의 이름(**`TestAutoScalingEvent-group`**)을 입력합니다.
2. 다음(Next)을 선택하여 인스턴스 시작 옵션 선택 페이지로 이동합니다.
3. 네트워크(Network)에서 VPC를 선택합니다.
4. 가용 영역 및 서브넷(Availability Zones and subnets)에 대해 하나 이상의 가용 영역에서 서브넷을 하나 이상 선택합니다.
5. 인스턴스 유형 요건(Instance type requirements) 섹션에서 기본 설정을 사용하여 이 단계를 단순화합니다. (시작 템플릿을 재정의하지 마세요.) 이 자습서에서는 시작 템플릿에 지정된 인스턴스 유형을 사용하여 하나의 온디맨드 인스턴스만 시작합니다.
6. 화면 맨 아래에 있는 검토로 건너뛰기(Skip to review)를 선택합니다.
7. 검토 페이지에서 Auto Scaling 그룹 세부 정보를 검토한 다음 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.

4단계: 수명 주기 후크 추가

수명 주기 작업이 완료될 때까지 인스턴스를 대기 상태로 유지하도록 수명 주기 후크를 추가합니다.

수명 주기 후크 추가

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다. 페이지 하단에 분할 창이 열립니다.
3. 창 하단 Instance management(인스턴스 관리) 탭의 Lifecycle hooks(수명 주기 후크)에서 Create lifecycle hook(수명 주기 후크 생성)를 선택합니다.
4. 스케일 아웃(인스턴스 실행)을 위한 수명 주기 후크를 정의하려면 다음을 수행하세요.
   a. Lifecycle hook name(수명 주기 후크 이름)으로 **`TestAutoScalingEvent-hook`**를 입력합니다.
   b. Lifecycle transition(수명 주기 전환)에서 Instance launch(인스턴스 시작)를 선택합니다.
   c. 하트비트 시간 제한(Heartbeat timeout)에 사용자 데이터 스크립트에서 콜백을 기다리는 시간(초)으로 **`300`**을 입력합니다.
   d. Default result(기본 결과)로 ABANDON(포기)을 선택합니다. 사용자 데이터 스크립트에서 콜백을 수신하지 않고 후크가 시간 초과되는 경우, Auto Scaling 그룹이 새 인스턴스를 해지합니다.

e. (옵션) 알림 메타데이터(Notification metadata)를 비워 둡니다.

5. 생성을 선택합니다.

5단계: 기능 테스트 및 확인

기능을 테스트하기 위해 Auto Scaling 그룹의 원하는 용량을 1씩 늘리면서 Auto Scaling 그룹을 업데이트합니다. 인스턴스가 시작된 직후 사용자 데이터 스크립트가 실행되고 인스턴스의 대상 수명 주기 상태를 확인하기 시작합니다. 대상 수명 주기 상태가 `InService`이면 스크립트가 호스트 이름을 변경하고 콜백 작업을 보냅니다. 완료하는 데 보통 몇 초 정도 소요됩니다.

Auto Scaling 그룹 크기 증가

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다. 위쪽 창의 맨 위 행을 계속 보면서 아래쪽 창에서 세부 정보를 확인합니다.
3. 아래 창의 Details(세부 정보) 탭에서 Group details(그룹 세부 정보), Edit(편집)을 선택합니다.
4. 원하는 용량에 대해 현재 값을 1씩 늘립니다.
5. Update(업데이트)를 선택합니다. 인스턴스가 시작되는 동안 위쪽 창의 Status(상태) 열에 Updating capacity(용량 업데이트 중) 상태가 표시됩니다.

원하는 용량을 늘린 후 인스턴스가 성공적으로 시작되었는지 그리고 크기 조정 활동의 설명에서 해지되지 않았는지 확인할 수 있습니다.

크기 조정 활동 확인

1. Auto Scaling groups(Auto Scaling 그룹) 페이지로 돌아가 그룹을 선택합니다.
2. Activity(활동) 탭의 Activity history(활동 기록)에 있는 Status(상태) 열에 Auto Scaling 그룹이 성공적으로 인스턴스를 시작했는지가 표시됩니다.
3. 사용자 데이터 스크립트가 실패하면 제한 시간이 경과한 후 상태가 `Canceled`이고 상태 메시지가 `Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result`인 크기 조정 활동이 표시됩니다.

6단계: 정리

이 자습서를 진행하기 위해 생성한 리소스로 작업을 마쳤으면 다음 단계에 따라 해당 리소스를 삭제합니다.

수명 주기 후크 삭제

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.
3. Instance management(인스턴스 관리) 탭의 Lifecycle hooks(수명 주기 후크)에서 수명 주기 후크(`TestAutoScalingEvent-hook`)를 선택합니다.
4. 작업, 삭제를 선택합니다.
5. 삭제(Delete)를 다시 선택하여 확인합니다.

시작 템플릿 삭제

1. Amazon EC2 콘솔의 시작 템플릿 페이지를 엽니다.
2. 시작 템플릿(`TestAutoScalingEvent-template`)을 선택한 다음 작업(Actions), 템플릿 삭제(Delete template)를 선택합니다.
3. 확인 메시지가 표시되면 **`Delete`**를 입력하여 지정된 시작 템플릿 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

Auto Scaling 그룹 예를 이용한 작업을 마쳤으면 해당 그룹을 삭제합니다. 생성한 IAM 역할 및 권한 정책도 삭제할 수 있습니다.

Auto Scaling 그룹 삭제

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹(`TestAutoScalingEvent-group`) 옆에 있는 확인란을 선택하고 삭제(Delete)를 선택합니다.
3. 확인 메시지가 표시되면 **`delete`**를 입력하여 지정된 Auto Scaling 그룹 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

   Name(이름) 열의 로딩 아이콘은 Auto Scaling 그룹이 삭제 중임을 나타납니다. 인스턴스를 해지하고 그룹을 삭제하는 데 몇 분 정도 걸립니다.

IAM 역할 삭제

1. IAM 콘솔에서 역할 페이지를 엽니다.
2. 함수의 역할(`TestAutoScalingEvent-role`)을 선택합니다.
3. 삭제를 선택합니다.
4. 확인 메시지가 나타나면 역할의 이름을 입력한 다음 Delete(삭제)를 선택합니다.

IAM 정책 삭제

1. IAM 콘솔에서 정책 페이지를 엽니다.
2. 생성한 정책을 선택합니다(`TestAutoScalingEvent-policy`).
3. 작업, 삭제를 선택합니다.
4. 확인 메시지가 나타나면 정책의 이름을 입력한 다음 Delete(삭제)를 선택합니다.

관련 리소스

인스턴스 메타데이터에서 사용 가능한 데이터에 근거하여 인스턴스에서 작업을 호출하는 코드를 개발할 때 다음과 같은 관련 주제가 도움이 될 수 있습니다.

- 인스턴스 메타데이터를 통해 대상 수명 주기 상태 검색. 이 섹션에서는 인스턴스 해지와 같은 다른 사용 사례의 수명 주기 상태에 대해 설명합니다.
- 수명 주기 후크 추가(콘솔). 이 절차에서는 스케일 아웃(인스턴스 시작) 및 스케일 인(인스턴스 해지 또는 웜 풀로의 복귀)을 위해 수명 주기 후크를 추가하는 방법을 설명합니다.
- Amazon EC2 사용 설명서의 인스턴스 메타데이터 카테고리. 이 주제에는 EC2 인스턴스에서 작업을 호출하는 데 사용할 수 있는 인스턴스 메타데이터의 모든 카테고리가 나열되어 있습니다.

EventBridge Amazon을 사용하여 Auto Scaling 그룹의 인스턴스에 발생하는 이벤트를 기반으로 Lambda 함수를 호출하는 규칙을 생성하는 방법을 보여주는 자습서는 을 참조하십시오. 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성

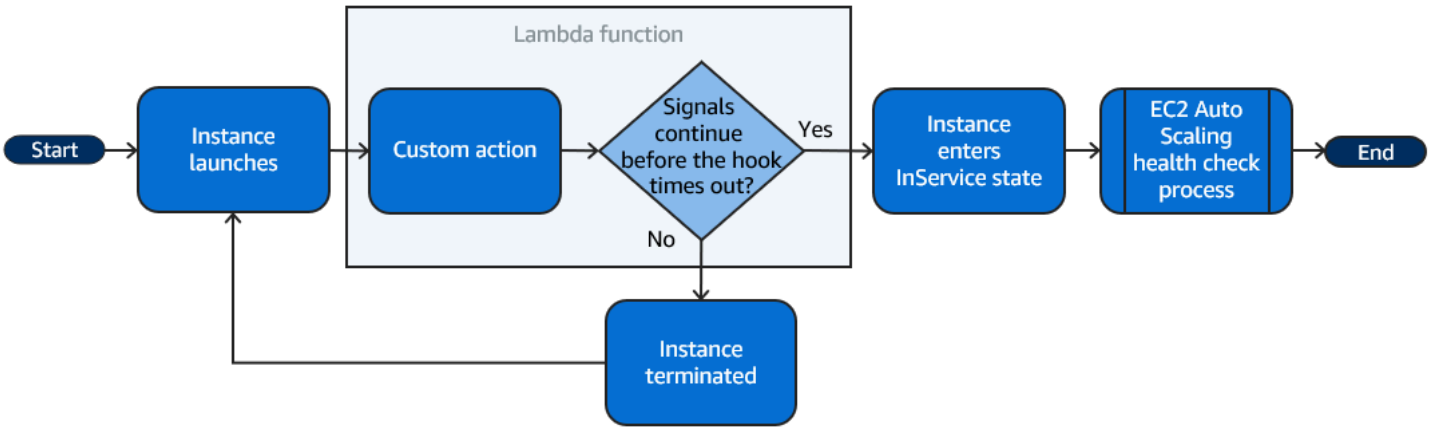
자습서: Lambda 함수를 호출하는 수명 주기 후크 구성

이 연습에서는 일치하는 경우 AWS Lambda 함수를 EventBridge 규칙 대상으로 호출하는 필터 패턴을 포함하는 Amazon 규칙을 생성합니다. 사용할 필터 패턴과 샘플 함수 코드가 제공됩니다.

모든 설정을 올바르게 구성하면 이 자습서를 마친 후 인스턴스가 시작될 때 Lambda 함수가 사용자 지정 작업을 수행합니다. 사용자 지정 작업은 Lambda 함수와 관련된 CloudWatch Logs 로그 스트림에 이벤트를 간단히 기록합니다.

또한 Lambda 함수는 이 작업에 성공할 경우, 인스턴스의 수명 주기가 진행되도록 하지만 작업에 실패하면 인스턴스가 시작을 중단하고 해지할 수 있도록 하는 콜백을 수행합니다.

다음 그림은 Lambda 함수를 사용하여 사용자 지정 작업을 수행할 때 발생하는 확장 이벤트의 흐름을 요약한 것입니다. 인스턴스가 시작된 후에는 제한 시간이 초과되거나 Amazon EC2 Auto Scaling에서 계속하라는 신호를 수신하여 수명 주기 후크가 완료될 때까지 인스턴스의 수명 주기가 일시 중지됩니다.



내용

- 사전 조건
- 1단계: 수명 주기 작업을 완료할 권한이 있는 IAM 역할 생성
- 2단계: Lambda 함수 생성
- 3단계: 규칙 생성 EventBridge
- 4단계: 수명 주기 후크 추가
- 5단계: 이벤트 테스트 및 확인
- 6단계: 정리
- 관련 리소스

사전 조건

Auto Scaling 그룹이 없는 경우, 이 자습서를 시작하기 전에 Auto Scaling 그룹을 생성합니다. Auto Scaling 그룹을 생성하려면 Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 열고 Auto Scaling 그룹 생성을 선택합니다.

1단계: 수명 주기 작업을 완료할 권한이 있는 IAM 역할 생성

Lambda 함수를 생성하기 전에 먼저 집행 역할과 사용 권한 정책을 만들어 Lambda가 수명 주기 후크를 완료할 수 있도록 해야 합니다.

정책 생성

1. IAM 콘솔에서 정책(Policies) 페이지를 열고 Create policy(정책 생성)를 선택합니다.
2. JSON 탭을 선택합니다.
3. Policy Document(정책 문서) 상자에 다음 정책 문서를 붙여 넣고 ####로 표시된 텍스트를 계정 번호 및 Auto Scaling 그룹의 이름으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:CompleteLifecycleAction"
      ],
      "Resource":
 "arn:aws:autoscaling:*:123456789012:autoScalingGroup:*:autoScalingGroupName/my-
asg"
    }
  ]
}
```

4. 다음을 선택합니다.
5. 정책 이름(Policy name)에 **LogAutoScalingEvent-policy**을 입력합니다. Create policy(정책 생성)를 선택합니다.

정책 생성을 완료하면 정책을 사용하는 역할을 생성할 수 있습니다.

역할 생성

1. 왼쪽의 탐색 창에서 역할을 선택합니다.
2. 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 선택(Select trusted entity)에서 AWS 서비스(service)를 선택합니다.
4. 사용 사례에 대해 Lambda를 선택하고 다음(Next)을 선택합니다.

5. 권한 추가에서 생성한 정책 (LogAutoScalingEvent-policy) 과 이름을 지정한 정책을 선택합니다. AWSLambdaBasicExecutionRole 그리고 다음을 선택합니다.

> **Note**
>
> AWSLambdaBasicExecutionRole정책에는 함수가 로그를 로그에 기록하는 데 필요한 권한이 있습니다. CloudWatch

6. 이름 지정, 검토 및 생성(Name, review, and create) 페이지에서 역할 이름(Role name)에 **`LogAutoScalingEvent-role`**을 입력하고 역할 생성(Create role)을 선택합니다.

2단계: Lambda 함수 생성

이벤트의 대상 역할을 할 Lambda 함수를 생성합니다. Node.js 로 작성된 샘플 Lambda 함수는 Amazon EC2 Auto Scaling에서 매칭 이벤트가 발생할 때 EventBridge 호출됩니다.

Lambda 함수 생성

1. Lambda 콘솔에서 함수 페이지를 엽니다.
2. Create function(함수 생성)과 Author from scratch(새로 작성)를 차례로 선택합니다.
3. 기본 정보(Basic information)에서 함수 이름(Function name)에 **`LogAutoScalingEvent`**을 입력합니다.
4. 런타임에서 Node.js 18.x를 선택합니다.
5. 아래로 스크롤하여 집행 역할 기본값 변경을 선택하고 집행 역할에서 기존 역할 사용을 선택합니다.
6. 기존 역할의 경우 -role을 선택합니다. LogAutoScalingEvent
7. 나머지는 기본값을 그대로 사용합니다.
8. Create function(함수 생성)을 선택합니다. 함수의 코드 및 구성으로 돌아갑니다.
9. 콘솔에 `LogAutoScalingEvent` 함수를 여전히 열어둔 상태에서 편집기의 코드 소스에서 다음 샘플 코드를 index.mjs 파일에 붙여 넣습니다.

```
import { AutoScalingClient, CompleteLifecycleActionCommand } from "@aws-sdk/client-
auto-scaling";
export const handler = async(event) => {
  console.log('LogAutoScalingEvent');
  console.log('Received event:', JSON.stringify(event, null, 2));
  var autoscaling = new AutoScalingClient({ region: event.region });
```

```
  var eventDetail = event.detail;
  var params = {
    AutoScalingGroupName: eventDetail['AutoScalingGroupName'], /* required */
    LifecycleActionResult: 'CONTINUE', /* required */
    LifecycleHookName: eventDetail['LifecycleHookName'], /* required */
    InstanceId: eventDetail['EC2InstanceId'],
    LifecycleActionToken: eventDetail['LifecycleActionToken']
  };
  var response;
  const command = new CompleteLifecycleActionCommand(params);
  try {
    var data = await autoscaling.send(command);
    console.log(data); // successful response
    response = {
      statusCode: 200,
      body: JSON.stringify('SUCCESS'),
    };
  } catch (err) {
    console.log(err, err.stack); // an error occurred
    response = {
      statusCode: 500,
      body: JSON.stringify('ERROR'),
    };
  }
  return response;
};
```

이 코드는 단순히 이벤트를 기록하므로 이 자습서의 끝에서 이 Lambda 함수와 관련된 로그 CloudWatch 로그 스트림에 이벤트가 나타나는 것을 볼 수 있습니다.

10. 배포를 선택합니다.

3단계: 규칙 생성 EventBridge

Lambda 함수를 실행하기 위한 EventBridge 규칙을 생성합니다. 사용에 대한 자세한 내용은 EventBridge 을 참조하십시오. Auto Scaling 이벤트를 처리하는 EventBridge 데 사용합니다.

콘솔을 사용하여 규칙 생성

1. EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.

4. 규칙 세부 정보 정의(Define rule detail)에 대해 다음을 수행하십시오:
   a. 이름에 **`LogAutoScalingEvent-rule`**을 입력합니다.
   b. 이벤트 버스에서 기본값을 선택합니다. AWS 서비스 계정에서 이벤트를 생성하면 해당 이벤트는 항상 계정의 기본 이벤트 버스로 이동합니다.
   c. 규칙 유형(Rule type)에서 이벤트 패턴이 있는 규칙(Rule with an event pattern)을 생성합니다.
   d. 다음을 선택합니다.
5. 이벤트 패턴 빌드(Build event pattern)에서 다음을 수행하십시오:
   a. 이벤트 소스의 경우 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
   b. 이벤트 패턴까지 아래로 스크롤하여 다음을 수행하십시오:
   c. i. 이벤트 소스(Event source)에서 AWS 서비스를 선택합니다.
      ii. AWS 서비스에서 Auto Scaling을 선택합니다.
      iii. 이벤트 유형에서 인스턴스 시작 및 해지를 선택합니다.
      iv. 기본적으로 규칙은 모든 스케일 인 또는 스케일 아웃 이벤트와 일치합니다. 스케일 아웃 이벤트가 있고 인스턴스가 수명 주기 후크로 인해 대기 상태가 될 때 알려주는 규칙을 생성하려면 특정 인스턴스 이벤트(Specific instance event(s))를 선택하고 EC2 인스턴스 시작 수명 주기 작업(EC2 Instance-launch Lifecycle Action)을 선택합니다.
      v. 기본적으로 규칙은 지역의 모든 Auto Scaling 그룹과 일치합니다. 규칙을 특정 Auto Scaling 그룹과 매칭시키려면 특정 그룹 명칭을 선택하고 해당 그룹을 선택합니다.
      vi. 다음을 선택합니다.
6. 대상 선택(Select target(s))에서 다음을 수행하십시오:
   a. 대상 유형(Target types)에서 AWS 서비스를 선택합니다.
   b. 대상 선택(Select a target)에서 Lambda 함수(Lambda function)를 선택합니다.
   c. 기능에서 선택합니다 LogAutoScalingEvent.
   d. 다음을 두 번 선택합니다
7. 검토 및 생성 페이지에서 규칙 생성을 선택합니다.

4단계: 수명 주기 후크 추가

이 섹션에서는 Lambda가 시작 시 인스턴스에 대해 함수를 실행할 수 있도록 수명 주기 후크를 추가합니다.

수명 주기 후크 추가

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다. 페이지 하단에 분할 창이 열립니다.
3. 창 하단 Instance management(인스턴스 관리) 탭의 Lifecycle hooks(수명 주기 후크)에서 Create lifecycle hook(수명 주기 후크 생성)를 선택합니다.
4. 스케일 아웃(인스턴스 실행)을 위한 수명 주기 후크를 정의하려면 다음을 수행하세요.
   a. Lifecycle hook name(수명 주기 후크 이름)으로 **`LogAutoScalingEvent-hook`**를 입력합니다.
   b. Lifecycle transition(수명 주기 전환)에서 Instance launch(인스턴스 시작)를 선택합니다.
   c. Heartbeat timeout(하트비트 시간 제한)에 Lambda 함수에서 콜백을 기다리는 시간(초)으로 **`300`**을 입력합니다.
   d. Default result(기본 결과)로 ABANDON(포기)을 선택합니다. 즉, Lambda 함수에서 콜백을 수신하지 않았는데 후크가 시간 초과되면 Auto Scaling 그룹이 새 인스턴스를 해지합니다.
   e. (옵션)Notification metadata(알림 메타데이터)를 비워 둡니다. 전달되는 이벤트 EventBridge 데이터에는 Lambda 함수를 호출하는 데 필요한 모든 정보가 들어 있습니다.
5. 생성을 선택합니다.

5단계: 이벤트 테스트 및 확인

이벤트를 테스트하기 위해 Auto Scaling 그룹의 원하는 용량을 1씩 늘리면서 Auto Scaling 그룹을 업데이트합니다. 원하는 용량을 늘린 후 몇 초 내에 Lambda 함수가 호출됩니다.

Auto Scaling 그룹 크기 증가

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택하여 아래쪽 창에 세부 정보를 표시하고 위쪽 창의 맨 위 행을 계속 표시합니다.
3. 아래 창의 Details(세부 정보) 탭에서 Group details(그룹 세부 정보), Edit(편집)을 선택합니다.
4. 원하는 용량에 대해 현재 값을 1씩 늘립니다.
5. Update(업데이트)를 선택합니다. 인스턴스가 시작되는 동안 위쪽 창의 Status(상태) 열에 Updating capacity(용량 업데이트 중) 상태가 표시됩니다.

원하는 용량을 늘린 후 Lambda 함수가 호출되었는지 확인할 수 있습니다.

Lambda 함수에서 출력 확인

1. 콘솔의 로그 그룹 페이지를 엽니다. CloudWatch
2. Lambda 함수에 대한 로그 그룹 명칭을 선택합니다(`/aws/lambda/LogAutoScalingEvent`).
3. 로그 스트림 이름을 선택하여 수명 주기 작업에 대해 해당 함수에서 제공하는 데이터를 확인합니다.

다음으로, 크기 조정 활동의 열거을 통해 인스턴스가 성공적으로 시작되었는지 확인할 수 있습니다.

크기 조정 활동 확인

1. Auto Scaling groups(Auto Scaling 그룹) 페이지로 돌아가 그룹을 선택합니다.
2. Activity(활동) 탭의 Activity history(활동 기록)에 있는 Status(상태) 열에 Auto Scaling 그룹이 성공적으로 인스턴스를 시작했는지가 표시됩니다.
   - 작업에 성공하면 크기 조정 활동의 상태가 “성공”으로 표시됩니다.
   - 실패하면 몇 분이 지난 후 “Cancelled(취소됨)” 상태의 크기 조정 활동이 표시되고 “Instance failed to complete user's Lifecycle Action: Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42EXAMPLE was abandoned: Lifecycle Action Completed with ABANDON Result(인스턴스가 사용자의 수명 주기 작업을 완료하지 못함: 토큰이 e85eb647-4fe0-4909-b341-a6c42EXAMPLE인 수명 주기 작업이 중단됨: 수명 주기 작업 완료되었으나 결과가 중단임)”이라는 상태 메시지가 표시됩니다.

Auto Scaling 그룹 크기 감소

이 테스트를 위해 시작한 추가 인스턴스가 필요하지 않으면 Details(세부 정보) 탭을 열고 원하는 용량을 1씩 줄일 수 있습니다.

6단계: 정리

이 자습서 전용으로 생성한 리소스 사용을 마친 경우, 다음 단계에 따라 리소스를 삭제합니다.

수명 주기 후크 삭제

1. Amazon EC2 콘솔에서 Auto Scaling 그룹 페이지를 엽니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.
3. Instance management(인스턴스 관리) 탭의 Lifecycle hooks(수명 주기 후크)에서 수명 주기 후크(`LogAutoScalingEvent-hook`)를 선택합니다.

4. 작업, 삭제를 선택합니다.
5. 삭제(Delete)를 다시 선택하여 확인합니다.

Amazon EventBridge 규칙을 삭제하려면

1. Amazon EventBridge 콘솔에서 규칙 페이지를 엽니다.
2. Event bus(이벤트 버스)에서 규칙과 연결된 이벤트 버스(`Default`)를 선택합니다.
3. 규칙(`LogAutoScalingEvent-rule`) 옆에 있는 확인란을 선택합니다.
4. 삭제를 선택합니다.
5. 확인 메시지가 나타나면 규칙의 이름을 입력한 다음 Delete(삭제)를 선택합니다.

예 함수를 사용한 작업이 완료되면 해당 함수를 삭제하세요. 함수의 로그를 저장하는 로그 그룹과 생성한 집행 역할 및 정책 권한을 삭제할 수도 있습니다.

Lambda 함수 삭제

1. Lambda 콘솔에서 함수 페이지를 엽니다.
2. 함수(`LogAutoScalingEvent`)를 선택합니다.
3. 작업, 삭제를 선택합니다.
4. 확인 메시지가 표시되면 **`delete`**를 입력하여 지정된 함수 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

로그 그룹 삭제

1. CloudWatch 콘솔의 로그 그룹 페이지를 엽니다.
2. 함수의 로그 그룹(`/aws/lambda/LogAutoScalingEvent`)을 선택합니다.
3. 작업(Actions), 로그 그룹 삭제(Delete log group(s))를 선택합니다.
4. 로그 그룹 삭제(Delete log group(s)) 대화 상자에서 삭제(Delete)를 선택합니다.

집행 역할 삭제

1. IAM 콘솔에서 역할 페이지를 엽니다.
2. 함수의 역할(`LogAutoScalingEvent-role`)을 선택합니다.
3. 삭제를 선택합니다.

4. 확인 메시지가 나타나면 역할의 이름을 입력한 다음 Delete(삭제)를 선택합니다.

IAM 정책 삭제

1. IAM 콘솔에서 정책 페이지를 엽니다.
2. 생성한 정책을 선택합니다(`LogAutoScalingEvent-policy`).
3. 작업, 삭제를 선택합니다.
4. 확인 메시지가 나타나면 정책의 이름을 입력한 다음 Delete(삭제)를 선택합니다.

관련 리소스

다음 관련 항목은 Auto Scaling 그룹의 인스턴스에 발생하는 이벤트를 기반으로 EventBridge 규칙을 생성할 때 유용할 수 있습니다.

- Auto Scaling 이벤트를 처리하는 EventBridge 데 사용합니다.. 이 섹션에서는 스케일 인을 위한 이벤트를 포함하여 다른 사용 사례에 대한 이벤트의 예를 보여줍니다.
- 수명 주기 후크 추가(콘솔). 이 절차에서는 스케일 아웃(인스턴스 시작) 및 스케일 인(인스턴스 해지 또는 웜 풀로의 복귀)을 위해 수명 주기 후크를 추가하는 방법을 설명합니다.

인스턴스 메타데이터 서비스(IMDS)를 사용하여 인스턴스 자체 내에서 작업을 호출하는 방법을 보여주는 자습서은 자습서: 데이터 스크립트와 인스턴스 메타데이터를 사용하여 수명 주기 상태 검색을 참조하세요.

웜 풀을 사용하여 부팅 시간이 긴 애플리케이션의 지연 시간을 줄입니다.

웜 풀을 사용하면 예컨대, 인스턴스가 많은 양의 데이터를 디스크에 기록해야 하기 때문에 부팅 시간이 매우 길어지는 애플리케이션의 지연 시간을 줄일 수 있습니다. 웜 풀을 사용하면 애플리케이션 성능 향상을 위해 대기 시간을 관리하려고 Auto Scaling 그룹을 과도하게 프로비저닝할 필요가 없습니다. 자세한 내용은 Auto Scaling 웜 풀을 사용하여 EC2 애플리케이션을 더 빠르게 확장하는 다음 블로그 게시물을 참조하십시오.

> **Important**
>
> 필요하지 않은 경우, 웜 풀을 생성하면 불필요한 비용이 발생할 수 있습니다. 첫 번째 부팅 시간으로 인해 애플리케이션에서 눈에 띄는 대기 시간 문제가 발생하지 않는 경우, 웜 풀을 사용할 필요가 없습니다.

주제

- 핵심 개념
- 사전 조건
- 웜 풀에서 인스턴스 업데이트
- 관련 리소스
- 제한 사항
- Auto Scaling 그룹의 웜 풀과 함께 라이프사이클 후크 사용
- Auto Scaling 그룹을 위한 웜 풀 생성
- 건전성 체크의 상태 및 건전성 체크 불합격 이유 보기
- 를 사용하여 웜 풀을 만들고 관리하는 예 AWS CLI

핵심 개념

시작하기 전에 다음과 같은 핵심 개념을 익힙니다.

웜 풀

웜 풀은 Auto Scaling 그룹과 나란히 있는 사전 초기화된 EC2 인스턴스 풀입니다. 애플리케이션을 스케일 아웃해야 할 때마다 Auto Scaling 그룹은 원하는 새 용량을 충족하기 위해 웜 풀을 사용할 수 있습니다. 이렇게 하면 인스턴스가 애플리케이션 트래픽 제공을 신속하게 시작하여 스케일 아웃 이벤트에 대한 응답 속도를 높일 수 있습니다. 인스턴스가 웜 풀을 벗어나면 그룹의 원하는 용량에 가산됩니다. 이를 웜 스타트라고 합니다.

인스턴스가 웜 풀에 있는 동안 조정 정책은 `InService` 상태에 있는 인스턴스의 지표 값이 대상 추적 조정 정책의 대상 사용률과 동일한 조정 정책의 경보 상한 임계값보다 큰 경우에만 스케일 아웃됩니다.

웜 풀 크기

기본적으로 웜 풀의 크기는 Auto Scaling 그룹의 최대 용량과 원하는 용량 간의 차이로 계산됩니다. 예컨대, Auto Scaling 그룹의 원하는 용량이 6이고 최대 용량이 10인 경우, 웜 풀을 처음 설정하고 풀을 초기화할 때 웜 풀의 크기는 4가 됩니다.

웜 풀의 최대 용량을 개별적으로 지정하려면 custom specification (`MaxGroupPreparedCapacity`) 옵션을 사용하고 그룹의 현재 용량보다 큰 사용자 지정 값을 설정합니다. 사용자 지정 값을 제공하는 경우 웜 풀의 크기는 사용자 지정 값과 그룹의 현재 원하는 용량 간의 차이로 계산됩니다. 예를 들어, Auto Scaling 그룹의 원하는 용량이 6이고, 최대 용량이 20이고, 사용자 지정 값이 8인 경우, 웜 풀을 처음 설정하고 풀을 초기화할 때 웜 풀의 크기는 2가 됩니다.

대규모 Auto Scaling 그룹으로 작업할 때는 사용자 지정 사양 (`MaxGroupPreparedCapacity`) 옵션만 사용하여 웜 풀 사용에 따른 비용 이점을 관리하면 될 수 있습니다. 예컨대, 최대 용량이 1,500(비상 트래픽 급증 시 추가 용량 제공용)이고, 100개의 인스턴스로 구성된 웜 풀이 있는, 인스턴스 1,000개의 Auto Scaling 그룹은 웜 풀 내에 향후 사용을 위해 예약된 인스턴스 500개를 유지하는 것보다 목표를 더 잘 달성할 수 있습니다.

최소 웜 풀 크기

최소 크기 설정을 사용해 웜 풀에 유지할 최소 인스턴스 수를 정적으로 설정하는 것이 좋습니다. 기본적으로 설정된 최소 크기는 없습니다.

웜 풀 인스턴스 상태

웜 풀의 인스턴스를 `Stopped`, `Running`, `Hibernated`의 세 가지 상태 중 하나로 유지할 수 있습니다. `Stopped` 상태로 인스턴스를 유지하는 것이 비용을 최소화하는 효과적인 방법입니다. 중지된 인스턴스의 경우, 사용하는 볼륨과 인스턴스에 연결된 탄력적 IP 주소에 대해서만 요금을 지불합니다.

또는 메모리 콘텐츠를 삭제하지 않고도 인스턴스를 중지한 `Hibernated` 상태로 인스턴스를 유지할 수 있습니다 (RAM). 인스턴스가 하이버네이션되면 운영 체제에 Amazon EBS 루트 볼륨에 사용자의 콘텐츠를 RAM 저장하라는 신호를 보냅니다. 인스턴스가 다시 시작되면 루트 볼륨이 이전 상태로 복원되고 RAM 콘텐츠가 다시 로드됩니다. 인스턴스가 하이버네이션에 있는 동안에는 RAM 콘텐츠 스토리지와 인스턴스에 연결된 엘라스틱 IP 주소를 포함한 EBS 볼륨에 대한 비용만 지불하면 됩니다.

웜 풀 내에서 인스턴스를 `Running` 상태로 유지하는 것도 가능하지만 불필요한 요금이 발생하지 않도록 하는 것이 좋습니다. 인스턴스가 중지되거나 최대 절전 모드로 전환되면 인스턴스 자체의 비용이 절감됩니다. 인스턴스가 실행 중일 때만 해당 인스턴스에 대한 요금을 지불합니다.

수명 주기 후크

수명 주기 후크를 사용하여 인스턴스를 대기 상태에 놓음으로써 인스턴스에서 사용자 지정 작업을 수행할 수 있습니다. 사용자 지정 작업은 인스턴스가 시작될 때 또는 해지되기 전에 수행됩니다.

또한 웜 풀 구성에서 수명 주기 후크는 인스턴스의 초기화가 완료될 때까지 스케일 아웃 이벤트 중에 인스턴스가 중지 또는 동면되거나 서비스 상태가 되지 않도록 인스턴스를 지연시킬 수 있습니다. 수명 주기 후크 없이 Auto Scaling 그룹에 웜 풀을 추가하는 경우, 초기화를 완료하는 데 시간이 오래 걸리는 인스턴스는 준비가 되기 전에 스케일 아웃 이벤트 동안 중지되거나 최대 절전 모드로 전환된 다음 서비스 상태가 될 수 있습니다.

인스턴스 재사용 정책

기본적으로 Amazon EC2 Auto Scaling은 Auto Scaling 그룹이 축소되면 인스턴스를 종료합니다. 그런 다음 웜 풀로 새 인스턴스를 시작하여 해지된 인스턴스를 교체합니다.

그러지 않고 웜 풀로 인스턴스를 반환하려는 경우, 인스턴스 재사용 정책을 지정할 수 있습니다. 이렇게 하면 이미 애플리케이션 트래픽을 처리하도록 구성된 인스턴스를 재사용할 수 있습니다. 웜 풀이 과도하게 프로비저닝되지 않도록 Amazon EC2 Auto Scaling은 설정에 따라 필요 이상으로 커지면 웜 풀의 인스턴스를 종료하여 크기를 줄일 수 있습니다. 웜 풀에서 인스턴스를 해지할 때 웜 풀은 기본 해지 정책을 사용하여 먼저 해지할 인스턴스를 선택합니다.

> **Important**
>
> 축소 시 인스턴스를 최대 절전 모드로 전환하려는 경우, Auto Scaling 그룹에 기존 인스턴스가 있으면 인스턴스가 인스턴스 최대 절전 모드에 대한 요건을 충족해야 합니다. 그렇지 않으면 인스턴스가 웜 풀로 반환될 때 최대 절전 모드 대신 중지 상태로 전환됩니다.

> **Note**
>
> 현재는 또는 an을 사용해야만 인스턴스 재사용 정책을 지정할 수 있습니다. AWS CLI SDK 콘솔에서는 이 기능을 사용할 수 없습니다.

사전 조건

Auto Scaling 그룹에 대한 웜 풀을 만들기 전에 수명 주기 후크를 사용하여 새 인스턴스를 적절한 초기 상태로 초기화할 방법을 결정하세요.

수명 주기 후크로 인해 대기 상태에 있는 인스턴스에 대해 사용자 지정 작업을 수행하려면 다음 두 가지 옵션이 있습니다.

- 시작 시 인스턴스에서 명령을 실행하는 간단한 시나리오의 경우, Auto Scaling 그룹에 대한 시작 템플릿 또는 시작 구성을 생성할 때 사용자 데이터 스크립트를 포함할 수 있습니다. 사용자 데이터 스크립트는 인스턴스를 시작할 때 cloud-init을 통해 실행되는 단순한 일반 셸 스크립트 또는 cloud-init 명령입니다. 이 스크립트는 스크립트가 실행되는 인스턴스의 ID를 사용하여 인스턴스가 다음 상태로 전환되는 시기를 제어할 수도 있습니다. 아직 없는 경우, 인스턴스 메타데이터에서 인스턴스의 인스턴스 ID를 검색하도록 스크립트를 업데이트합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 검색을 참조하십시오.

  > **Tip**
  >
  > 인스턴스를 다시 시작할 때 사용자 데이터 스크립트를 실행하려면 사용자 데이터가 MIME 멀티파트 형식이어야 하며 사용자 데이터 `#cloud-config` 섹션에 다음을 지정해야 합니다.
  >
  > ```
  > #cloud-config
  > cloud_final_modules:
  > - [scripts-user, always]
  > ```

- 인스턴스가 웜 풀에 들어오거나 나갈 때 작업을 수행하는 등 AWS Lambda 서비스가 필요한 고급 시나리오의 경우, Auto Scaling 그룹에 대한 수명 주기 후크를 생성하고 수명 주기 알림을 기반으로 사용자 지정 작업을 수행하도록 대상 서비스를 구성할 수 있습니다. 자세한 내용은 지원되는 알림 대상 단원을 참조하십시오.

인스턴스의 최대 절전 모드 준비

`Hibernated`풀 상태를 사용하도록 Auto Scaling 인스턴스를 준비하려면 Amazon User Guide의 하이버네이션 사전 요구 사항 항목에 설명된 대로 인스턴스 하이버네이션을 지원하도록 올바르게 설정된 새 시작 템플릿 또는 시작 구성을 생성하십시오. EC2 그런 다음 새 시작 템플릿 또는 시작 구성을 Auto Scaling 그룹과 연결하고 인스턴스 새로 고침을 시작하여 이전 시작 템플릿 또는 시작 구성과 연결된 인스턴스를 교체합니다. 자세한 설명은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다. 섹션을 참조하세요.

웜 풀에서 인스턴스 업데이트

웜 풀의 인스턴스를 업데이트하려면 새 시작 템플릿 또는 시작 구성을 만들어 Auto Scaling 그룹에 연결하면 됩니다. 모든 새 인스턴스는 시작 템플릿 또는 시작 구성에 지정된 새 업데이트 AMI 및 기타 업데이트를 사용하여 시작되지만 기존 인스턴스는 영향을 받지 않습니다.

새 시작 템플릿 또는 시작 구성을 사용하는 교체 웜 풀 인스턴스가 강제로 시작되도록 하려면 인스턴스 새로 고침을 시작하여 그룹의 롤링 업데이트를 수행하면 됩니다. 인스턴스 새로 고침은 먼저 `InService` 인스턴스를 교체합니다. 그런 다음 웜 풀의 인스턴스를 교체합니다. 자세한 설명은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다. 섹션을 참조하세요.

관련 리소스

GitHub리포지토리를 방문하여 웜 풀용 라이프사이클 후크의 예를 확인할 수 있습니다.

제한 사항

- 혼합 인스턴스 정책이 있는 Auto Scaling 그룹에는 웜 풀을 추가할 수 없습니다. 또한 스팟 인스턴스를 요청하는 시작 템플릿 또는 시작 구성이 있는 Auto Scaling 그룹에는 웜 풀을 추가할 수 없습니다.
- Amazon EC2 Auto Scaling은 Amazon EBS 볼륨을 루트 디바이스로 사용하는 경우에만 인스턴스를 `Stopped` 또는 `Hibernated` 상태로 설정할 수 있습니다. 루트 디바이스에 인스턴스 스토어를 사용하는 인스턴스는 중지하거나 최대 전절 모드로 전환할 수 없습니다.
- Amazon EC2 Auto Scaling은 Amazon 사용 설명서의 하이버네이션 사전 요구 사항 항목에 나열된 요구 사항을 모두 충족하는 경우에만 인스턴스를 *`Hibernated`* 상태로 만들 수 있습니다. EC2
- 스케일 아웃 이벤트가 있을 때 웜 풀이 고갈되면 인스턴스는 Auto Scaling 그룹으로 바로 시작됩니다(콜드 스타트). 또한 가용 영역의 용량이 부족할 경우, 콜드 스타트가 발생할 수도 있습니다.
- 시작 프로세스 중에 웜 풀 내 인스턴스에 문제가 발생하여 해당 `InService` 상태가 되지 않는 경우 해당 인스턴스는 시작 실패로 간주되어 종료됩니다. 이는 용량 부족 오류나 기타 요인과 같은 근본 원인과 상관없이 적용됩니다.
- Amazon Elastic Kubernetes Service (Amazon) 관리형 노드 그룹과 함께 웜 풀을 사용하려는 경우 아직 초기화 중인 인스턴스가 EKS Amazon 클러스터에 등록될 수 있습니다. EKS 따라서 클러스터가 중지 또는 최대 절전 모드를 준비할 때 인스턴스에 대한 작업을 예약하게 될 수 있습니다.
- 마찬가지로 Amazon ECS 클러스터에서 웜 풀을 사용하려고 하면 초기화가 완료되기 전에 인스턴스가 클러스터에 등록될 수 있습니다. 이 문제를 해결하려면 사용자 데이터에 특수 에이전트 구성 변수를 포함하는 시작 템플릿 또는 시작 구성을 구성해야 합니다. 자세한 설명은 Amazon Elastic Container Service 개발자 안내서의 Auto Scaling 그룹의 웜 풀 사용을 참조하세요.

- 웜 풀에 대한 최대 절전 모드 지원은 AWS 리전 Amazon EC2 Auto Scaling 및 하이버네이션을 사용할 수 있는 모든 상업용에서 사용할 수 있습니다. 단, 다음과 같은 경우는 예외입니다.
  - 아시아 태평양(하이데라바드)
  - 아시아 태평양(멜버른)
  - 캐나다 서부(캘거리)
  - 중국(베이징) 리전
  - 중국(닝샤) 리전
  - 유럽(스페인)
  - 이스라엘(텔아비브)

Auto Scaling 그룹의 웜 풀과 함께 라이프사이클 후크 사용

웜 풀의 인스턴스는 각 전환에 적절한 사용자 지정 작업을 생성하는 데 도움이 되도록 독립적인 자체 수명 주기를 유지합니다. 이 수명 주기는 인스턴스가 아직 초기화 중이고 서비스 상태가 되기 전에 대상 서비스에서 작업(예: Lambda 함수)을 호출하는 데 도움이 되도록 설계되었습니다.

> **Note**
> 수명 주기 후크를 추가 및 관리하고 수명 주기 API 작업을 완료하는 데 사용하는 작업은 변경되지 않습니다. 인스턴스 수명 주기만 변경됩니다.

수명 주기 후크 사용에 대한 자세한 설명은 Auto Scaling 그룹에 라이프사이클 후크 추가 섹션을 참조하세요. 수명 주기 작업 완료에 대한 자세한 설명은 Auto Scaling 그룹에서 라이프사이클 작업 완료 섹션을 참조하세요.

웜 풀에 들어가는 인스턴스의 경우, 다음 이유 중 하나로 인해 수명 주기 후크가 필요할 수 있습니다.

- 초기화를 완료하는 데 시간이 오래 걸리는 EC2 인스턴스에서 인스턴스를 AMI 시작하려고 합니다.
- 사용자 데이터 스크립트를 실행하여 인스턴스를 부트스트랩하려고 합니다. EC2

웜 풀에서 나가는 인스턴스의 경우, 다음 이유 중 하나로 인해 수명 주기 후크가 필요할 수 있습니다.

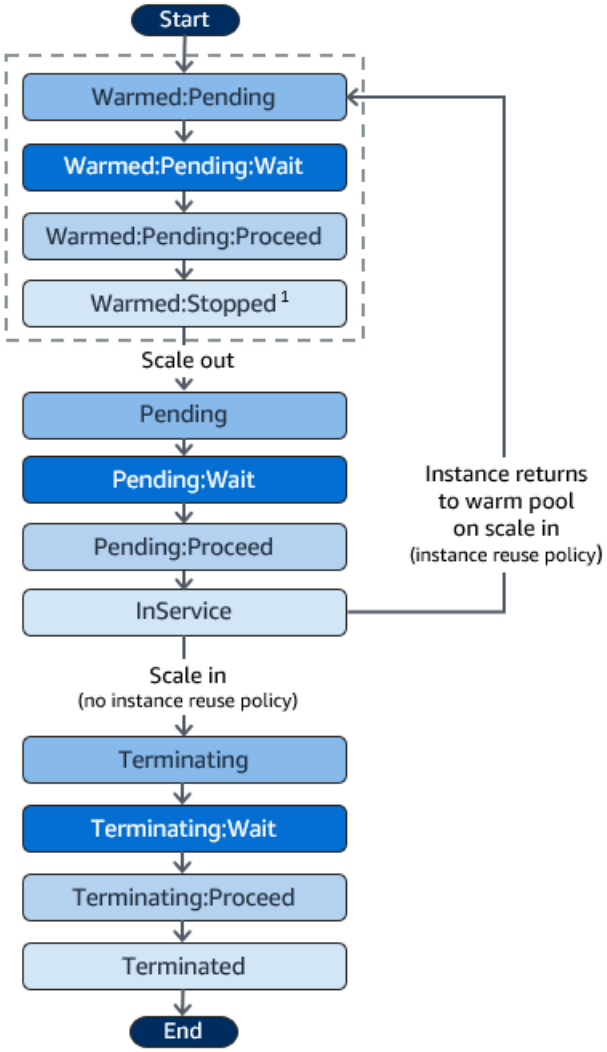
- 추가 시간을 할애하여 EC2 인스턴스를 사용할 수 있도록 준비할 수 있습니다. 예컨대, 애플리케이션의 올바른 작동을 위해 인스턴스를 다시 시작할 때 반드시 시작해야 하는 서비스가 있을 수 있습니다.

- 새 서버가 비어 있는 캐시로 시작되지 않도록 캐시 데이터를 미리 채우려고 합니다.
- 새 인스턴스를 구성 관리 서비스에 관리형 인스턴스로 등록하려고 합니다.

웜 풀의 인스턴스에 대한 수명 주기 상태 전환

Auto Scaling 인스턴스는 수명 주기의 일부로 여러 가지 상태로 전환할 수 있습니다.

다음 다이어그램은 웜 풀을 사용할 때 Auto Scaling 상태 간의 전환을 보여줍니다.

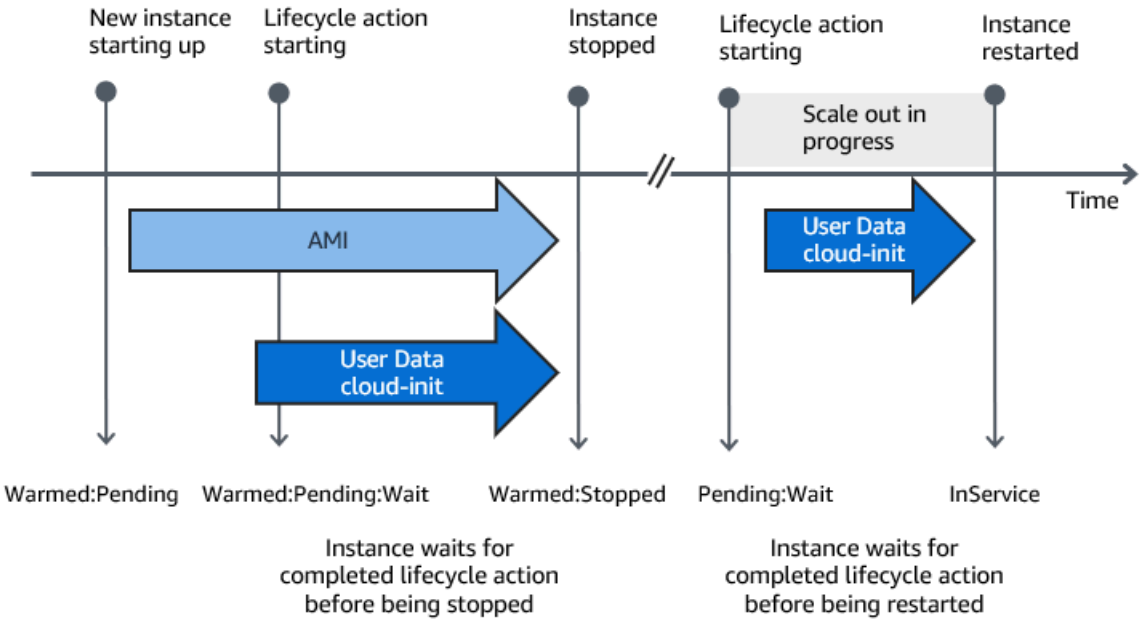


[1] 이 상태는 웜 풀의 풀 상태 설정에 따라 다릅니다. 풀 상태가 `Running`으로 설정된 경우는 이 상태는 `Warmed:Running`이 됩니다. 풀 상태가 `Hibernated`로 설정된 경우, 이 상태는 `Warmed:Hibernated`가 됩니다.

수명 주기 후크를 추가할 때 다음 사항을 고려하세요.

- `autoscaling:EC2_INSTANCE_LAUNCHING` 수명 주기 작업에 대해 수명 주기 후크가 구성된 경우, 새로 시작된 인스턴스가 `Warmed:Pending:Wait` 상태에 도달하면 먼저 일시 중지하여 사용자 지정 작업을 수행하고 인스턴스가 다시 시작되어 `Pending:Wait` 상태에 도달합니다.
- `EC2_INSTANCE_TERMINATING` 수명 주기 작업에 대해 수명 주기 후크가 구성된 경우, 해지 인스턴스가 `Terminating:Wait` 상태에 도달하면 일시 중지하여 사용자 지정 작업을 수행합니다. 하지만 인스턴스를 해지하지 않고 스케일 인 시 웜 풀에 반환하도록 인스턴스 재사용 정책을 지정하는 경우, `EC2_INSTANCE_TERMINATING` 수명 주기 작업에 대해 `Warmed:Pending:Wait` 상태에서 웜 풀에 반환한 인스턴스가 일시 중지되어 사용자 지정 작업을 수행합니다.
- 애플리케이션에 대한 수요로 인해 웜 풀이 고갈되는 경우, Amazon EC2 Auto Scaling은 Auto Scaling 그룹에서 인스턴스를 직접 시작할 수 있습니다. 단, 그룹의 용량이 아직 최대 용량에 도달하지 않은 경우에 한합니다. 인스턴스가 그룹으로 직접 시작되는 경우, `Pending:Wait` 상태에서 사용자 지정 작업을 수행하기 위해서만 일시 중지됩니다.
- 인스턴스가 다음 상태로 전환되기 전에 대기 상태를 유지하는 기간을 제어하려면 complete-lifecycle-action 명령을 사용하도록 사용자 지정 작업을 구성합니다. 수명 주기 후크를 사용하면 Amazon EC2 Auto Scaling에 지정된 수명 주기 작업이 완료되었음을 알릴 때까지 또는 제한 시간이 종료될 때까지 (기본적으로 1시간) 인스턴스가 대기 상태로 유지됩니다.

다음은 스케일 아웃 이벤트의 흐름을 요약한 것입니다.



인스턴스가 대기 상태에 도달하면 Amazon EC2 Auto Scaling에서 알림을 보냅니다. 이러한 알림의 예는 이 가이드의 EventBridge 섹션에서 확인할 수 있습니다. 자세한 내용은 웜 풀 예 이벤트 및 패턴 단원을 참조하십시오.

지원되는 알림 대상

Amazon EC2 Auto Scaling은 다음 중 하나를 수명 주기 알림의 알림 대상으로 정의할 수 있도록 지원합니다.

- EventBridge 규칙
- 아마존 SNS 토픽
- 아마존 SQS 대기열

> **Important**
> 시작 템플릿 또는 시작 구성에 인스턴스가 시작될 때 인스턴스를 구성하는 사용자 데이터 (cloud-init) 스크립트가 있는 경우, 시작하거나 다시 시작하고 있는 인스턴스에 사용자 지정 작업을 수행하기 위해 알림을 받을 필요가 없습니다.

다음 섹션에는 알림 대상을 구성하는 방법을 설명하는 문서에 대한 링크가 포함되어 있습니다.

EventBridge 규칙: Amazon EC2 Auto Scaling이 인스턴스를 대기 상태로 만들 때 코드를 실행하려면 EventBridge 규칙을 생성하고 Lambda 함수를 대상으로 지정하면 됩니다. 수명 주기 알림에 따라 서로 다른 Lambda 함수를 호출하려면 여러 규칙을 생성하고 각 규칙을 특정 이벤트 패턴 및 Lambda 함수에 연결할 수 있습니다. 자세한 내용은 웜풀 이벤트에 대한 EventBridge 규칙 생성 단원을 참조하십시오.

Amazon SNS 주제: 인스턴스가 대기 상태로 전환되었을 때 알림을 받으려면 Amazon SNS 주제를 생성한 다음, SNS 메시지 속성에 따라 수명 주기 알림을 다르게 전송하도록 Amazon 메시지 필터링을 설정합니다. 자세한 내용은 Amazon SNS를 사용하여 알림 수신 단원을 참조하십시오.

아마존 SQS 대기열: 관련 소비자가 라이프사이클 알림을 수신하여 처리할 수 있는 배송 지점을 설정하려면 Amazon SQS 대기열과 대기열에서 메시지를 처리하는 대기열 소비자를 생성하면 됩니다. SQS 대기열 소비자가 메시지 속성에 따라 수명 주기 알림을 다르게 처리하도록 하려면 메시지를 구문 분석한 다음 특정 속성이 원하는 값과 일치하면 메시지에 대해 조치를 취하도록 대기열 소비자를 설정해야 합니다. 자세한 설명은 Amazon SQS 사용하여 알림 수신 섹션을 참조하세요.

Auto Scaling 그룹을 위한 웜 풀 생성

이 주제에서는 Auto Scaling 그룹을 위한 웜 풀을 생성하는 방법을 설명합니다.

> **Important**
>
> 계속하기 전에 웜 풀을 만들기 위한 전제 조건을 완료하고 Auto Scaling 그룹에 대한 수명 주기 후크를 만들었는지 확인하세요.

웜 풀 생성

다음 절차에 따라 Auto Scaling 그룹에 대한 웜 풀을 생성할 수 있습니다.

웜 풀을 생성하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. Instance management(인스턴스 관리) 탭을 선택합니다.
4. Warm pool(웜 풀)에서 Create warm pool(웜 풀 만들기)을 선택합니다.
5. 웜 풀을 구성하려면 다음을 수행하십시오:

   a. Warm pool instance state(웜 풀 인스턴스 상태)에서 웜 풀에 들어갈 때 인스턴스를 전환할 상태를 선택합니다. 기본값은 `Stopped`입니다.

   b. Minimum warm pool size(최소 웜 풀 크기)에 웜 풀에서 유지할 최소 인스턴스 수를 입력합니다.

   c. 인스턴스 재사용의 경우 스케일 인 시 재사용 확인란을 선택하여 Auto Scaling 그룹의 인스턴스가 스케일 인 시 웜 풀로 돌아갈 수 있도록 합니다.

   d. 웜 풀 크기의 경우 사용 가능한 옵션 중 하나를 선택합니다.

      - 기본 사양: 웜 풀의 크기는 Auto Scaling 그룹의 최대 용량과 원하는 용량의 차이에 따라 결정됩니다. 이 옵션은 웜 풀 관리를 간소화합니다. 웜 풀을 만든 후에는 그룹의 최대 용량을 조정하기만 하면 웜 풀 크기를 쉽게 업데이트할 수 있습니다.
      - 사용자 지정 사양: 웜 풀의 크기는 사용자 지정 값과 Auto Scaling 그룹의 원하는 용량 간의 차이에 따라 결정됩니다. 이 옵션을 사용하면 웜 풀의 크기를 그룹의 최대 용량과 독립적으로 유연하게 관리할 수 있습니다.

6. 현재 설정을 기반으로 한 예상 웜 풀 크기 섹션을 보고 기본 또는 사용자 지정 사양이 웜 풀 크기에 어떻게 적용되는지 확인하십시오. 웜 풀 크기는 Auto Scaling 그룹의 원하는 용량에 따라 달라지며, 그룹이 확장되면 용량이 변경된다는 점을 기억하십시오.
7. 생성(Create)을 선택합니다.

웜 풀 삭제

더 이상 웜 풀이 필요하지 않으면 다음 절차에 따라 삭제합니다.

웜 풀을 삭제하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. Instance management(인스턴스 관리) 탭을 선택합니다.
4. 웜 풀(Warm pool)에서 작업(Actions), 삭제(Delete)를 선택합니다.
5. 확인 메시지가 나타나면 삭제를 선택합니다.

건전성 체크의 상태 및 건전성 체크 불합격 이유 보기

상태 확인을 통해 Amazon EC2 Auto Scaling은 인스턴스가 비정상이어서 종료되어야 하는 시기를 판단할 수 있습니다. 한 `Stopped` 상태로 유지되는 웜 풀 인스턴스의 경우 Amazon 이 EBS 보유한 `Stopped` 인스턴스 가용성을 파악하여 비정상 인스턴스를 식별합니다. 이를 `DescribeVolumeStatus` API 위해 를 호출하여 인스턴스에 연결된 EBS 볼륨의 상태를 확인합니다. 웜 풀 인스턴스가 특정 `Running` 상태로 유지되는 경우 EC2 상태 검사를 통해 인스턴스 상태를 확인합니다. 웜 풀 인스턴스에는 상태 확인 유예 기간이 없지만 Amazon EC2 Auto Scaling은 수명 주기 후크가 완료될 때까지 인스턴스 상태 확인을 시작하지 않습니다.

인스턴스가 비정상인 것으로 확인되면 Amazon EC2 Auto Scaling은 비정상 인스턴스를 자동으로 삭제하고 이를 대체할 새 인스턴스를 생성합니다. 인스턴스는 일반적으로 건전성 체크에 실패한 후 몇 분 이내에 해지됩니다. 자세한 설명은 상태 확인 불합격 이유 확인을 참조하세요.

사용자 지정 건전성 체크도 지원됩니다. 이는 인스턴스의 상태를 감지하고 이 정보를 Amazon EC2 Auto Scaling으로 전송할 수 있는 자체 상태 점검 시스템이 있는 경우 유용할 수 있습니다. 자세한 내용은 Auto Scaling 그룹에 대한 사용자 지정 상태 점검 설정 단원을 참조하십시오.

Amazon EC2 Auto Scaling 콘솔에서는 웜 풀 인스턴스의 상태 (정상 또는 비정상) 를 볼 수 있습니다. AWS CLI 또는 중 하나를 사용하여 상태를 볼 수도 있습니다. SDKs

웜 풀 인스턴스의 건전성 체크(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹(Auto Scaling groups) 페이지 하단에 분할 창이 열립니다.
3. 인스턴스 관리(Instance management) 탭의 웜 풀 인스턴스(Warm pool instances)에서 수명 주기(Lifecycle) 열은 인스턴스의 상태를 포함합니다.

   상태 열에는 Amazon EC2 Auto Scaling이 인스턴스 상태에 대해 수행한 평가가 표시됩니다.

   > **Note**
   > 새 인스턴스가 정상으로 시작됩니다. 수명 주기 후크가 완료될 때까지 인스턴스의 상태는 확인되지 않습니다.

건전성 체크 불합격 이유를 확인하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹(Auto Scaling groups) 페이지 하단에 분할 창이 열립니다.
3. 활동(Activity) 탭에서 활동 기록(Activity history)의 상태(Status) 열에 Auto Scaling 그룹이 성공적으로 인스턴스를 시작 또는 해지했는지가 표시됩니다.

   비건전 인스턴스를 해지한 경우, 원인(Cause) 열에는 해지 날짜 및 시간과 건전성 체크 불합격 이유가 표시됩니다. 예: “2021-04-01T21:48:35 Z에서 EBS 볼륨 상태 확인 실패에 대한 응답으로 인스턴스가 서비스 중단되었습니다.”

웜 풀 인스턴스의 건전성 체크(AWS CLI)

다음 describe-warm-pool명령을 사용하여 Auto Scaling 그룹의 웜 풀을 확인합니다.

```
aws autoscaling describe-warm-pool --auto-scaling-group-name my-asg
```

출력 예.

```
{
    "WarmPoolConfiguration": {
        "MinSize": 0,
        "PoolState": "Stopped"
    },
    "Instances": [
        {
            "InstanceId": "i-0b5e5e7521cfaa46c",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
            }
        },
        {
            "InstanceId": "i-0e21af9dcfb7aa6bf",
            "InstanceType": "t2.micro",
            "AvailabilityZone": "us-west-2a",
            "LifecycleState": "Warmed:Stopped",
            "HealthStatus": "Healthy",
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-08c4cd42f320d5dcd",
                "LaunchTemplateName": "my-template-for-auto-scaling",
                "Version": "1"
            }
        }
    ]
}
```

건전성 체크 불합격 이유 확인(AWS CLI)

다음 describe-scaling-activities 명령을 사용합니다.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

다음은 응답의 예입니다. `Description`은 Auto Scaling 그룹이 인스턴스를 해지했음을 나타내며 `Cause`는 건전성 체크 불합격의 원인을 나타냅니다.

크기 조정 활동은 시작 시간으로 정렬됩니다. 아직 진행 중인 활동이 먼저 설명됩니다.

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in
 response to EBS volume health check failure.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

를 사용하여 웜 풀을 만들고 관리하는 예 AWS CLI

AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 를 사용하여 웜 풀을 만들고 관리할 수 SDKs 있습니다.

다음 예에서는 AWS CLI를 사용하여 웜 풀을 생성하고 관리하는 방법을 보여줍니다.

내용

- 예 1: Stopped 상태로 인스턴스 유지
- 예 2: Running 상태로 인스턴스 유지
- 예 3: Hibernated 상태로 인스턴스 유지
- 예 4: 축소 시 웜 풀로 인스턴스 반환
- 예 5: 웜 풀의 최소 인스턴스 수 지정

- 예 6: 사용자 지정 사양을 사용하여 웜 풀 크기 정의
- 예 7: 절대 웜 풀 크기 정의
- 예 8: 웜 풀 삭제

예 1: **Stopped** 상태로 인스턴스 유지

다음 put-warm-pool예제에서는 인스턴스를 일정한 상태로 유지하는 웜 풀을 생성합니다. `Stopped`

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped
```

예 2: **Running** 상태로 인스턴스 유지

다음 put-warm-pool예제는 인스턴스를 `Running` 상태가 아닌 상태로 유지하는 웜 풀을 만듭니다. `Stopped`

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Running
```

예 3: **Hibernated** 상태로 인스턴스 유지

다음 put-warm-pool예제는 인스턴스를 `Hibernated` 상태가 아닌 상태로 유지하는 웜 풀을 만듭니다. `Stopped` 이렇게 하면 메모리 콘텐츠를 삭제하지 않고도 인스턴스를 중지할 수 있습니다 (RAM).

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Hibernated
```

예 4: 축소 시 웜 풀로 인스턴스 반환

다음 put-warm-pool예제는 인스턴스를 일정한 `Stopped` 상태로 유지하고 `--instance-reuse-policy` 옵션을 포함하는 웜 풀을 만듭니다. 인스턴스 재사용 정책 값은 EC2 Auto Scaling 그룹이 축소될 때 Amazon Auto Scaling에서 인스턴스를 웜 풀로 반환하도록 `'{"ReuseOnScaleIn": true}'` 지시합니다.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --instance-reuse-policy '{"ReuseOnScaleIn": true}'
```

예 5: 웜 풀의 최소 인스턴스 수 지정

다음 put-warm-pool예제는 최소 4개의 인스턴스를 유지 관리하는 웜 풀을 생성하여 트래픽 스파이크를 처리하는 데 사용할 수 있는 인스턴스가 4개 이상 있도록 합니다.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 4
```

예 6: 사용자 지정 사양을 사용하여 웜 풀 크기 정의

기본적으로 Amazon EC2 Auto Scaling은 웜 풀의 크기를 Auto Scaling 그룹의 최대 용량과 원하는 용량 간의 차이로 관리합니다. 하지만 `--max-group-prepared-capacity` 옵션을 사용하여 웜 풀의 크기를 그룹의 최대 용량과 독립적으로 관리할 수 있습니다.

다음 put-warm-pool예제에서는 웜 풀을 만들고 웜 풀과 Auto Scaling 그룹 모두에서 동시에 존재할 수 있는 최대 인스턴스 수를 설정합니다. 그룹의 원하는 용량이 800인 경우 이 명령을 실행한 후 초기화되는 웜 풀의 초기 크기는 100이 됩니다.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900
```

웜 풀의 최소 인스턴스 수를 유지하려면 다음과 같이 명령에 `--min-size` 옵션을 포함합니다.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --max-group-prepared-capacity 900 --min-size 25
```

예 7: 절대 웜 풀 크기 정의

`--max-group-prepared-capacity` 및 `--min-size` 옵션에 동일한 값을 설정하는 경우, 웜 풀은 절대 크기를 갖습니다. 다음 put-warm-pool예제에서는 10개의 인스턴스라는 웜 풀 크기를 일정하게 유지하는 웜 풀을 만듭니다.

```
aws autoscaling put-warm-pool --auto-scaling-group-name my-asg /
  --pool-state Stopped --min-size 10 --max-group-prepared-capacity 10
```

예 8: 웜 풀 삭제

다음 delete-warm-pool명령을 사용하여 웜 풀을 삭제합니다.

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg
```

웜 풀에 인스턴스가 있거나 조정 작업이 진행 중인 경우 delete-warm-pool명령을 `--force-delete` 옵션과 함께 사용하십시오. 또한 이 옵션은 Amazon EC2 인스턴스와 모든 미해결 수명 주기 작업을 종료합니다.

```
aws autoscaling delete-warm-pool --auto-scaling-group-name my-asg --force-delete
```

Auto Scaling 그룹에서 인스턴스 분리 또는 연결

Auto Scaling 그룹에서 인스턴스를 분리할 수 있습니다. 인스턴스가 분리되면 해당 인스턴스는 독립적이 되고 자체적으로 관리되거나 원래 속해 있던 그룹과 별도로 다른 Auto Scaling 그룹에 연결할 수 있습니다. 예를 들어 애플리케이션을 이미 실행 중인 기존 인스턴스를 사용하여 테스트를 수행하려는 경우 유용할 수 있습니다.

이 주제에서는 인스턴스를 분리하고 연결하는 방법에 대한 지침을 제공합니다. 인스턴스를 연결할 때 분리된 인스턴스가 아닌 기존 인스턴스를 사용할 수도 있습니다.

인스턴스를 분리했다가 동일한 그룹에 다시 연결하는 대신 대기 절차를 사용하여 그룹에서 인스턴스를 일시적으로 제거하는 것이 좋습니다. 자세한 내용은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 단원을 참조하십시오.

내용

- 인스턴스 분리 시 고려 사항
- 인스턴스 연결 시 고려 사항
- 분리 및 연결을 사용하여 인스턴스를 다른 그룹으로 이동

인스턴스 분리 시 고려 사항

인스턴스를 분리할 때는 다음 사항에 유의하세요.

- 인스턴스가 `InService` 상태에 있는 경우에만 인스턴스를 분리할 수 있습니다.
- 인스턴스를 분리한 후에도 계속 실행되고 요금이 발생합니다. 불필요한 요금을 피하려면 분리된 인스턴스가 더 이상 필요하지 않을 때 다시 연결하거나 종료해야 합니다.
- 분리하려는 인스턴스 수에 따라 원하는 용량을 줄이도록 선택할 수 있습니다. 용량을 줄이지 않기로 선택하면 Amazon EC2 Auto Scaling은 새 인스턴스를 시작하여 분리된 인스턴스를 대체하여 원하는 용량을 유지합니다.
- 분리하려는 인스턴스 수가 Auto Scaling 그룹을 최소 용량 미만으로 만드는 경우 최소 용량을 줄여야 합니다.

- 원하는 용량을 줄이지 않고 동일한 가용 영역에서 여러 인스턴스를 분리하면 `AZRebalance` 프로세스를 일시 중지하지 않는 한 그룹이 자체적으로 균형을 다시 맞춥니다. 자세한 내용은 Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개 단원을 참조하십시오.
- 연결된 로드 밸런서 대상 그룹 또는 Classic Load Balancer가 있는 Auto Scaling 그룹에서 인스턴스를 분리하면 해당 인스턴스가 로드 밸런서에서 등록 취소됩니다. 로드 밸런서에 연결 드레이닝(등록 취소 지연)이 활성화된 경우 Amazon EC2 Auto Scaling은 전송 중 요청이 완료될 때까지 기다립니다.

> **Note**
>
> `Standby` 상태에 있는 인스턴스를 분리하는 경우, 주의하세요. 인스턴스를 `Standby` 상태로 전환한 후 인스턴스를 분리하려고 하면 다른 인스턴스가 예기치 않게 해지될 수 있습니다.

인스턴스 연결 시 고려 사항

인스턴스를 연결할 때 다음 사항에 유의하세요.

- Amazon EC2 Auto Scaling은 연결된 인스턴스를 그룹 자체에서 시작된 인스턴스와 동일하게 처리합니다. 즉, 연결된 인스턴스를 선택한 경우 스케일 인 이벤트 중에 종료할 수 있습니다. 에서 부여한 권한 AWSServiceRoleForAutoScaling 서비스 연결 역할을 사용하면 Amazon EC2 Auto Scaling에서 이렇게 할 수 있습니다.
- 인스턴스를 연결하면 연결되는 인스턴스 수에 따라 그룹의 원하는 용량을 늘립니다. 새 인스턴스를 추가한 후 원하는 용량이 그룹의 최대 크기를 초과하는 경우 인스턴스를 더 연결하라는 요청이 실패합니다.
- 그룹에 인스턴스를 추가하여 가용 영역 간에 고르지 않은 배포가 발생하는 경우 Amazon EC2 Auto Scaling은 `AZRebalance` 프로세스를 일시 중지하지 않는 한 그룹을 리밸런싱하여 균등한 배포를 다시 설정합니다. 자세한 내용은 Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개 단원을 참조하십시오.
- 연결된 로드 밸런서 대상 그룹 또는 Classic Load Balancer가 있는 Auto Scaling 그룹에 인스턴스를 연결하면 해당 인스턴스가 로드 밸런서에 등록됩니다.

연결할 인스턴스는 다음 기준을 충족해야 합니다.

- 인스턴스가 Amazon 의 `running` 상태입니다EC2.
- 인스턴스를 시작하는 데 AMI 사용되는 가 여전히 있어야 합니다.

- 인스턴스가 다른 Auto Scaling 그룹의 구성원이 아닙니다.
- 인스턴스는 Auto Scaling 그룹에 정의된 가용 영역 중 하나로 시작됩니다.
- Auto Scaling 그룹에 로드 밸런서 대상 그룹 또는 Classic Load Balancer 가 연결된 경우 인스턴스와 로드 밸런서는 모두 동일한 에 있어야 합니다VPC.

분리 및 연결을 사용하여 인스턴스를 다른 그룹으로 이동

다음 절차 중 하나를 사용하여 Auto Scaling 그룹에서 인스턴스를 분리하고 다른 Auto Scaling 그룹에 연결합니다.

분리된 인스턴스에서 새 Auto Scaling 그룹을 생성하려면 기존 인스턴스의 파라미터를 사용하여 Auto Scaling 그룹 생성 (권장하지 않음, 시작 구성 생성)을 참조하세요.

Console

Auto Scaling 그룹에서 인스턴스를 분리하려면

1. 에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 https://console.aws.amazon.com/ec2/선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 인스턴스 관리(Instance management) 탭의 인스턴스(Instances)에서 인스턴스를 선택하고 작업(Actions), 분리(Detach)를 선택합니다.
4. 인스턴스 분리 대화 상자에서 인스턴스 교체 확인란을 선택한 상태로 유지하여 교체 인스턴스를 시작합니다. 원하는 용량을 줄이려면 확인란을 선택 취소합니다.
5. 확인하라는 메시지가 표시되면 **detach**를 입력하여 Auto Scaling 그룹에서 지정된 인스턴스의 삭제를 확인한 다음 인스턴스 분리를 선택합니다.

이제 인스턴스를 다른 Auto Scaling 그룹에 연결할 수 있습니다.

새 Auto Scaling 그룹에 인스턴스 연결

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. (옵션) 탐색 창의 Auto Scaling에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다. Auto Scaling 그룹을 선택하고 Auto Scaling 그룹의 최대 크기가 다른 인스턴스를 추가할 수 있을 만큼 큰지 확인합니다. 그렇지 않으면 세부 정보(Details) 탭에서 최대 용량을 늘립니다.

3. 탐색 창의 인스턴스(Instances)에서 인스턴스(Instances)를 선택한 다음 인스턴스를 선택합니다.
4. Actions(작업), Instance settings(인스턴스 설정), Attach to Auto Scaling Group(Auto Scaling 그룹에 연결)을 차례로 선택합니다.
5. Auto Scaling 그룹에 연결 페이지에서 Auto Scaling Group(Auto Scaling 그룹)에 대한 Auto Scaling 그룹을 선택하고 Attach(연결)를 선택합니다.
6. 인스턴스가 기준을 충족하지 않으면 세부 정보가 포함된 오류 메시지가 표시됩니다. 예컨대, Auto Scaling 그룹과 동일한 가용 영역에 있지 않을 수도 있습니다. 닫기를 선택하고 기준을 충족하는 Auto Scaling 그룹으로 다시 시도하세요.

AWS CLI

인스턴스를 분리하고 연결하려면 다음 예제 명령을 사용합니다. 각각 바꾸기 *user input placeholder* 자신의 정보를 사용합니다.

Auto Scaling 그룹에서 인스턴스를 분리하려면

1. 현재 인스턴스를 설명하려면 다음 describe-auto-scaling-instances 명령을 사용합니다.

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

다음 예제는 이 명령을 실행할 때 생성된 출력을 보여줍니다.

그룹에서 제거하려는 인스턴스의 ID를 기록해 둡니다. 다음 단계에서 이 ID가 필요합니다.

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
```

```
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0c20ac468fa3049e8",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0787762faf1c28619",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-0f280a4c58d319a8a",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
```

```
        }
    ]
}
```

2. 원하는 용량을 줄이지 않고 인스턴스를 분리하려면 다음 detach-instances 명령을 사용합니다.

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg
```

인스턴스를 분리하고 원하는 용량을 줄이려면 `--should-decrement-desired-capacity` 옵션을 포함합니다.

```
aws autoscaling detach-instances --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

이제 인스턴스를 다른 Auto Scaling 그룹에 연결할 수 있습니다.

새 Auto Scaling 그룹에 인스턴스 연결

1. 다른 Auto Scaling 그룹에 인스턴스를 연결하려면 다음 attach-instances 명령을 사용합니다.

```
aws autoscaling attach-instances --instance-ids i-05b4f7d5be44822a6 --auto-
scaling-group-name my-asg-for-testing
```

2. 인스턴스를 연결한 후 Auto Scaling 그룹의 크기를 확인하려면 다음 describe-auto-scaling-groups 명령을 사용합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg-
for-testing
```

다음 예제 응답은 그룹에 실행 중인 인스턴스가 두 개 있으며, 그 중 하나가 연결한 인스턴스임을 보여줍니다.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg-for-testing",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
```

```
                "Version": "2",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 2,
            ...
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "2",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-00dcdfffdf5175890",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            ...
        }
    ]
}
```

Auto Scaling 그룹에서 일시적으로 인스턴스 제거

`InService` 상태의 인스턴스를 `Standby` 상태로 설정하고, 인스턴스를 업데이트하거나 문제 해결한 다음, 해당 인스턴스를 서비스 상태로 되돌릴 수 있습니다. 대기 상태의 인스턴스는 Auto Scaling 그룹에 여전히 속하나, 로드 밸런서 트래픽을 처리하지는 못합니다.

이 기능을 사용하면 Amazon EC2 Auto Scaling이 상태 확인의 일부로 또는 스케일 인 이벤트 중에 인스턴스를 종료하는 것에 대해 걱정하지 않고 인스턴스를 중지 및 시작하거나 재부팅할 수 있습니다.

예를 들어 시작 템플릿 또는 시작 구성을 변경하여 Auto Scaling 그룹의 Amazon Machine Image(AMI)를 언제든지 변경할 수 있습니다. Auto Scaling 그룹이 시작하는 모든 후속 인스턴스는 이 를 사용합니다AMI. 그러나 Auto Scaling 그룹은 현재 서비스 중인 인스턴스를 업데이트하지 않습니다. 이러한 인스턴스를 종료하고 Amazon EC2 Auto Scaling이 해당 인스턴스를 교체하도록 하거나 인스턴스 새로 고침 기능을 사용하여 인스턴스를 종료하고 교체할 수 있습니다. 또는 인스턴스 상태를 대기로 설정하고 소프트웨어를 업데이트한 후 인스턴스를 다시 서비스 상태로 설정할 수 있습니다.

Auto Scaling 그룹에서 인스턴스를 분리하는 것은 인스턴스를 대기 상태로 전환하는 것과 유사합니다. 인스턴스를 다른 그룹에 연결하거나 독립 실행형 인스턴스와 같은 인스턴스를 관리하고 종료하려는 경우 EC2 인스턴스를 분리하는 것이 유용할 수 있습니다. 자세한 내용은 Auto Scaling 그룹에서 인스턴스 분리 또는 연결 단원을 참조하십시오.

내용

- 대기 상태를 작동하는 방법
- 고려 사항
- 대기 상태의 인스턴스 상태
- 인스턴스를 대기 상태로 설정하여 임시로 제거

대기 상태를 작동하는 방법

대기 상태는 Auto Scaling 그룹에서 인스턴스를 일시적으로 제거할 수 있도록 다음과 같이 작동합니다.

1. 인스턴스를 대기 상태에 놓습니다. 인스턴스는 대기 상태를 끝낼 때까지 이 상태로 유지됩니다.
2. Auto Scaling 그룹에 연결된 로드 밸런서 대상 그룹 또는 Classic Load Balancer가 있는 경우, 로드 밸런서에서 해당 인스턴스가 등록 취소됩니다. 로드 밸런서에 대해 Connection Draining이 활성화된 경우, Elastic Load Balancing은 등록 취소 프로세스를 완료하기 전에 300초 동안 대기하는데, 이는 진행 중인 요청을 완료하는 데 도움이 될 수 있습니다.

3. 인스턴스를 업데이트하거나 문제를 해결할 수 있습니다.
4. 대기 상태를 끝내 인스턴스를 서비스 상태로 되돌립니다.
5. Auto Scaling 그룹에 연결된 로드 밸런서 대상 그룹 또는 Classic Load Balancer가 있는 경우, 로드 밸런서에 해당 인스턴스가 등록됩니다.

Auto Scaling 그룹에서 인스턴스의 수명 주기에 대한 자세한 설명은 아마존 EC2 Auto Scaling 인스턴스 라이프사이클 섹션을 참조하세요.

고려 사항

인스턴스를 대기 상태로 전환하거나 대기 상태에서 벗어날 때 고려할 사항은 다음과 같습니다.

- 인스턴스를 대기 상태로 설정할 때 이 작업을 통해 원하는 용량을 줄이거나 동일한 값을 유지할 수 있습니다.
  - Auto Scaling 그룹의 원하는 용량을 줄이지 않도록 선택하면 Amazon EC2 Auto Scaling은 인스턴스를 시작하여 대기 중인 인스턴스를 대체합니다. 하나 이상의 인스턴스가 대기 상태에 있는 동안 애플리케이션의 용량을 유지하기 위해서입니다.
  - Auto Scaling 그룹의 원하는 용량을 점감하도록 선택하면 대기 중인 인스턴스를 교체할 인스턴스가 시작되지 않습니다.
- 인스턴스를 다시 서비스하면 Auto Scaling 그룹에 있는 인스턴스 수를 반영하여 원하는 용량이 증가합니다.
- 증가(및 감소)를 수행하려면 새로 원하는 용량이 최소 그룹 크기와 최대 그룹 크기 사이여야 합니다. 그렇지 않은 경우, 작업이 실패합니다.
- 인스턴스를 대기 상태로 전환하거나 대기 상태를 종료하여 인스턴스를 서비스로 반환한 후 언제든지 Auto Scaling 그룹이 가용 영역 간에 균형을 이루지 못하는 것으로 밝혀지면 Amazon EC2 Auto Scaling은 `AZRebalance` 프로세스를 일시 중지하지 않는 한 가용 영역의 균형을 재조정하여 보상합니다. 자세한 내용은 Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개 단원을 참조하십시오.
- 대기 상태의 인스턴스에 대해 요금이 청구됩니다.

대기 상태의 인스턴스 상태

Amazon EC2 Auto Scaling은 대기 상태인 인스턴스에 대해 상태 확인을 수행하지 않습니다. 인스턴스가 대기 상태에 있는 동안 이 인스턴스의 상태는 대기 상태로 설정되기 전의 상태를 반영합니다. Amazon EC2 Auto Scaling은 인스턴스를 다시 사용할 때까지 인스턴스에 대한 상태 확인을 수행하지 않습니다.

예를 들어 정상 인스턴스를 대기 상태로 설정한 다음 종료하면 Amazon EC2 Auto Scaling은 인스턴스를 정상으로 계속 보고합니다. 대기 상태였던 종료된 인스턴스를 다시 서비스 상태로 전환하려고 하면 Amazon EC2 Auto Scaling은 인스턴스에 대한 상태 확인을 수행하고, 종료 중이고 비정상인지 확인한 다음 대체 인스턴스를 시작합니다. 자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 단원을 참조하십시오.

인스턴스를 대기 상태로 설정하여 임시로 제거

다음 절차 중 하나를 사용하여 인스턴스를 대기 상태로 전환하여 일시적으로 서비스에서 제외합니다.

Console

인스턴스를 일시적으로 제거하려면

1. 에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 https://console.aws.amazon.com/ec2/선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 인스턴스 관리(Instance management) 탭의 인스턴스(Instances)에서 인스턴스를 선택합니다.
4. 작업(Actions), 대기로 설정(Set to Standby)을 선택합니다.
5. 교체 인스턴스를 시작시키려면 대기로 설정 대화상자에서 인스턴스 교체 체크박스를 선택된 대로 유지하십시오. 원하는 용량을 줄이려면 확인란을 선택 취소합니다.
6. 확인 메시지가 표시되면 **standby**를 입력하여 지정된 인스턴스를 `Standby` 상태로 설정하는 것을 확인한 다음 대기 상태로 설정을 선택합니다.
7. 필요에 따라 인스턴스를 업데이트하거나 문제 해결할 수 있습니다. 모두 마쳤으면 다음 단계를 계속하여 인스턴스를 서비스 상태로 되돌립니다.
8. 인스턴스를 선택하고 작업 , 로 설정을 InService 선택합니다. 다음으로 설정 InService 대화 상자에서 로 설정을 InService선택합니다.

AWS CLI

Auto Scaling 그룹에서 인스턴스를 일시적으로 제거하려면 다음 예제 명령을 사용합니다. 각각 바꾸기 *`user input placeholder`* 자신의 정보를 사용합니다.

인스턴스를 일시적으로 제거하려면

1. 다음 describe-auto-scaling-instances 명령을 사용하여 업데이트할 인스턴스를 식별합니다.

```
aws autoscaling describe-auto-scaling-instances \
  --query 'AutoScalingInstances[?AutoScalingGroupName==`my-asg`]'
```

다음 예제는 이 명령을 실행할 때 생성된 출력을 보여줍니다.

그룹에서 제거하려는 인스턴스의 ID를 기록해 둡니다. 다음 단계에서 이 ID가 필요합니다.

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceId": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
       ...
    ]
}
```

2. enter-standby 명령을 사용하여 인스턴스를 Standby 상태로 이동합니다. `--should-decrement-desired-capacity` 옵션을 사용하면 원하는 용량을 줄이므로 Auto Scaling 그룹에서는 교체 인스턴스를 시작하지 않습니다.

```
aws autoscaling enter-standby --instance-ids i-05b4f7d5be44822a6 \
  --auto-scaling-group-name my-asg --should-decrement-desired-capacity
```

다음은 응답의 예입니다.

```
{
    "Activities": [
        {
            "ActivityId": "3b1839fe-24b0-40d9-80ae-bcd883c2be32",
```

```
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby:
 i-05b4f7d5be44822a6",
            "Cause": "At 2023-12-15T21:31:26Z instance i-05b4f7d5be44822a6 was
 moved to standby
              in response to a user request, shrinking the capacity from 4 to
 3.",
            "StartTime": "2023-12-15T21:31:26.150Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\":\"us-west-2a\"}"
        }
    ]
}
```

3. (선택 사항) 다음 describe-auto-scaling-instances 명령을 Standby 사용하여 인스턴스가 에 있는지 확인합니다.

```
aws autoscaling describe-auto-scaling-instances --instance-
ids i-05b4f7d5be44822a6
```

다음은 응답의 예입니다. 인스턴스가 현재 Standby 상태임을 확인할 수 있습니다.

```
{
    "AutoScalingInstances": [
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "Standby"
        },
        ...
    ]
```

```
}
```

4. 필요에 따라 인스턴스를 업데이트하거나 문제 해결할 수 있습니다. 모두 마쳤으면 다음 단계를 계속하여 인스턴스를 서비스 상태로 되돌립니다.
5. 다음 exit-standby 명령을 사용하여 인스턴스를 다시 서비스 상태로 설정합니다.

```
aws autoscaling exit-standby --instance-ids i-05b4f7d5be44822a6 --auto-scaling-
group-name my-asg
```

다음은 응답의 예입니다.

```
{
    "Activities": [
        {
            "ActivityId": "db12b166-cdcc-4c54-8aac-08c5935f8389",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-05b4f7d5be44822a6",
            "Cause": "At 2023-12-15T21:46:14Z instance i-05b4f7d5be44822a6 was
 moved out of standby in
               response to a user request, increasing the capacity from 3 to
 4.",
            "StartTime": "2023-12-15T21:46:14.678Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-c934b782\",\"Availability Zone
\":\"us-west-2a\"}"
        }
    ]
}
```

6. (옵션) 다음 `describe-auto-scaling-instances` 명령을 사용하여 인스턴스가 다시 서비스 상태가 되었는지 확인합니다.

```
aws autoscaling describe-auto-scaling-instances --instance-
ids i-05b4f7d5be44822a6
```

다음은 응답의 예입니다. 인스턴스의 상태가 `InService`입니다.

```
{
    "AutoScalingInstances": [
```

```
        {
            "ProtectedFromScaleIn": false,
            "AvailabilityZone": "us-west-2a",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "InstanceId": "i-05b4f7d5be44822a6",
            "InstanceType": "t3.micro",
            "AutoScalingGroupName": "my-asg",
            "HealthStatus": "HEALTHY",
            "LifecycleState": "InService"
        },
       ...
    ]
}
```

Auto Scaling 인프라 삭제

조정 인프라를 완전히 삭제하려면 다음 작업을 완료하세요.

Tasks

- Auto Scaling 그룹 삭제
- (옵션) 시작 구성 삭제
- (옵션) 시작 템플릿 삭제
- (옵션) 로드 밸런서 및 대상 그룹 삭제
- (선택 사항) 경보 삭제 CloudWatch

Auto Scaling 그룹 삭제

Auto Scaling 그룹을 삭제하면 원하는 값과 최소, 최대값이 0으로 설정됩니다. 그리고 인스턴스가 해지됩니다. 인스턴스를 삭제하면 연결된 로그 또는 데이터, 그리고 인스턴스의 모든 볼륨도 모두 삭제됩니다. 하나 이상의 인스턴스를 해지하지 않으려면 Auto Scaling 그룹을 삭제하기 전에 인스턴스를 분리할 수 있습니다. 그룹에 조정 정책이 있는 경우, 그룹을 삭제하면 정책, 기본 경보 작업 및 더 이상 연결된 작업이 없는 경보가 삭제됩니다.

Auto Scaling 그룹을 삭제하려면(콘솔)

1. 에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 https://console.aws.amazon.com/ec2/선택합니다.
2. Auto Scaling 그룹 옆에 있는 확인란을 선택하고 조치, 삭제를 선택합니다.
3. 확인 메시지가 표시되면 **delete**를 입력하여 지정된 Auto Scaling 그룹 삭제를 확인한 다음 Delete(삭제)를 선택합니다.

   Name(이름) 열의 로딩 아이콘은 Auto Scaling 그룹이 삭제 중임을 나타납니다. 원하는 수(Desired), 최소(Min) 및 최대(Max) 열에 Auto Scaling 그룹에 대한 인스턴스가 `0`으로 표시됩니다. 인스턴스를 해지하고 그룹을 삭제하는 데 몇 분 정도 걸립니다. 목록을 새로 고침하여 상태를 확인합니다.

Auto Scaling 그룹을 삭제하려면(AWS CLI)

다음 delete-auto-scaling-group 명령을 사용하여 Auto Scaling 그룹을 삭제합니다. 그룹에 EC2 인스턴스가 있는 경우 이 작업은 작동하지 않습니다. 인스턴스가 없는 그룹 전용입니다.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg
```

그룹에 인스턴스 또는 조정 작업이 진행 중인 경우 `--force-delete` 옵션과 함께 delete-auto-scaling-group 명령을 사용합니다. 이렇게 하면 EC2 인스턴스도 종료됩니다. Amazon Auto Scaling 콘솔에서 EC2 Auto Scaling 그룹을 삭제하면 콘솔은 이 작업을 사용하여 EC2 인스턴스를 종료하고 그룹을 동시에 삭제합니다.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name my-asg --force-
delete
```

(옵션) 시작 구성 삭제

나중에 시작 구성을 사용하려면 이 단계를 건너뛸 수 있습니다.

시작 구성을 삭제하는 방법(콘솔)

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 왼쪽 탐색 창의 Auto Scaling에서 Auto Scaling 그룹을 선택합니다.
3. 페이지 상단에서 시작 구성을 선택합니다. 확인 메시지가 표시되면 시작 구성 보기를 선택하여 시작 구성 페이지를 볼 것인지 확인합니다.

4. 시작 구성을 선택한 다음 조치, 시작 구성 복사를 선택합니다.
5. 확인 메시지가 나타나면 삭제를 선택합니다.

시작 구성을 삭제하는 방법(AWS CLI)

다음 delete-launch-configuration 명령을 사용합니다.

```
aws autoscaling delete-launch-configuration --launch-configuration-name my-launch-
config
```

(옵션) 시작 템플릿 삭제

시작 템플릿 또는 시작 템플릿의 한 가지 버전만 삭제할 수 있습니다. 시작 템플릿을 삭제하면 모든 해당 버전이 삭제됩니다.

나중에 시작 템플릿을 사용하려면 이 단계를 건너뛸 수 있습니다.

시작 템플릿을 삭제하려면(콘솔)

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창의 Instances(인스턴스)에서 Launch Templates(시작 템플릿)을 선택합니다.
3. 시작 템플릿을 선택한 다음, 다음 중 하나를 수행하세요.
   - Actions(작업)와 Delete template(템플릿 삭제)을 차례로 선택합니다. 확인 메시지가 표시되면 **Delete**를 입력하여 지정된 시작 템플릿 삭제를 확인한 다음 Delete(삭제)를 선택합니다.
   - [Actions]와 [Delete template version]을 차례로 선택합니다. 삭제할 버전을 선택하고 삭제를 선택합니다.

시작 템플릿 삭제(AWS CLI)

다음 delete-launch-template 명령을 사용하여 템플릿과 모든 버전을 삭제합니다.

```
aws ec2 delete-launch-template --launch-template-id lt-068f72b72934aff71
```

또는 delete-launch-template-versions 명령을 사용하여 시작 템플릿의 특정 버전을 삭제할 수 있습니다.

```
aws ec2 delete-launch-template-versions --launch-template-id lt-068f72b72934aff71 --
versions 1
```

(옵션) 로드 밸런서 및 대상 그룹 삭제

Auto Scaling 그룹이 Elastic Load Balancing 로드 밸런서에 연결되지 않았거나 로드 밸런서를 나중에 사용하려면 이 단계를 건너뛰세요.

로드 밸런서를 삭제하려면(콘솔)

1. 에서 Amazon EC2 콘솔을 엽니다https://console.aws.amazon.com/ec2/.
2. 탐색 창의 로드 밸런싱(Load Balancing) 아래에서 로드 밸런서(Load Balancers)를 선택합니다.
3. 로드 밸런서를 선택한 다음 작업, 삭제를 선택합니다.
4. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.

대상 그룹을 삭제하려면(콘솔)

1. 탐색 창의 Load Balancing 아래에서 대상 그룹을 선택합니다.
2. 대상 그룹을 선택하고 [Actions], [Delete]를 차례로 선택합니다.
3. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.

Auto Scaling 그룹과 연결된 로드 밸런서를 삭제하려면(AWS CLI)

Application Load Balancer 및 Network Load Balancer의 경우 다음 delete-load-balancer 및 delete-target-group 명령을 사용합니다.

```
aws elbv2 delete-load-balancer --load-balancer-arn my-load-balancer-arn
aws elbv2 delete-target-group --target-group-arn my-target-group-arn
```

Classic Load Balancer의 경우 다음 delete-load-balancer 명령을 사용합니다.

```
aws elb delete-load-balancer --load-balancer-name my-load-balancer
```

(선택 사항) 경보 삭제 CloudWatch

Auto Scaling 그룹과 연결된 CloudWatch 경보를 삭제하려면 다음 단계를 완료합니다. 예컨대, 단계별 스케일링 또는 단순 스케일링 정책과 관련된 경보가 있을 수 있습니다.

> **Note**
>
> Auto Scaling 그룹을 삭제하면 Amazon EC2 Auto Scaling이 대상 추적 조정 정책에 대해 관리하는 CloudWatch 경보가 자동으로 삭제됩니다.

Auto Scaling 그룹이 CloudWatch 경보와 연결되어 있지 않거나 나중에 사용할 수 있도록 경보를 유지하려는 경우 이 단계를 건너뛸 수 있습니다.

CloudWatch 경보를 삭제하려면(콘솔)

1. 에서 CloudWatch 콘솔을 엽니다https://console.aws.amazon.com/cloudwatch/.
2. 탐색 창에서 Alarms(경보)를 선택합니다.
3. 경보를 선택하고 작업, 삭제를 선택합니다.
4. 확인 메시지가 나타나면 삭제를 선택합니다.

CloudWatch 경보를 삭제하려면(AWS CLI)

delete-alarms 명령을 사용합니다. 한 번에 하나 이상의 경보를 삭제할 수 있습니다. 예컨대, 다음 명령을 사용하여 `Step-Scaling-AlarmHigh-AddCapacity` 및 `Step-Scaling-AlarmLow-RemoveCapacity` 경보를 삭제합니다.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-Scaling-AlarmLow-RemoveCapacity
```

Auto Scaling 그룹의 인스턴스를 교체합니다.

Amazon EC2 Auto Scaling은 업데이트를 수행한 후 Auto Scaling 그룹의 Amazon EC2 인스턴스를 교체할 수 있는 기능 (예: 새 Amazon 머신 이미지 (AMI) 으로 새 시작 템플릿을 추가하거나 새 인스턴스 유형을 추가할 수 있는 기능을 제공합니다. 또한 인스턴스를 대체하는 것과 동일한 작업에 포함시킬 수 있는 옵션을 제공하여 업데이트를 간소화하는 데도 도움이 됩니다.

이 섹션에서는 다음을 수행하는 데 도움이 되는 정보를 제공합니다.

- 인스턴스 새로 고침을 시작하여 Auto Scaling 그룹의 인스턴스를 교체합니다.
- 원하는 구성을 설명하는 특정 업데이트를 선언하고 Auto Scaling 그룹을 원하는 구성으로 업데이트합니다.
- 이미 업데이트된 인스턴스 교체를 건너뜁니다.
- 체크포인트를 사용하면 단계별로 인스턴스를 업데이트하고 특정 시점에서 인스턴스에 대한 검증을 수행할 수 있습니다.
- 체크포인트에 도달하면 이메일로 알림을 수신합니다.
- 롤백을 사용하여 Auto Scaling 그룹을 이전에 사용한 구성으로 복원합니다.
- 어떤 이유로든 인스턴스 새로 고침에 실패하거나 지정한 Amazon CloudWatch 경보가 이 `ALARM` 상태로 전환되면 자동으로 롤백합니다.
- 인스턴스의 수명을 제한하여 Auto Scaling 그룹 전체에서 일관된 소프트웨어 버전과 인스턴스 구성을 제공합니다.

내용

- 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다.
- 최대 인스턴스 수명을 기준으로 Auto Scaling 인스턴스 교체

인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다.

인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트할 수 있습니다. 이 기능은 구성 변경으로 인해 인스턴스를 교체해야 하는 경우, 특히 Auto Scaling 그룹에 많은 수의 인스턴스가 포함된 경우 유용할 수 있습니다.

인스턴스 새로 고침이 도움이 될 수 있는 몇 가지 상황은 다음과 같습니다.

- Auto Scaling 그룹 전체에 새 Amazon 머신 이미지 (AMI) 또는 사용자 데이터 스크립트 배포. 변경 내용이 포함된 새 시작 템플릿을 생성한 다음 인스턴스 새로 고침을 사용하여 업데이트를 즉시 배포할 수 있습니다.
- 인스턴스를 새 인스턴스 유형으로 마이그레이션하여 최신 개선 사항 및 최적화를 활용하십시오.
- Auto Scaling 그룹을 시작 구성 사용에서 시작 템플릿 사용으로 전환합니다. 시작 구성을 시작 템플릿에 복사한 다음 인스턴스 새로 고침을 사용하여 인스턴스를 새 템플릿으로 업데이트할 수 있습니다. 시작 템플릿으로의 마이그레이션에 대한 자세한 설명은 Auto Scaling 그룹을 마이그레이션하여 템플릿 시작를 참조하세요.

내용

- Auto Scaling 그룹에서 인스턴스 새로 고침이 작동하는 방식
- 인스턴스 새로 고침의 기본값 이해
- 또는 를 사용하여 인스턴스 새로 고침을 AWS Management Console 시작합니다. AWS CLI
- AWS Management Console 또는 를 사용하여 인스턴스 새로 고침을 모니터링합니다. AWS CLI
- 또는 를 사용하여 인스턴스 새로 고침을 AWS Management Console 취소하십시오. AWS CLI
- 수동 또는 자동 롤백으로 변경 취소
- 매칭 건너뛰기와 함께 인스턴스 새로 고침 사용
- 인스턴스 새로 고침에 체크포인트 추가

Auto Scaling 그룹에서 인스턴스 새로 고침이 작동하는 방식

이 주제에서는 인스턴스 새로 고침의 작동 방식을 설명하고 이를 효과적으로 사용하기 위해 이해해야 하는 주요 개념을 소개합니다.

내용

- 작동 방식
- 핵심 개념
- 건전성 체크 유예 기간
- 인스턴스 유형 호환성
- 제한 사항

작동 방식

Auto Scaling 그룹의 인스턴스를 새로 고치려면 최신 버전의 애플리케이션과 원하는 기타 업데이트가 포함된 새 구성을 정의하면 됩니다. 그런 다음 인스턴스 새로 고침을 시작하여 해당 구성을 기반으로 기존 인스턴스를 새 인스턴스로 교체하십시오.

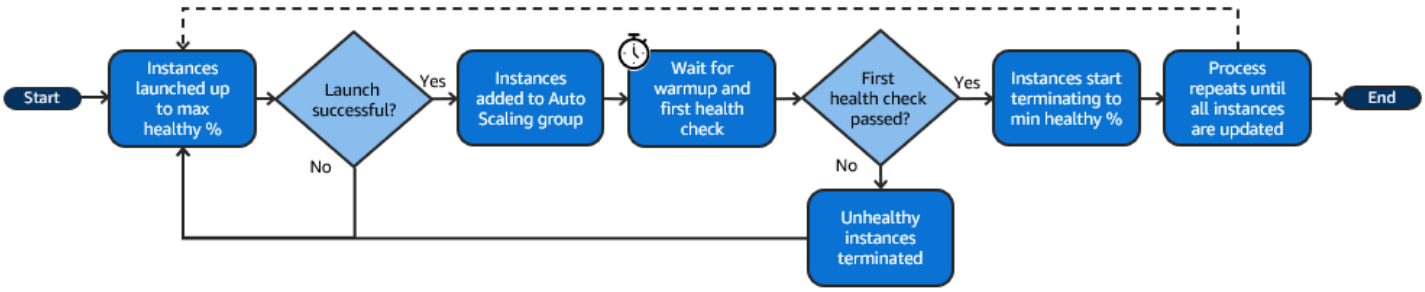
인스턴스 새로 고침을 수행하려면:

1. 새 시작 템플릿을 생성하거나 기존 템플릿을 원하는 구성 변경 (예: 새 Amazon Machine Image (AMI)) 으로 업데이트하십시오. 자세한 정보는 Auto Scaling 그룹에 대한 시작 템플릿 생성을 참조하세요.
2. Amazon EC2 Auto Scaling 콘솔 또는 SDK를 사용하여 인스턴스 새로 고침을 시작합니다. AWS CLI
   - 생성한 새 시작 템플릿 또는 시작 템플릿 버전을 지정합니다. 이는 새 인스턴스를 시작하는 데 사용됩니다.
   - 선호하는 최소 및 최대 정상 비율을 설정합니다. 이는 동시에 교체되는 인스턴스 수와 이전 인스턴스를 종료하기 전에 새 인스턴스를 시작할지 여부를 제어합니다.
   - 다음과 같은 모든 선택적 설정을 구성하십시오.
     - 체크포인트 — 일정 비율의 교체가 완료되면 인스턴스 새로 고침을 일시 중지하여 진행 상황을 확인합니다.
     - 매칭 건너뛰기 — 이전 인스턴스를 새 구성과 비교하고 일치하지 않는 인스턴스만 교체합니다. 콘솔에서 인스턴스 새로 고침을 시작하면 기본적으로 스킵 매칭이 켜져 있습니다.
     - 여러 인스턴스 유형 - 새 혼합 인스턴스 정책 또는 업데이트된 혼합 인스턴스 정책을 원하는 구성의 일부로 적용합니다.

인스턴스 새로 고침이 시작되면 Amazon EC2 Auto Scaling은 다음을 수행합니다.

- 최소 및 최대 정상 비율을 기준으로 인스턴스를 일괄적으로 교체합니다.
- 최소 정상 백분율이 100% 로 설정된 경우 이전 인스턴스를 종료하기 전에 새 인스턴스를 먼저 시작하십시오. 이렇게 하면 원하는 용량을 항상 유지할 수 있습니다.
- 인스턴스의 상태를 확인하고 추가 인스턴스가 교체되기 전에 인스턴스가 워밍업할 시간을 주십시오.
- 비정상으로 확인된 인스턴스를 종료하고 교체하십시오.
- 인스턴스 새로 고침이 성공하면 Auto Scaling 그룹 설정을 새 구성 변경 사항으로 자동 업데이트합니다.
- 웜 풀에 있는 인스턴스보다 먼저 인스턴스를 `InService` 교체하십시오.

다음 흐름도는 최소 정상 비율을 100% 로 설정한 경우의 종료 전 시작 동작을 보여줍니다.



> **Note**
>
> 인스턴스 유지 관리 정책을 설정하지 않았거나 기존 정책을 재정의해야 하는 경우에만 인스턴스 새로 고침의 최소 및 최대 정상 비율을 지정하면 됩니다. 자세한 정보는 인스턴스 유지 관리 정책을 참조하세요.
>
> 마찬가지로, 기본 웜업을 활성화하지 않았거나 기본값을 재정의해야 하는 경우에만 인스턴스 새로 고침의 인스턴스 워밍업 기간을 지정하면 됩니다. 자세한 정보는 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정을 참조하세요.

핵심 개념

시작하기 전에 인스턴스의 다음과 같은 핵심 개념을 익히십시오:

최소 건전 백분율

최소 정상 백분율은 인스턴스를 새로 고치는 동안 서비스를 유지하고 정상 상태로 유지하며 새로 고침을 계속할 수 있도록 사용할 준비가 된 상태로 유지하려는 용량의 백분율입니다. 예컨대, 최소 건전 백분율이 90%이고 최대 건전 백분율이 100%인 경우, 용량의 10%는 한 번에 교체될 것입니다. 새 인스턴스가 건전성 체크를 통과하지 못하면 Amazon EC2 Auto Scaling이 인스턴스를 해지하고 교체합니다. 인스턴스 새로 고침이 건전 인스턴스를 시작시킬 수 없으면 인스턴스 새로 고침이 결국 실패하고 그룹의 다른 90%는 그대로 유지됩니다. 새 인스턴스가 정상적으로 유지되고 준비 기간이 끝나면 Amazon EC2 Auto Scaling은 계속해서 다른 인스턴스를 대체할 수 있습니다.

인스턴스 새로 고침은 인스턴스를 한 번에 하나씩, 한 번에 여러 개 또는 한 번에 모두 교체할 수 있습니다. 한 번에 하나의 인스턴스를 교체하려면 최소 및 최대 건전 백분율을 100%로 설정하십시오. 이렇게 하면 인스턴스 새로 고침이 해지되기 전에 실행되도록 동작이 변경되어 그룹의 용량이 원하는 용량의 100% 미만으로 떨어지는 것을 방지할 수 있습니다. 모든 인스턴스를 한 번에 교체하려면 최소 건전 백분율을 0%로 설정합니다.

최대 건전 백분율

최대 건전 백분율은 인스턴스를 교체할 때 Auto Scaling 그룹이 늘릴 수 있는 원하는 용량의 백분율입니다. 최소값과 최대값의 차이가 100을 초과할 수 없습니다. 범위가 증가할 수록 동시에 바꿀 수 있는 인스턴스의 수가 늘어납니다.

인스턴스 워밍업

인스턴스 워밍업은 초기화가 완료된 것으로 간주될 때 새 인스턴스의 상태가 `InService`로 변경된 시점부터의 시간 기간입니다. 인스턴스 새로 고침 중 인스턴스가 건전성 체크를 통과하면 Amazon EC2 Auto Scaling은 새로 시작된 인스턴스가 정상임을 확인한 후 즉시 다음 인스턴스 교체로 이동하지 않습니다. 다음 인스턴스 교체로 넘어가기 전에 워밍업 기간을 기다립니다. 이는 애플리케이션이 요청에 응답하기 전에 초기화 시간이 필요한 경우에 유용할 수 있습니다.

인스턴스 워밍업은 기본 인스턴스 워밍업과 동일한 방식으로 작동합니다. 따라서 동일한 스케일링 고려 사항이 적용됩니다. 자세한 설명은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 섹션을 참조하세요.

원하는 구성

원하는 구성은 Amazon EC2 Auto Scaling이 Auto Scaling 그룹 전체에 배치하도록 하려는 새로운 구성입니다. 예컨대, 인스턴스에 대한 새 시작 템플릿과 새 인스턴스 유형을 지정할 수 있습니다. 인스턴스를 새로 고치는 동안 Amazon EC2 Auto Scaling은 Auto Scaling 그룹을 원하는 구성으로 업데이트합니다. 인스턴스를 새로 고치는 중에 스케일 아웃 이벤트가 발생하면 Amazon EC2 Auto Scaling은 그룹의 현재 설정 대신 원하는 구성으로 새 인스턴스를 시작합니다. 인스턴스 새로 고침에 성공하면 Amazon EC2 Auto Scaling이 Auto Scaling 그룹의 설정을 업데이트하여 인스턴스 새로 고침의 일부로 지정한 원하는 새 구성을 반영합니다.

매칭 건너뛰기

매칭 건너뛰기는 Amazon EC2 Auto Scaling에서 이미 최신 업데이트가 있는 인스턴스를 무시하도록 합니다. 이렇게 하면 필요 이상으로 많은 인스턴스를 교체하지 않습니다. 이는 Auto Scaling 그룹이 특정 버전의 시작 템플릿을 사용하고 다른 버전을 사용하는 인스턴스만 교체하게 하려는 경우, 유용합니다.

체크포인트

체크포인트는 지정된 시간 동안 인스턴스 새로 고침이 일시 중지되는 시점입니다. 인스턴스 새로 고침에는 체크포인트가 여러 개 포함될 수 있습니다. Amazon EC2 Auto Scaling은 각 체크포인트에 대한 이벤트를 생성합니다. 따라서 Amazon SNS와 같은 대상으로 이벤트를 전송하여 체크포인트에 도달했을 때 알림을 받는 EventBridge 규칙을 추가할 수 있습니다. 체크포인트에 도달하면 배치를 확인할 수 있습니다. 문제가 발견되면 인스턴스 새로 고침을 취소하거나 롤백할 수 있습니다.

단계별로 업데이트를 배치하는 기능은 체크포인트의 주요 이점입니다. 체크포인트를 사용하지 않으면 롤링 교체가 계속해서 수행됩니다.

인스턴스 새로 고침을 시작할 때 구성할 수 있는 모든 기본 설정에 대해 자세히 알아보려면 인스턴스 새로 고침의 기본값 이해 섹션을 참조하세요.

건전성 체크 유예 기간

Amazon EC2 Auto Scaling은 Auto Scaling 그룹이 사용하는 건전성 체크의 상태에 따라 인스턴스가 정상인지 여부를 결정합니다. 자세한 설명은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 섹션을 참조하세요.

건전성 체크가 최대한 빨리 시작되게 하려면 그룹의 건전성 체크 유예 기간을 너무 길게 설정하지 마세요. 하지만 Elastic Load Balancing 건전성 체크에서 대상이 요청을 처리할 수 있는지 확인할 수 있을 만큼 길게 설정하세요. 자세한 설명은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 섹션을 참조하세요.

인스턴스 유형 호환성

인스턴스 유형을 변경하기 전에 시작 템플릿에서 유효한지 확인하는 것이 좋습니다. 이렇게 하면 지정한 AMI와의 호환성이 확인됩니다. 예컨대, 반가상화(PV) AMI에서 원본 인스턴스를 시작했지만 하드웨어 가상 머신(HVM) AMI에서만 지원되는 현재 세대 인스턴스 유형으로 변경하려고 합니다. 이 경우, 시작 템플릿에 HVM AMI를 사용해야 합니다.

인스턴스를 시작하지 않고 인스턴스 유형의 호환성을 확인하려면 다음 예와 같이 run-instance 명령을 `--dry-run` 옵션과 함께 사용합니다.

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --
dry-run
```

호환성 결정 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 유형 변경을 위한 호환성을 참조하십시오.

제한 사항

- 총 기간: 인스턴스 새로 고침이 인스턴스를 계속해서 적극적으로 교체할 수 있는 최대 시간은 14일입니다.
- 가중 그룹별 동작 차이: 혼합 인스턴스 그룹이 원하는 그룹 용량보다 크거나 같은 인스턴스 가중치로 구성된 경우, Amazon EC2 Auto Scaling이 모든 `InService` 인스턴스를 한 번에 바꿀 수 있습니다.

이러한 상황을 피하려면 인스턴스 가중치를 사용하도록 Auto Scaling 그룹을 구성합니다. 주제의 권장 사항을 따르세요. Auto Scaling 그룹에 가중치를 사용할 때 최대 가중치보다 큰 원하는 용량을 지정합니다.

- 1시간 제한: 인스턴스 새로 고침이 대기 인스턴스나 스케일 인 방비된 인스턴스를 교체하기 위해 대기 중이기 때문에 교체를 계속할 수 없거나 새 인스턴스가 건전성 체크를 통과하지 못하면 Amazon EC2 Auto Scaling에서 한 시간 동안 계속 재시도합니다. 문제 해결에 도움이 되는 상태 메시지도 제공합니다. 1시간 후에도 문제가 계속되면 교체 작업에 실패합니다. 이 제한은 일시적인 문제가 발생할 경우, 복구할 시간을 제공하기 위한 것입니다.
- 사용자 데이터를 통한 코드 배포: 매칭 건너뛰기는 사용자 데이터 스크립트에서 배포된 코드 변경 사항을 확인하지 않습니다. 사용자 데이터를 사용하여 새 코드를 가져와 새 인스턴스에 이러한 업데이트를 설치하는 경우 시작 템플릿 버전 업데이트 없이도 모든 인스턴스에 최신 코드가 전송되도록 건너뛰기 매칭을 끄는 것이 좋습니다.
- 업데이트 제한: 원하는 구성으로 인스턴스 새로 고침이 활성화된 상태에서 Auto Scaling 그룹의 시작 템플릿, 시작 구성 또는 혼합 인스턴스 정책을 업데이트하려고 하면 요청이 실패하고 다음 검증 오류가 발생합니다. `An active instance refresh with a desired configuration exists. All configuration options derived from the desired configuration are not available for update while the instance refresh is active.`

인스턴스 새로 고침의 기본값 이해

인스턴스 새로 고침을 시작하기 전에 인스턴스 새로 고침에 영향을 주는 다양한 기본 설정을 사용자 지정할 수 있습니다. 일부 환경설정 기본값은 콘솔을 사용하는지, 명령줄 (AWS CLI 또는 AWS SDK) 을 사용하는지에 따라 달라집니다.

다음 표에는 인스턴스 새로 고침 설정의 기본값이 나와 있습니다.

설정	AWS CLI 또는 SDK AWS	Amazon EC2 Auto Scaling 콘솔
CloudWatch 알람	비활성화됨(null)	Disabled(비활성)
자동 롤백	비활성화됨(`false`)	Disabled(비활성)
체크포인트	비활성화됨(`false`)	Disabled(비활성)
체크포인트 지연	1시간(`3600`초)	1시간

설정	AWS CLI 또는 SDK AWS	Amazon EC2 Auto Scaling 콘솔
인스턴스 워밍업	기본 인스턴스 워밍업(정의된 경우) 또는 건전성 체크 유예 기간(정의되지 않은 경우)	기본 인스턴스 워밍업(정의된 경우) 또는 건전성 체크 유예 기간(정의되지 않은 경우)
최대 건전 백분율	인스턴스 유지 관리 정책에 따라 다릅니다. 인스턴스 유지 관리 정책이 없는 경우, 기본값은 100%(null)입니다.	인스턴스 유지 관리 정책에 따라 다릅니다. 인스턴스 유지 관리 정책이 없는 경우, 기본값은 100%(null)입니다.
최소 건전 백분율	인스턴스 유지 관리 정책에 따라 다릅니다. 인스턴스 유지 관리 정책이 없는 경우, 기본값은 90%입니다.	인스턴스 유지 관리 정책에 따라 다릅니다. 인스턴스 유지 관리 정책이 없는 경우, 기본값은 90%입니다.
스케일 인 방비 인스턴스	Wait	Ignore
매칭 건너뛰기	비활성화됨(`false`)	활성화됨
대기 인스턴스	Wait	Ignore

각 설정에 대한 설명은 다음과 같습니다:

CloudWatch 알람 (**`AlarmSpecification`**)

CloudWatch 알람 사양. CloudWatch 알람을 사용하여 문제를 식별하고 알람이 `ALARM` 상태가 되면 작업을 중단할 수 있습니다. 자세한 정보는 자동 롤백과 함께 인스턴스 새로 고침 시작을 참조하세요.

자동 롤백(**`AutoRollback`**)

인스턴스 새로 고침이 실패한다면 Amazon EC2 Auto Scaling이 Auto Scaling 그룹을 이전 구성으로 복귀시킬지의 여부를 통제합니다. 자세한 설명은 수동 또는 자동 롤백으로 변경 취소 섹션을 참조하세요.

체크포인트(**`CheckpointPercentages`**)

Amazon EC2 Auto Scaling이 인스턴스를 단계적으로 교체할지 여부를 제어합니다. 이 기능은 모든 인스턴스를 교체하기 전에 인스턴스에 대한 검증을 수행해야 하는 경우에 유용합니다. 자세한 설명은 인스턴스 새로 고침에 체크포인트 추가 섹션을 참조하세요.

체크포인트 지연(**`CheckpointDelay`**)

계속하기 전에 체크포인트 후 기다리는 시간(초)입니다. 자세한 설명은 인스턴스 새로 고침에 체크포인트 추가 섹션을 참조하세요.

인스턴스 워밍업(**`InstanceWarmup`**)

Amazon EC2 Auto Scaling이 다음 인스턴스 교체로 넘어가기 전에 새 인스턴스가 초기화를 완료한 것으로 간주될 때까지 기다리는 기간(초)입니다. Auto Scaling 그룹의 기본 인스턴스 워밍업을 이미 올바르게 정의한 경우에는 인스턴스 워밍업을 변경할 필요가 없습니다(기본값을 재정의하려는 경우, 제외). 자세한 설명은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 섹션을 참조하세요.

최대 건전 백분율 (**`MaxHealthyPercentage`**)

인스턴스를 교체할 때 그룹이 늘릴 수 있는 Auto Scaling 그룹의 원하는 용량의 백분율입니다.

최소 건전 백분율 (**`MinHealthyPercentage`**)

조업이 계속되려면 Auto Scaling 그룹이 서비스 중이고 건전하며 사용할 준비가 되어 있어야 하는 원하는 용량의 백분율입니다.

스케일 인 방비 인스턴스(**`ScaleInProtectedInstances`**)

스케일 인이 방비된 인스턴스가 발견되는 경우, Amazon EC2 Auto Scaling이 무엇을 할지를 통제합니다. 이러한 인스턴스에 대한 자세한 설명은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 섹션을 참조하세요.

Amazon EC2 Auto Scaling은 다음과 같은 옵션을 제공합니다.

- Replace (`Refresh`) - 스케일 인으로부터 보호되는 인스턴스를 대체합니다.
- Ignore (`Ignore`) - 스케일 인으로부터 보호되는 인스턴스는 무시하고 보호되지 않는 인스턴스는 계속 교체합니다.
- Wait (`Wait`) - 스케일 인 보호를 제거할 때까지 한 시간 동안 기다립니다. 그렇게 하지 않으면 인스턴스 새로 고침이 실패합니다.

매칭 건너뛰기(**`SkipMatching`**)

Amazon EC2 Auto Scaling이 원하는 구성과 일치하는 인스턴스 교체를 건너뛸지의 여부를 통제합니다. 원하는 구성이 지정되지 않은 경우, 인스턴스 새로 고침이 시작되기 전에 Auto Scaling 그룹이

사용했던 것과 동일한 시작 템플릿 및 인스턴스 유형을 가진 인스턴스 교체를 건너뜁니다. 자세한 설명은 매칭 건너뛰기와 함께 인스턴스 새로 고침 사용 섹션을 참조하세요.

대기 인스턴스(**`StandbyInstances`**)

`Standby` 상태의 인스턴스가 발견되는 경우, Amazon EC2 Auto Scaling이 무엇을 할지를 통제합니다. 이러한 인스턴스에 대한 자세한 설명은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 섹션을 참조하세요.

Amazon EC2 Auto Scaling은 다음과 같은 옵션을 제공합니다.

- Terminate (`Terminate`) - 실행 중인 인스턴스를 종료합니다. `Standby`
- Ignore (`Ignore`) — 상태에 있는 인스턴스를 `Standby` 무시하고 해당 상태에 있는 인스턴스를 계속 대체합니다`InService`.
- Wait (`Wait`) — 인스턴스를 서비스 상태로 되돌릴 때까지 한 시간 동안 기다립니다. 그렇게 하지 않으면 인스턴스 새로 고침이 실패합니다.

또는 를 사용하여 인스턴스 새로 고침을 AWS Management Console 시작합니다. AWS CLI

> **Important**
>
> 진행 중인 인스턴스 새로 고침을 롤백하여 변경을 취소할 수 있습니다. 이를 위해서는 Auto Scaling 그룹이 인스턴스 새로 고침을 시작하기 전에 롤백을 사용하기 위한 사전 조건을 충족해야 합니다. 자세한 정보는 수동 또는 자동 롤백으로 변경 취소을 참조하세요.

다음 절차는 or를 사용하여 인스턴스 새로 고침을 시작하는 데 도움이 AWS CLI됩니다. AWS Management Console

인스턴스 새로 고침 시작(콘솔)

인스턴스 새로 고침을 처음 생성하는 경우, 콘솔을 사용하면 사용 가능한 기능과 옵션을 이해하는 데 도움이 됩니다.

콘솔에서 인스턴스 새로 고침 시작(기본 절차)

이전에 Auto Scaling 그룹에 대해 혼합 인스턴스 정책을 정의하지 않은 경우, 다음 절차를 따릅니다. 이전에 혼합 인스턴스 정책을 정의한 경우, 콘솔에서 인스턴스 새로 고침 시작(혼합 인스턴스 그룹)의 내용을 참조하여 인스턴스 새로 고침을 시작하세요.

인스턴스 새로 고침을 시작하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 아래쪽에 분할 창이 열립니다.
3. 인스턴스 새로 고침(Instance refresh) 탭의 활성 인스턴스 새로 고침(Active Instance refreshes)에서 인스턴스 새로 고침 시작(Start instance refresh)을 선택합니다.
4. 가용성 설정에서 다음을 수행하십시오:

   a. 인스턴스 교체 방법:

      - Auto Scaling 그룹에 인스턴스 유지 관리 정책을 설정하지 않은 경우, 인스턴스 교체 방법의 기본 설정은 해지 후 시작입니다. 이것은 인스턴스 새로 고침의 기존 기본 동작입니다.
      - Auto Scaling 그룹에 인스턴스 유지 관리 정책을 설정하면 인스턴스 교체 방법에 대한 기본값이 제공됩니다. 인스턴스 유지 관리 정책을 재정의하려면 재정의를 선택합니다. 재정의는 현재 인스턴스 새로 고침에만 적용됩니다. 다음에 인스턴스 새로 고침을 시작하면 이 값은 인스턴스 유지 관리 정책 기본값으로 재설정됩니다.

      다음 절차에서는 인스턴스 교체 방법을 업데이트하는 방법을 설명합니다.

      i. 다음 인스턴스 교체 방법 중 하나를 선택합니다.

         - 해지 전 시작: 기존 인스턴스를 해지하기 전에 새 인스턴스를 프로비저닝해야 합니다. 이 방식은 비용 절감보다 가용성을 선호하는 경우에 적합합니다.
         - 해지 후 시작: 기존 인스턴스가 해지되는 것과 동시에 새 인스턴스가 시작됩니다. 이 방식은 가용성보다 비용 절감을 선호하는 경우에 적합합니다. 또한 현재 사용 가능한 용량보다 더 많은 용량을 시작시켜서는 안되는 경우에도 적합합니다.
         - 사용자 지정 동작: 이 옵션을 사용하면 인스턴스를 교체할 때 사용할 수 있는 용량에 대한 사용자 지정 최소 및 최대 범위를 설정할 수 있습니다. 이는 비용과 가용성의 균형을 적절하게 조정하는 데 도움이 될 수 있습니다.

      ii. 건전 백분율 설정의 경우, 다음 필드 중 하나 또는 둘 다에 값을 입력합니다. 활성화 필드는 인스턴스 교체 방법에 대해 선택한 옵션에 따라 달라집니다.

         - 최소: 인스턴스 새로 고침을 진행하는 데 필요한 최소 건전 백분율을 설정합니다.
         - 최대: 인스턴스 새로 고침 작업 중 가능한 최대 건전 백분율을 설정합니다.

iii. 현재의 그룹 크기에 근거한 교체 시의 추정 임시 용량 보기 섹션을 펼쳐서 최소 및 최대 값이 귀하의 그룹에 어떻게 적용되는지를 확인하십시오. 사용되는 정확한 값은 원하는 용량 값에 의존하며, 이 값은 그룹이 조정되면 바뀝니다.

iv. 유효하지 않은 교체 크기에 대한 교체 동작 설정 섹션을 펼친 다음 가용성의 우선순위를 정하기 위해 최대 건전 백분율 위반 또는 최소 건전 백분율 위반 중 하나를 선택합니다.

기본값인 최소 건전 백분율 위반 옵션을 유지하는 것은 매우 작은 그룹의 경우, 권장되지 않습니다. 시작하기 전에 인스턴스가 해지됨: Auto Scaling 그룹에 인스턴스가 하나만 있는 경우, 인스턴스 새로 고침을 시작하면 인스턴스가 중단될 수 있습니다.

이 단계에서는 아직 인스턴스 유지 관리 정책이 없는 Auto Scaling 그룹을 사용하는 경우, 폴백 동작을 구성합니다. 이 옵션은 사용할 수 없으며 그룹에 인스턴스 유지 관리 정책이 있는 경우에는 표시되지 않습니다. 이 옵션은 해지 후 시작 교체 방법에서만 사용할 수 있습니다. 다른 교체 방법은 가용성의 우선순위를 정하기 위해 최대 건전 백분율을 위반하게 됩니다.

b. 인스턴스 워밍업의 경우, 그 초기화가 완료될 때 새 인스턴스의 상태가 `InService`로 바뀐 때로 부터의 시간(초)을 입력합니다. Amazon EC2 Auto Scaling은 다음 인스턴스 교체를 진행하기 전에 이 시간 동안 기다립니다.

워밍업 중에 새로 시작된 인스턴스는 Auto Scaling 그룹의 집계된 인스턴스 지표 (`CPUUtilization`, `NetworkIn`, `NetworkOut` 등)로 계산되지 않습니다. Auto Scaling 그룹에 조정 정책을 추가한 경우, 크기 조정 활동이 병렬로 실행됩니다. 인스턴스 새로 고침 준비 기간을 길게 설정하면 새로 시작한 인스턴스가 지표에 표시되는 데 시간이 더 걸립니다. 따라서 준비 기간이 적절하면 Amazon EC2 Auto Scaling이 오래된 지표 데이터를 기반으로 규모를 조정할 수 없습니다.

Auto Scaling 그룹의 기본 인스턴스 워밍업을 이미 올바르게 정의한 경우에는 인스턴스 워밍업을 변경할 필요가 없습니다. 그러나 기본값을 재정의하려면 이 옵션에 값을 설정하면 됩니다. 기본 인스턴스 워밍업 설정에 대한 자세한 설명은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 섹션을 참조하세요.

5. 새로 고침 설정에서 다음을 수행하십시오:

a. (옵션) 인스턴스 새로 고침에 대한 증분 또는 단계적 접근 방식을 사용하여 인스턴스를 교체하도록 체크포인트(Checkpoints)에서 체크포인트 사용(Enable checkpoints)을 선택합니다. 이렇게 하면 교체 세트 간의 검증을 위한 추가 시간이 제공됩니다. 체크포인트를 사용하지 않도록 선택하는 경우, 거의 연속적인 한 번의 작업으로 인스턴스가 교체됩니다.

체크포인트를 사용하도록 설정하는 경우, 체크포인트 활성화(콘솔)에서 추가 단계를 참조하세요.

b. 매칭 건너뛰기(Skip matching) 사용 또는 해제:

- 시작 템플릿과 이미 일치하는 인스턴스의 교체를 건너뛰려면 매칭 건너뛰기 활성화 확인란을 선택된 상태로 둡니다.
- 이 확인란의 선택을 취소하여 매칭 건너뛰기를 해제하면 모든 인스턴스를 교체할 수 있습니다.

매칭 건너뛰기를 활성화하면 새 시작 템플릿을 설정하거나 또는 기존의 시작 템플릿을 사용하는 대신 새 버전의 시작 템플릿을 사용할 수 있습니다. 인스턴스 새로 고침 시작 페이지의 원하는 구성 섹션에서 이 작업을 수행하십시오.

> **Note**
>
> 매칭 건너뛰기 기능을 사용하여 현재 시작 구성을 사용하는 Auto Scaling 그룹을 업데이트하려면 Desired configuration(원하는 구성)에서 시작 템플릿을 선택해야 합니다. 시작 구성시 매칭 건너뛰기는 지원되지 않습니다.

c. 대기 인스턴스에서 무시, 해지 또는 대기를 선택합니다. 이에 따라 인스턴스가 `Standby` 상태인 경우, 수행되는 작업이 결정됩니다. 자세한 설명은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 섹션을 참조하세요.

대기를 선택하면 이러한 인스턴스를 서비스 상태로 되돌리는 추가 단계를 수행해야 합니다. 그렇지 않으면 인스턴스 새로 고침이 모든 `InService` 인스턴스를 교체하고 1시간 동안 기다립니다. 그런 다음 `Standby` 인스턴스가 남아 있으면 인스턴스 새로 고침이 실패합니다. 이 상황을 방지하려면 대신 이러한 인스턴스를 무시 또는 해지하도록 선택합니다.

d. 스케일 인 방비 인스턴스에서 무시, 바꾸기 또는 대기를 선택합니다. 이에 따라 스케일 인 방비 인스턴스가 발견되는 경우, 수행되는 작업이 결정됩니다. 자세한 설명은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 섹션을 참조하세요.

대기를 선택하면 이러한 인스턴스에서 스케일 인 방비를 제거하는 추가 단계를 수행해야 합니다. 그렇지 않으면 인스턴스 새로 고침이 보호되지 않는 모든 인스턴스를 교체하고 1시간 동안 기다립니다. 그런 다음 스케일 인 방비 인스턴스가 남아 있으면 인스턴스 새로 고침이 실패합니다. 이 상황을 방지하려면 대신 이러한 인스턴스를 무시 또는 교체하도록 선택합니다.

6. (선택 사항) CloudWatch 경보의 경우 [Enable CloudWatch alarms] 를 선택한 다음 하나 이상의 경보를 선택합니다. CloudWatch 알람을 사용하여 문제를 식별하고 알람이 상태가 되면 작업을 중단할 수 있습니다. ALARM 자세한 정보는 자동 롤백과 함께 인스턴스 새로 고침 시작을 참조하세요.
7. (옵션) 원하는 구성 섹션을 스케일 아웃하여 Auto Scaling 그룹에 적용할 업데이트를 지정합니다.

   이 단계에서는 콘솔 인터페이스에서 선택하는 대신 JSON 또는 YAML 구문을 사용하여 파라미터 값을 편집하도록 선택할 수 있습니다. 이렇게 하려면 콘솔 인터페이스 사용(Use console interface) 대신 코드 편집기 사용(Use code editor)을 선택합니다. 다음 절차에서는 콘솔을 사용하여 설정을 선택하는 방법을 설명합니다.

   a. 시작 템플릿 업데이트(Update launch template):

      - Auto Scaling 그룹에 대한 새 시작 템플릿 또는 새 시작 템플릿 버전을 생성하지 않은 경우, 이 확인란을 선택하지 않습니다.
      - Auto Scaling 그룹에 대한 새 시작 템플릿 또는 새 시작 템플릿 버전을 생성한 경우, 이 확인란을 선택합니다. 이 옵션을 선택하면 Amazon EC2 Auto Scaling에 현재 시작 템플릿과 현재 시작 템플릿 버전이 표시됩니다. 사용 가능한 다른 버전도 열거됩니다. 시작 템플릿을 선택한 다음 버전을 선택합니다.

        버전을 선택하면 버전 정보가 표시됩니다. 이 버전은 인스턴스 새로 고침의 일부로 인스턴스를 교체할 때 사용되는 시작 템플릿 버전입니다. 인스턴스 새로 고침이 성공하면 그룹이 스케일 아웃될 때와 같이 새 인스턴스가 시작될 때마다 이 버전의 시작 템플릿도 사용됩니다.

   b. 시작 템플릿의 인스턴스 유형을 재정의할 일련의 인스턴스 유형 및 구매 옵션 선택(Choose a set of instance types and purchase options to override the instance type in the launch template):

      - 시작 템플릿에서 지정한 인스턴스 유형 및 구매 옵션을 사용하려면 이 확인란을 선택하지 않습니다.
      - 시작 템플릿의 인스턴스 유형을 재정의하거나 스팟 인스턴스를 시작하려면 이 확인란을 선택합니다. 각 인스턴스 유형을 수동으로 추가하거나 기본 인스턴스 유형을 선택하고 일치하는 추가 인스턴스 유형을 검색하는 권장 옵션을 선택할 수 있습니다. 스팟 인스턴스를 시작할 계획이면 몇 가지 다른 인스턴스 유형을 추가하는 것이 좋습니다. 이러한 방식으로 Amazon EC2 Auto Scaling은 선택한 가용 영역에 인스턴스 용량이 부족한 경우, 다른 인스턴스 유형을 시작할 수 있습니다. 자세한 설명은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.

> **Warning**
> 스팟 인스턴스 중단을 처리할 수 없는 애플리케이션에는 스팟 인스턴스를 사용하지 마세요. Amazon EC2 스팟 서비스에서 용량을 회수해야 하는 경우, 중단이 발생할 수 있습니다.

이 확인란을 선택하는 경우, 시작 템플릿이 아직 스팟 인스턴스를 요청하지 않았는지 확인하세요. 스팟 인스턴스를 요청하는 시작 템플릿을 사용하면 여러 인스턴스 유형을 사용하고 스팟 및 온디맨드 인스턴스를 시작하는 Auto Scaling 그룹을 생성할 수 없습니다.

> **Note**
> 현재 시작 구성을 사용하는 Auto Scaling 그룹에 이러한 옵션을 구성하려면 Update launch template(시작 템플릿 업데이트)에서 시작 템플릿을 선택해야 합니다. 시작 구성에서 인스턴스 유형 재정의는 지원되지 않습니다.

8. (옵션) 롤백 설정에서 자동 롤백 활성화를 선택하여 인스턴스 새로 고침이 실패할 경우, 자동으로 롤백합니다.

   Auto Scaling 그룹이 롤백을 사용하기 위한 사전 조건을 충족하는 경우에만 이 설정을 활성화할 수 있습니다.

   자세한 설명은 수동 또는 자동 롤백으로 변경 취소 섹션을 참조하세요.
9. 모든 선택 사항을 검토하여 모든 항목이 올바르게 설정되었는지 확인하세요.

   이 시점에서 현재 및 제안된 변경 사항 간의 차이가 예기치 않거나 원치 않는 방식으로 애플리케이션에 영향을 미치지 않는지 확인하는 것이 좋습니다. 인스턴스 유형이 시작 템플릿과 호환되는지 확인하려면 인스턴스 유형 호환성 섹션을 참조하세요.
10. 인스턴스 새로 고침 선택에 만족하면 인스턴스 새로 고침 시작을 선택하세요.

콘솔에서 인스턴스 새로 고침 시작(혼합 인스턴스 그룹)

혼합 인스턴스 정책을 포함하는 Auto Scaling 그룹을 생성한 경우, 다음 절차를 따릅니다. 그룹에 대한 혼합 인스턴스 정책을 정의하지 않은 경우, 콘솔에서 인스턴스 새로 고침 시작(기본 절차)에서 인스턴스 새로 고침을 시작하는 방법을 참조하세요.

인스턴스 새로 고침을 시작하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 아래쪽에 분할 창이 열립니다.
3. 인스턴스 새로 고침(Instance refresh) 탭의 활성 인스턴스 새로 고침(Active Instance refreshes)에서 인스턴스 새로 고침 시작(Start instance refresh)을 선택합니다.
4. 가용성 설정에서 다음을 수행하십시오:

   a. 인스턴스 교체 방법:

      - Auto Scaling 그룹에 인스턴스 유지 관리 정책을 설정하지 않은 경우, 인스턴스 교체 방법의 기본 설정은 해지 후 시작입니다. 이것은 인스턴스 새로 고침의 기존 기본 동작입니다.
      - Auto Scaling 그룹에 인스턴스 유지 관리 정책을 설정하면 인스턴스 교체 방법에 대한 기본값이 제공됩니다. 인스턴스 유지 관리 정책을 재정의하려면 재정의를 선택합니다. 재정의는 현재 인스턴스 새로 고침에만 적용됩니다. 다음에 인스턴스 새로 고침을 시작하면 이 값은 인스턴스 유지 관리 정책 기본값으로 재설정됩니다.

      다음 절차에서는 인스턴스 교체 방법을 업데이트하는 방법을 설명합니다.

      i. 다음 인스턴스 교체 방법 중 하나를 선택합니다.

         - 해지 전 시작: 기존 인스턴스를 해지하기 전에 새 인스턴스를 프로비저닝해야 합니다. 이 방식은 비용 절감보다 가용성을 선호하는 경우에 적합합니다.
         - 해지 후 시작: 기존 인스턴스가 해지되는 것과 동시에 새 인스턴스가 시작됩니다. 이 방식은 가용성보다 비용 절감을 선호하는 경우에 적합합니다. 또한 현재 사용 가능한 용량보다 더 많은 용량을 시작시켜서는 안되는 경우에도 적합합니다.
         - 사용자 지정 동작: 이 옵션을 사용하면 인스턴스를 교체할 때 사용할 수 있는 용량에 대한 사용자 지정 최소 및 최대 범위를 설정할 수 있습니다. 이는 비용과 가용성의 균형을 적절하게 조정하는 데 도움이 될 수 있습니다.

      ii. 건전 백분율 설정의 경우, 다음 필드 중 하나 또는 둘 다에 값을 입력합니다. 활성화 필드는 인스턴스 교체 방법에 대해 선택한 옵션에 따라 달라집니다.

         - 최소: 인스턴스 새로 고침을 진행하는 데 필요한 최소 건전 백분율을 설정합니다.
         - 최대: 인스턴스 새로 고침 작업 중 가능한 최대 건전 백분율을 설정합니다.

iii. 현재의 그룹 크기에 근거한 교체 시의 추정 임시 용량 보기 섹션을 펼쳐서 최소 및 최대 값이 귀하의 그룹에 어떻게 적용되는지를 확인하십시오. 사용되는 정확한 값은 원하는 용량 값에 의존하며, 이 값은 그룹이 조정되면 바뀝니다.

iv. 유효하지 않은 교체 크기에 대한 교체 동작 설정 섹션을 펼친 다음 가용성의 우선순위를 정하기 위해 최대 건전 백분율 위반 또는 최소 건전 백분율 위반 중 하나를 선택합니다.

기본값인 최소 건전 백분율 위반 옵션을 유지하는 것은 매우 작은 그룹의 경우, 권장되지 않습니다. 시작하기 전에 인스턴스가 해지됨: Auto Scaling 그룹에 인스턴스가 하나만 있는 경우, 인스턴스 새로 고침을 시작하면 인스턴스가 중단될 수 있습니다.

이 단계에서는 아직 인스턴스 유지 관리 정책이 없는 Auto Scaling 그룹을 사용하는 경우, 폴백 동작을 구성합니다. 이 옵션은 사용할 수 없으며 그룹에 인스턴스 유지 관리 정책이 있는 경우에는 표시되지 않습니다. 이 옵션은 해지 후 시작 교체 방법에서만 사용할 수 있습니다. 다른 교체 방법은 가용성의 우선순위를 정하기 위해 최대 건전 백분율을 위반하게 됩니다.

b. 인스턴스 워밍업의 경우, 그 초기화가 완료될 때 새 인스턴스의 상태가 `InService`로 바뀐 때로 부터의 시간(초)을 입력합니다. Amazon EC2 Auto Scaling은 다음 인스턴스 교체를 진행하기 전에 이 시간 동안 기다립니다.

워밍업 중에 새로 시작된 인스턴스는 Auto Scaling 그룹의 집계된 인스턴스 지표 (`CPUUtilization`, `NetworkIn`, `NetworkOut` 등)로 계산되지 않습니다. Auto Scaling 그룹에 조정 정책을 추가한 경우, 크기 조정 활동이 병렬로 실행됩니다. 인스턴스 새로 고침 준비 기간을 길게 설정하면 새로 시작된 인스턴스가 지표에 표시되는 데 시간이 더 걸립니다. 따라서 준비 기간이 적절하면 Amazon EC2 Auto Scaling이 오래된 지표 데이터를 기반으로 규모를 조정할 수 없습니다.

Auto Scaling 그룹의 기본 인스턴스 워밍업을 이미 올바르게 정의한 경우에는 인스턴스 워밍업을 변경할 필요가 없습니다. 그러나 기본값을 재정의하려면 이 옵션에 값을 설정하면 됩니다. 기본 인스턴스 워밍업 설정에 대한 자세한 설명은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 섹션을 참조하세요.

5. 새로 고침 설정에서 다음을 수행하십시오:

a. (옵션) 인스턴스 새로 고침에 대한 증분 또는 단계적 접근 방식을 사용하여 인스턴스를 교체하도록 체크포인트(Checkpoints)에서 체크포인트 사용(Enable checkpoints)을 선택합니다. 이렇게 하면 교체 세트 간의 검증을 위한 추가 시간이 제공됩니다. 체크포인트를 사용하지 않도록 선택하는 경우, 거의 연속적인 한 번의 작업으로 인스턴스가 교체됩니다.

체크포인트를 사용하도록 설정하는 경우, 체크포인트 활성화(콘솔)에서 추가 단계를 참조하세요.

b. 매칭 건너뛰기(Skip matching) 사용 또는 해제:

- 시작 템플릿 및 인스턴스 유형 재정의와 이미 일치하는 인스턴스의 교체를 건너뛰려면 매칭 건너뛰기 활성화 확인란을 선택된 상태로 둡니다.
- 이 확인란의 선택을 취소하여 매칭 건너뛰기를 끄면 모든 인스턴스를 교체할 수 있습니다.

매칭 건너뛰기를 활성화하면 새 시작 템플릿을 설정하거나 또는 기존의 시작 템플릿을 사용하는 대신 새 버전의 시작 템플릿을 사용할 수 있습니다. 인스턴스 새로 고침 시작 페이지의 원하는 구성 섹션에서 이 작업을 수행하십시오. 원하는 구성에서 인스턴스 유형 재정의를 업데이트할 수도 있습니다.

c. 대기 인스턴스에서 무시, 해지 또는 대기를 선택합니다. 이에 따라 인스턴스가 `Standby` 상태인 경우, 수행되는 작업이 결정됩니다. 자세한 설명은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 섹션을 참조하세요.

대기를 선택하면 이러한 인스턴스를 서비스 상태로 되돌리는 추가 단계를 수행해야 합니다. 그렇지 않으면 인스턴스 새로 고침이 모든 `InService` 인스턴스를 교체하고 1시간 동안 기다립니다. 그런 다음 `Standby` 인스턴스가 남아 있으면 인스턴스 새로 고침이 실패합니다. 이 상황을 방지하려면 대신 이러한 인스턴스를 무시 또는 해지하도록 선택합니다.

d. 스케일 인 방비 인스턴스에서 무시, 바꾸기 또는 대기를 선택합니다. 이에 따라 스케일 인 방비 인스턴스가 발견되는 경우, 수행되는 작업이 결정됩니다. 자세한 설명은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 섹션을 참조하세요.

대기를 선택하면 이러한 인스턴스에서 스케일 인 방비를 제거하는 추가 단계를 수행해야 합니다. 그렇지 않으면 인스턴스 새로 고침이 보호되지 않는 모든 인스턴스를 교체하고 1시간 동안 기다립니다. 그런 다음 스케일 인 방비 인스턴스가 남아 있으면 인스턴스 새로 고침이 실패합니다. 이 상황을 방지하려면 대신 이러한 인스턴스를 무시 또는 교체하도록 선택합니다.

6. (선택 사항) CloudWatch 경보의 경우 [Enable CloudWatch alarms] 를 선택한 다음 하나 이상의 경보를 선택합니다. CloudWatch 알람을 사용하여 문제를 식별하고 알람이 상태가 되면 작업을 중단할 수 있습니다. `ALARM` 자세한 정보는 자동 롤백과 함께 인스턴스 새로 고침 시작을 참조하세요.

7. 원하는 구성(Desired configuration) 섹션에서 다음을 수행하십시오:

이 단계에서는 콘솔 인터페이스에서 선택하는 대신 JSON 또는 YAML 구문을 사용하여 파라미터 값을 편집하도록 선택할 수 있습니다. 이렇게 하려면 콘솔 인터페이스 사용(Use console

interface) 대신 코드 편집기 사용(Use code editor)을 선택합니다. 다음 절차에서는 콘솔을 사용하여 설정을 선택하는 방법을 설명합니다.

a. 시작 템플릿 업데이트(Update launch template):

- Auto Scaling 그룹에 대한 새 시작 템플릿 또는 새 시작 템플릿 버전을 생성하지 않은 경우, 이 확인란을 선택하지 않습니다.
- Auto Scaling 그룹에 대한 새 시작 템플릿 또는 새 시작 템플릿 버전을 생성한 경우, 이 확인란을 선택합니다. 이 옵션을 선택하면 Amazon EC2 Auto Scaling에 현재 시작 템플릿과 현재 시작 템플릿 버전이 표시됩니다. 사용 가능한 다른 버전도 열거됩니다. 시작 템플릿을 선택한 다음 버전을 선택합니다.

  버전을 선택하면 버전 정보가 표시됩니다. 이 버전은 인스턴스 새로 고침의 일부로 인스턴스를 교체할 때 사용되는 시작 템플릿 버전입니다. 인스턴스 새로 고침이 성공하면 그룹이 스케일 아웃될 때와 같이 새 인스턴스가 시작될 때마다 이 버전의 시작 템플릿도 사용됩니다.

b. 이 설정을 사용하여 시작 템플릿에 정의된 인스턴스 유형 및 구매 옵션 재정의(Use these settings to override the instance type and purchase option defined in the launch template):

기본적으로 이 확인란은 선택되어 있습니다. Amazon EC2 Auto Scaling은 각 파라미터를 현재 Auto Scaling 그룹의 혼합 인스턴스 정책에 설정되어 있는 값으로 채웁니다. 변경하고자 하는 파라미터의 값만 업데이트하세요. 이러한 설정에 대한 지침은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.

> **Warning**
>
> 이 확인란의 선택은 취소하지 않는 것이 좋습니다. 혼합 인스턴스 정책 사용을 중지하려는 경우에만 선택을 취소하세요. 인스턴스를 새로 고침을 완료하면 Amazon EC2 Auto Scaling은 그룹을 원하는 구성(Desired configuration)과 일치하도록 업데이트합니다. 혼합 인스턴스 정책이 더 이상 포함되어 있지 않은 경우, Amazon EC2 Auto Scaling은 현재 실행 중인 스팟 인스턴스를 점차 해지하고 온디맨드 인스턴스로 교체합니다. 그렇지 않고 시작 템플릿이 스팟 인스턴스를 요청하는 경우, Amazon EC2 Auto Scaling은 현재 실행 중인 온디맨드 인스턴스를 점차 해지하고 스팟 인스턴스로 교체합니다.

8. (옵션) 롤백 설정에서 자동 롤백 활성화를 선택하여 인스턴스 새로 고침이 실패할 경우, 자동으로 롤백합니다.

Auto Scaling 그룹이 롤백을 사용하기 위한 사전 조건을 충족하는 경우에만 이 설정을 활성화할 수 있습니다.

자세한 설명은 수동 또는 자동 롤백으로 변경 취소 섹션을 참조하세요.

9. 모든 선택 사항을 검토하여 모든 항목이 올바르게 설정되었는지 확인하세요.

   이 시점에서 현재 및 제안된 변경 사항 간의 차이가 예기치 않거나 원치 않는 방식으로 애플리케이션에 영향을 미치지 않는지 확인하는 것이 좋습니다. 인스턴스 유형이 시작 템플릿과 호환되는지 확인하려면 인스턴스 유형 호환성 섹션을 참조하세요.

   인스턴스 새로 고침 선택에 만족하면 인스턴스 새로 고침 시작을 선택하세요.

인스턴스 새로 고침 시작(AWS CLI)

인스턴스 새로 고침을 시작하려면

다음 start-instance-refresh명령을 사용하여 에서 인스턴스 새로 고침을 AWS CLI시작합니다. JSON 구성 파일에서 변경하려는 기본 설정을 지정할 수 있습니다. 구성 파일을 참조할 때 다음 예와 같이 파일 경로와 이름을 제공합니다.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json`의 콘텐츠:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 50,
      "AutoRollback": true,
      "ScaleInProtectedInstances": Ignore,
      "StandbyInstances": Terminate
    }
}
```

기본 설정이 제공되지 않으면 기본값이 사용됩니다. 자세한 설명은 인스턴스 새로 고침의 기본값 이해 섹션을 참조하세요.

출력 예제:

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

AWS Management Console 또는 를 사용하여 인스턴스 새로 고침을 모니터링합니다. AWS CLI

또는 를 사용하여 진행 중인 인스턴스 새로 고침을 모니터링하거나 지난 6주 동안의 이전 인스턴스 새로 고침 상태를 조회할 수 있습니다. AWS Management Console AWS CLI

인스턴스 새로 고침 상태를 모니터링하고 확인합니다.

인스턴스 새로 고침 상태를 모니터링하고 확인하려면 다음 방법 중 하나를 사용하십시오.

Console

> **Tip**
> 이 절차에서는 이름이 지정된 열이 이미 표시되어 있어야 합니다. 숨겨진 열을 표시하거나 표시된 행 수를 변경하려면 섹션 오른쪽 상단의 기어 아이콘을 선택하여 기본 설정 모달을 엽니다. 필요에 따라 설정을 업데이트하고 확인을 선택합니다.

인스턴스 새로 고침 상태를 모니터링하고 확인하려면 (콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 인스턴스 새로 고침(Instance refresh) 탭의 인스턴스 새로 고침 기록(Instance refresh history)에서 상태(Status) 열을 보고 요청 상태를 확인할 수 있습니다. 작업이 초기화되는 동안 `Pending` 상태가 됩니다. 그런 다음 상태는 `InProgress`로 빠르게 변경됩니다. 모든 인스턴스가 업데이트되면 상태가 `Successful`로 변경됩니다.
4. 그룹의 조정 활동을 확인하여 진행 중인 활동의 성공 또는 실패를 추가로 모니터링할 수 있습니다. 활동(Actvity) 탭의 활동 기록(Activity history)에서, 인스턴스 새로 고침을 시작하면 인스턴스가 해지되었을 때의 항목과 인스턴스가 시작되었을 때의 다른 항목 집합이 표시됩니다. 스케일링 활동이 많으면 활동 기록 상단의 > 아이콘을 선택하여 더 많은 스케일링 활동을 확인

할 수 있습니다. 활동 실패의 원인이 될 수 있는 문제 해결에 대한 자세한 내용은 을 참조하십시오Amazon EC2 Auto Scaling의 문제 해결.

5. (선택 사항) 인스턴스 관리 탭의 인스턴스에서 필요에 따라 특정 인스턴스의 진행 상황을 검토할 수 있습니다.

AWS CLI

인스턴스 새로 고침 상태를 모니터링하고 확인하려면 (AWS CLI)

다음 describe-instance-refreshes 명령을 사용합니다.

```
aws autoscaling describe-instance-refreshes --auto-scaling-group-name my-asg
```

출력의 예제는 다음과 같습니다.

인스턴스 새로 고침은 시작 시간을 기준으로 정렬됩니다. 아직 진행 중인 인스턴스 새로 고침이 먼저 설명됩니다.

```
{
  "InstanceRefreshes":[
    {
      "InstanceRefreshId":"08b91cf7-8fa6-48af-b6a6-d227f40f1b9b",
      "AutoScalingGroupName":"my-asg",
      "Status":"InProgress",
      "StatusReason":"Waiting for instances to warm up before continuing. For
 example: i-0645704820a8e83ff is warming up.",
      "StartTime":"2023-11-24T16:46:52+00:00",
      "PercentageComplete":50,
      "InstancesToUpdate":0,
      "Preferences":{
        "MaxHealthyPercentage":120,
        "MinHealthyPercentage":90,
        "InstanceWarmup":60,
        "SkipMatching":false,
        "AutoRollback":true,
        "ScaleInProtectedInstances":"Ignore",
        "StandbyInstances":"Ignore"
      }
    },
    {
      "InstanceRefreshId":"0e151305-1e57-4a32-a256-1fd14157c5ec",
      "AutoScalingGroupName":"my-asg",
```

```
      "Status":"Successful",
      "StartTime":"2023-11-22T13:53:37+00:00",
      "EndTime":"2023-11-22T13:59:45+00:00",
      "PercentageComplete":100,
      "InstancesToUpdate":0,
      "Preferences":{
        "MaxHealthyPercentage":120,
        "MinHealthyPercentage":90,
        "InstanceWarmup":60,
        "SkipMatching":false,
        "AutoRollback":true,
        "ScaleInProtectedInstances":"Ignore",
        "StandbyInstances":"Ignore"
      }
    }
  ]
}
```

그룹의 조정 활동을 확인하여 진행 중인 활동의 성공 또는 실패를 추가로 모니터링할 수 있습니다. 또한 조정 활동을 통해 인스턴스 새로 고침과 관련된 문제를 해결하는 데 도움이 되는 세부 정보를 자세히 살펴볼 수 있습니다. 자세한 정보는 Amazon EC2 Auto Scaling의 문제 해결을 참조하세요.

인스턴스 새로 고침 상태

인스턴스 새로 고침을 시작하면 대기 중 상태가 됩니다. 성공, 실패 RollbackSuccessful, 취소 또는 에 도달할 InProgress때까지 보류 중으로 넘어갑니다. RollbackFailed

인스턴스 새로 고침은 다음과 같은 상태일 수 있습니다.

상태 표시기	설명
보류중	요청이 생성되었지만 인스턴스 새로 고침이 시작되지 않았습니다.
InProgress	인스턴스 새로 고침이 진행 중입니다.
성공	인스턴스 새로 고침이 성공적으로 완료되었습니다.
실패	인스턴스 새로 고침을 완료하지 못했습니다. 상태 이유 및 조정 활동을 사용하여 문제를 해결할 수 있습니다.
취소 중	진행 중인 인스턴스 새로 고침이 취소되고 있습니다.

상태 표시기	설명
취소됨	인스턴스 새로 고침이 취소되었습니다.
RollbackInProgress	인스턴스 새로 고침이 롤백되고 있습니다.
RollbackFailed	롤백을 완료하지 못했습니다. 상태 이유 및 조정 활동을 사용하여 문제를 해결할 수 있습니다.
RollbackSuccessful	롤백이 성공적으로 완료되었습니다.

또는 를 사용하여 인스턴스 새로 고침을 AWS Management Console 취소하십시오. AWS CLI

아직 진행 중인 인스턴스 새로 고침을 취소할 수 있습니다. 완료된 후에는 취소할 수 없습니다.

인스턴스 새로 고침을 취소해도 이미 교체된 인스턴스는 롤백되지 않습니다. 인스턴스의 변경 사항을 롤백하려면 대신 롤백을 수행합니다. 자세한 설명은 수동 또는 자동 롤백으로 변경 취소 섹션을 참조하세요.

주제

- 인스턴스 새로 고침 취소(콘솔)
- 인스턴스 새로 고침 취소(AWS CLI)

인스턴스 새로 고침 취소(콘솔)

인스턴스 새로 고침을 취소하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.
3. 인스턴스 새로 고침 탭의 활성 인스턴스 새로 고침에서 작업, 취소를 선택합니다.
4. 확인 메시지가 표시되면 확인을 선택합니다.

인스턴스 새로 고침 상태가 취소 중으로 설정됩니다. 취소가 완료되면 인스턴스 새로 고침 상태가 취소됨으로 설정됩니다.

인스턴스 새로 고침 취소(AWS CLI)

인스턴스 새로 고침을 취소하려면

AWS CLI 및 의 cancel-instance-refresh명령을 사용하여 Auto Scaling 그룹 이름을 입력합니다.

```
aws autoscaling cancel-instance-refresh --auto-scaling-group-name my-asg
```

출력 예제:

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

수동 또는 자동 롤백으로 변경 취소

아직 진행 중인 인스턴스 새로 고침을 롤백할 수 있습니다. 완료된 후에는 롤백할 수 없습니다. 그러나 새 인스턴스 새로 고침을 시작하여 Auto Scaling 그룹을 다시 업데이트할 수 있습니다.

롤백 시 Amazon EC2 Auto Scaling은 지금까지 배치된 인스턴스를 교체합니다. 새 인스턴스는 인스턴스 새로 고침을 시작하기 전에 Auto Scaling 그룹에 마지막으로 저장한 구성과 일치합니다.

Amazon EC2 Auto Scaling은 다음과 같은 롤백 방법을 제공합니다.

- 수동 롤백: 롤백을 수동으로 시작하여 롤백 지점까지 배치된 내용을 되돌립니다.
- 자동 롤백: Amazon EC2 Auto Scaling은 어떤 CloudWatch 이유로든 인스턴스 새로 고침이 실패하거나 지정한 경보가 해당 상태로 전환되는 경우 배포된 내용을 자동으로 되돌립니다. `ALARM`

내용

- 고려 사항
- 수동으로 롤백 시작하기
- 자동 롤백과 함께 인스턴스 새로 고침 시작

고려 사항

롤백 사용시 다음 고려 사항이 적용됩니다:

- 롤백 옵션은 인스턴스 새로 고침을 시작할 때 원하는 구성을 지정한 경우에만 사용할 수 있습니다.
- 시작 템플릿의 이전 버전으로 롤백할 수 있는 버전은 특정 번호가 지정된 버전인 경우에만 가능합니다. Auto Scaling 그룹이 `$Latest` 또는 `$Default` 시작 템플릿 버전을 사용하도록 구성된 경우에는 롤백 옵션을 사용할 수 없습니다.
- 또한 AWS Systems Manager 파라미터 스토어의 AMI 별칭을 사용하도록 구성된 시작 템플릿으로 롤백할 수 없습니다.
- Auto Scaling 그룹에 마지막으로 저장한 구성이 안정적 상태에 있어야 합니다. 안정적 상태가 아닌 경우에도 롤백 워크플로우는 계속 발생하지만 결국 실패하게 됩니다. 문제를 해결할 때까지 Auto Scaling 그룹은 더 이상 인스턴스를 성공적으로 시작할 수 없는 실패 상태에 있을 수 있습니다. 이는 서비스 또는 애플리케이션의 가용성에 영향을 미칠 수 있습니다.

수동으로 롤백 시작하기

Console

수동으로 인스턴스 새로 고침의 롤백 시작 (콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.
3. 인스턴스 새로 고침 탭의 활성 인스턴스 새로 고침에서 작업, 롤백 시작을 선택합니다.
4. 확인 메시지가 표시되면 확인을 선택합니다.

AWS CLI

수동으로 인스턴스 새로 고침의 롤백 시작 (AWS CLI)

AWS CLI 및 의 rollback-instance-refresh명령을 사용하여 Auto Scaling 그룹 이름을 입력합니다.

```
aws autoscaling rollback-instance-refresh --auto-scaling-group-name my-asg
```

출력 예제:

```
{
    "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

> **Tip**
> 이 명령에서 오류가 발생하는 경우 AWS CLI 로컬에서 최신 버전으로 업데이트했는지 확인하십시오.

자동 롤백과 함께 인스턴스 새로 고침 시작

자동 롤백 기능을 사용하면 오류가 발생하거나 지정된 Amazon CloudWatch 경보가 `ALARM` 상태로 전환되는 경우와 같이 인스턴스 새로 고침이 실패할 때 자동으로 인스턴스 새로 고침을 롤백할 수 있습니다.

자동 롤백을 사용하도록 설정한 경우 인스턴스를 교체하는 동안 오류가 발생하면, 인스턴스 새로 고침은 1시간 동안 모든 교체를 완료하려고 시도한 후 실패하고 롤백합니다. 이러한 오류는 일반적으로 EC2 시작 실패, 잘못 구성된 상태 검사, `Standby` 상태 또는 스케일 인으로부터 보호되는 인스턴스의 해지를 무시하거나 허용하지 않는 등의 이유로 인해 발생합니다.

CloudWatch 알람 지정은 선택 사항입니다. 경보를 지정하려면 먼저 이를 생성해야 합니다. 지표 경보 및 복합 경보를 지정할 수 있습니다. 경보 생성에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오. Elastic Load Balancing 지표를 예로 들면, Application Load Balancer를 사용하는 경우, `HTTPCode_ELB_5XX_Count` 및 `HTTPCode_ELB_4XX_Count` 지표를 사용할 수 있습니다.

고려 사항

- CloudWatch 경보를 지정하지만 자동 롤백을 활성화하지 않고 경보 상태가 로 `ALARM` 전환되면 롤백하지 않고 인스턴스 새로 고침이 실패합니다.
- 인스턴스 새로 고침을 시작할 때 최대 10개의 경보를 선택할 수 있습니다.
- CloudWatch 경보를 선택할 때는 경보가 호환 가능한 상태여야 합니다. 경보 상태가 `INSUFFICIENT_DATA` 또는 `ALARM` 인 경우, 인스턴스 새로 고침을 시작하려고 할 때 오류가 발생합니다.
- Amazon EC2 Auto Scaling에서 사용할 경보를 생성할 때, 경보에는 누락된 데이터 요소를 처리하는 방법이 포함되어야 합니다. 설계에 따라 지표가 데이터 포인트를 자주 누락하는 경우, 경보 상태는 그 기간에 `INSUFFICIENT_DATA`입니다. 이 경우, 새 데이터 포인트를 찾을 때까지 Amazon EC2 Auto Scaling은 인스턴스를 교체할 수 없습니다. 경보가 이전의 `ALARM` 또는 `OK` 상태를 유지하도록 강제하기 위해, 귀하는 대신 누락된 데이터를 무시하도록 선택할 수 있습니다. 자세한 내용은 Amazon CloudWatch User Guide의 경보가 누락된 데이터를 처리하는 방법 구성을 참조하십시오.

Console

자동 롤백과 함께 인스턴스 새로 고침을 시작하려면 (콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.
3. 인스턴스 새로 고침(Instance refresh) 탭의 활성 인스턴스 새로 고침(Active Instance refreshes)에서 인스턴스 새로 고침 시작(Start instance refresh)을 선택합니다.
4. 필요한 경우 인스턴스 새로 고침 시작(콘솔) 절차에 따라 인스턴스 새로 고침 설정을 구성합니다.
5. (선택 사항) 새로 고침 설정에서 CloudWatch 경보에 대해 [CloudWatch 경보 활성화] 를 선택한 다음 하나 이상의 경보를 선택하여 문제를 식별하고 경보가 상태로 전환되면 작업을 중단합니다. `ALARM`
6. 롤백 설정에서 자동 롤백 사용을 선택하여 인스턴스 새로 고침을 시작하기 전에 Auto Scaling 그룹에 마지막으로 저장한 구성으로 실패한 인스턴스 새로 고침을 자동으로 롤백할 수 있습니다.
7. 선택한 내용을 검토한 다음 인스턴스 새로 고침 시작을 선택합니다.

AWS CLI

자동 롤백과 함께 인스턴스 새로 고침 시작(AWS CLI)

start-instance-refresh명령을 사용하고 의 `AutoRollback` 옵션에 `true` 대해 지정합니다. `Preferences`

다음 예는 장애 발생 시 자동으로 롤백되는 인스턴스 새로 고침을 시작하는 방법을 보여줍니다. *`italicized`* 파라미터 값을 자신의 값으로 바꾸세요.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json`의 콘텐츠.

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
```

```
            "LaunchTemplateName": "my-launch-template",
            "Version": "1"
        }
    },
    "Preferences": {
      "AutoRollback": true
    }
}
```

또는 인스턴스 새로 고침이 실패하거나 지정된 CloudWatch 경보가 ALARM 상태에 있을 때 자동으로 롤백하려면 다음 Preferences 예와 같이 에서 AlarmSpecification 옵션을 지정하고 경보 이름을 제공하십시오. *italicized* 파라미터 값을 자신의 값으로 바꾸세요.

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
            "LaunchTemplateName": "my-launch-template",
            "Version": "1"
        }
    },
    "Preferences": {
      "AutoRollback": true,
      "AlarmSpecification": { "Alarms": [ "my-alarm" ] }
    }
}
```

이 작업이 성공하면 다음과 비슷한 출력이 반환됩니다.

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

> **Tip**
>
> 이 명령에서 오류가 발생하는 경우 AWS CLI 로컬에서 최신 버전으로 업데이트했는지 확인하십시오.

매칭 건너뛰기와 함께 인스턴스 새로 고침 사용

매칭 건너뛰기는 Amazon EC2 Auto Scaling에서 이미 최신 업데이트가 있는 인스턴스를 무시하도록 합니다. 이렇게 하면 필요 이상으로 많은 인스턴스를 교체하지 않습니다. 이는 Auto Scaling 그룹이 특정 버전의 시작 템플릿을 사용하고 다른 버전을 사용하는 인스턴스만 교체하게 하려는 경우, 유용합니다.

매칭 건너뛰기에는 다음 고려 사항이 적용됩니다.

- 매칭 건너뛰기와 원하는 구성을 모두 사용하여 인스턴스 새로 고침을 시작하는 경우, Amazon EC2 Auto Scaling은 원하는 구성과 일치하는 인스턴스가 있는지 확인합니다. 그런 다음 원하는 구성과 일치하지 않는 인스턴스만 교체합니다. 인스턴스 새로 고침 성공 후 Amazon EC2 Auto Scaling이 원하는 구성을 반영하도록 그룹을 업데이트합니다.
- 매칭 건너뛰기와 함께 인스턴스 새로 고침을 시작했지만 원하는 구성을 지정하지 않은 경우, Amazon EC2 Auto Scaling은 Auto Scaling 그룹에 마지막으로 저장한 구성과 일치하는 인스턴스가 있는지 확인합니다. 그런 다음 마지막으로 저장된 구성과 일치하지 않는 인스턴스만 교체합니다.
- 새 시작 템플릿, 시작 템플릿의 새 버전 또는 인스턴스 유형 집합에 매칭 건너뛰기를 사용할 수 있습니다. 매칭 건너뛰기를 사용 설정했지만 어떠한 변경 사항도 없는 경우, 인스턴스를 교체하지 않고 인스턴스 새로 고침이 즉시 성공합니다. 원하는 구성을 변경한 경우(예: 스팟 할당 전략 변경) Amazon EC2 Auto Scaling은 인스턴스 새로 고침이 성공할 때까지 기다립니다. 그런 다음 Auto Scaling 그룹 설정을 업데이트하여 원하는 새 구성을 반영합니다.
- 새 시작 구성에서는 매칭 건너뛰기를 사용할 수 없습니다.
- 인스턴스 새로 고침을 시작하고 원하는 구성을 제공하면 Amazon EC2 Auto Scaling은 모든 인스턴스가 원하는 구성을 사용하도록 합니다. 따라서 시작 템플릿의 원하는 버전 중 하나를 `$Default` `$Latest` 지정하거나 인스턴스 새로 고침이 진행되는 동안 시작 템플릿의 새 버전을 생성하면 이미 교체된 모든 인스턴스가 다시 교체됩니다.
- 스킵 매칭은 시작 템플릿의 사용자 데이터 스크립트가 업데이트된 코드를 가져와 새 인스턴스에 설치할지 여부를 알 수 없습니다. 따라서 스킵 매칭을 수행하면 오래된 코드가 설치된 인스턴스 교체를 건너뛸 수 있습니다. 이 경우 시작 템플릿 버전 업데이트 없이도 모든 인스턴스가 최신 코드를 받을 수 있도록 스킵 매칭을 꺼야 합니다.

이 섹션에는 스킵 매칭을 활성화한 상태에서 인스턴스 새로 고침을 시작하는 AWS CLI 방법에 대한 지침이 포함되어 있습니다. 콘솔을 사용하는 방법은 인스턴스 새로 고침 시작(콘솔) 섹션을 참조하십시오.

매칭 건너뛰기(기본 절차)

이 섹션의 단계에 따라 를 AWS CLI 사용하여 다음을 수행하십시오.

- 인스턴스에 적용할 시작 템플릿을 생성합니다.
- 인스턴스 새로 고침을 시작하여 Auto Scaling 그룹에 시작 템플릿을 적용합니다. 매칭 건너뛰기를 활성화하지 않으면 모든 인스턴스가 교체됩니다. 이는 인스턴스를 프로비저닝하는 데 사용되는 시작 템플릿이 원하는 구성에 대해 지정한 시작 템플릿과 동일한 경우에도 마찬가지입니다.

새 시작 템플릿에 매칭 건너뛰기 사용

1. create-launch-template명령을 사용하여 Auto Scaling 그룹을 위한 새 시작 템플릿을 생성합니다. Auto Scaling 그룹에 대해 생성된 인스턴스의 세부 정보를 정의하는 `--launch-template-data` 옵션과 JSON 입력을 포함합니다.

   예컨대, 다음 명령을 사용하여 AMI ID *ami-0123456789abcdef0* 및 *t2.micro* 인스턴스 유형을 사용하여 기본 시작 템플릿을 생성합니다.

   ```
   aws ec2 create-launch-template --launch-template-name my-template-for-auto-scaling
    --version-description version1 \
     --launch-template-data
    '{"ImageId":"ami-0123456789abcdef0","InstanceType":"t2.micro"}'
   ```

   이 작업이 성공하면 다음과 비슷한 출력이 반환됩니다.

   ```
   {
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-068f72b729example",
        "LaunchTemplateName": "my-template-for-auto-scaling",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2023-01-30T18:16:06.000Z",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
     }
   }
   ```

   자세한 정보는 를 사용하여 시작 템플릿을 생성하고 관리하는 예제 AWS CLI을 참조하세요.

2. start-instance-refresh명령을 사용하여 인스턴스 교체 워크플로를 시작하고 ID를 *lt-068f72b729example*사용하여 새 시작 템플릿을 적용합니다. 시작 템플릿은 새 템플릿이므

로 버전이 하나만 있습니다. 즉, 시작 템플릿의 버전 1이 이 인스턴스 새로 고침의 대상입니다. 인스턴스 새로 고침 중 스케일아웃 이벤트가 발생하고 Amazon EC2 Auto Scaling이 이 시작 템플릿의 버전 1을 사용하여 새 인스턴스를 프로비저닝하는 경우, 인스턴스는 교체되지 않습니다. 작업이 성공적으로 완료되면 새 시작 템플릿이 Auto Scaling 그룹에 성공적으로 적용됩니다.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json`의 콘텐츠.

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredConfiguration": {
      "LaunchTemplate": {
          "LaunchTemplateId": "lt-068f72b729example",
          "Version": "$Default"
       }
    },
    "Preferences": {
      "SkipMatching": true
    }
}
```

이 작업이 성공하면 다음과 비슷한 출력이 반환됩니다.

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

매칭 건너뛰기(혼합 인스턴스 그룹)

혼합 인스턴스 정책을 사용하는 Auto Scaling 그룹이 있는 경우 이 섹션의 단계에 따라 를 사용하여 스킵 매칭으로 인스턴스 새로 고침을 시작하십시오. AWS CLI 다음과 같은 옵션이 있습니다:

- 정책에 지정된 모든 인스턴스 유형에 적용할 새 시작 템플릿을 제공합니다.
- 정책에서 시작 템플릿을 변경하거나 변경하지 않고 업데이트된 인스턴스 유형 세트를 제공합니다. 예컨대, 원하지 않는 인스턴스 유형에서 마이그레이션하려고 할 수 있습니다. AMI, 보안 그룹 또는 교체 중인 인스턴스의 기타 세부 사항을 변경하지 않고 시작 템플릿을 그대로 사용합니다.

필요에 맞는 옵션에 따라 다음 섹션 중 하나의 단계를 따르세요.

새 시작 템플릿에 매칭 건너뛰기 사용

1. create-launch-template명령을 사용하여 Auto Scaling 그룹을 위한 새 시작 템플릿을 생성합니다. Auto Scaling 그룹에 대해 생성된 인스턴스의 세부 정보를 정의하는 `--launch-template-data` 옵션과 JSON 입력을 포함합니다.

   예컨대, 다음 명령을 사용하여 AMI ID가 *ami-0123456789abcdef0*인 시작 템플릿을 생성합니다.

   ```
   aws ec2 create-launch-template --launch-template-name my-new-template --version-description version1 \
     --launch-template-data '{"ImageId":"ami-0123456789abcdef0"}'
   ```

   이 작업이 성공하면 다음과 비슷한 출력이 반환됩니다.

   ```
   {
      "LaunchTemplate": {
        "LaunchTemplateId": "lt-04d5cc9b88example",
        "LaunchTemplateName": "my-new-template",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2023-01-31T15:56:02.000Z",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
     }
   }
   ```

   자세한 정보는 를 사용하여 시작 템플릿을 생성하고 관리하는 예제 AWS CLI을 참조하세요.

2. Auto Scaling 그룹의 기존 혼합 인스턴스 정책을 보려면 describe-auto-scaling-groups명령을 실행합니다. 인스턴스 새로 고침을 시작할 때 다음 단계에서 이 정보가 필요합니다.

   다음 예 명령은 *my-asg*라는 Auto Scaling 그룹에 대해 구성된 혼합 인스턴스 정책을 반환합니다.

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

   이 작업이 성공하면 다음과 비슷한 출력이 반환됩니다.

   ```
   {
     "AutoScalingGroups":[
   ```

```
    {
      "AutoScalingGroupName":"my-asg",
      "AutoScalingGroupARN":"arn",
      "MixedInstancesPolicy":{
        "LaunchTemplate":{
          "LaunchTemplateSpecification":{
            "LaunchTemplateId":"lt-073693ed27example",
            "LaunchTemplateName":"my-old-template",
            "Version":"$Default"
          },
          "Overrides":[
            {
              "InstanceType":"c5.large"
            },
            {
              "InstanceType":"c5a.large"
            },
            {
              "InstanceType":"m5.large"
            },
            {
              "InstanceType":"m5a.large"
            }
          ]
        },
        "InstancesDistribution":{
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"price-capacity-optimized"
        }
      },
      "MinSize":1,
      "MaxSize":5,
      "DesiredCapacity":4,
      ...
    }
  ]
}
```

3. start-instance-refresh명령을 사용하여 인스턴스 교체 워크플로를 시작하고 ID를 *lt-04d5cc9b88example*사용하여 새 시작 템플릿을 적용합니다. 시작 템플릿은 새 템플릿이므로 버전이 하나만 있습니다. 즉, 시작 템플릿의 버전 1이 이 인스턴스 새로 고침의 대상입니다. 인

스턴스 새로 고침 중 스케일아웃 이벤트가 발생하고 Amazon EC2 Auto Scaling이 이 시작 템플릿의 버전 1을 사용하여 새 인스턴스를 프로비저닝하는 경우, 인스턴스는 교체되지 않습니다. 작업이 성공적으로 완료되면 업데이트된 혼합 인스턴스 정책이 Auto Scaling 그룹에 성공적으로 적용됩니다.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json`의 콘텐츠.

```
{
  "AutoScalingGroupName":"my-asg",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate":{
        "LaunchTemplateSpecification":{
          "LaunchTemplateId":"lt-04d5cc9b88example",
          "Version":"$Default"
        },
        "Overrides":[
          {
            "InstanceType":"c5.large"
          },
          {
            "InstanceType":"c5a.large"
          },
          {
            "InstanceType":"m5.large"
          },
          {
            "InstanceType":"m5a.large"
          }
        ]
      },
      "InstancesDistribution":{
        "OnDemandAllocationStrategy":"prioritized",
        "OnDemandBaseCapacity":1,
        "OnDemandPercentageAboveBaseCapacity":50,
        "SpotAllocationStrategy":"price-capacity-optimized"
        }
      }
    }
  },
```

```
  "Preferences":{
    "SkipMatching":true
  }
}
```

이 작업이 성공하면 다음과 비슷한 출력이 반환됩니다.

```
{
  "InstanceRefreshId": "08b91cf7-8fa6-48af-b6a6-d227f40f1b9b"
}
```

다음 절차에서는 시작 템플릿을 변경하지 않고 업데이트된 인스턴스 유형 세트를 제공합니다.

업데이트된 인스턴스 유형 세트와 함께 매칭 건너뛰기 사용

1. Auto Scaling 그룹의 기존 혼합 인스턴스 정책을 보려면 describe-auto-scaling-groups명령을 실행합니다. 인스턴스 새로 고침을 시작할 때 다음 단계에서 이 정보가 필요합니다.

   다음 예 명령은 *my-asg*라는 Auto Scaling 그룹에 대해 구성된 혼합 인스턴스 정책을 반환합니다.

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

   이 작업이 성공하면 다음과 비슷한 출력이 반환됩니다.

   ```
   {
     "AutoScalingGroups":[
       {
         "AutoScalingGroupName":"my-asg",
         "AutoScalingGroupARN":"arn",
         "MixedInstancesPolicy":{
           "LaunchTemplate":{
             "LaunchTemplateSpecification":{
               "LaunchTemplateId":"lt-073693ed27example",
               "LaunchTemplateName":"my-template-for-auto-scaling",
               "Version":"$Default"
             },
             "Overrides":[
               {
                 "InstanceType":"c5.large"
               },
               {
   ```

```
              "InstanceType":"c5a.large"
            },
            {
              "InstanceType":"m5.large"
            },
            {
              "InstanceType":"m5a.large"
            }
          ]
        },
        "InstancesDistribution":{
          "OnDemandAllocationStrategy":"prioritized",
          "OnDemandBaseCapacity":1,
          "OnDemandPercentageAboveBaseCapacity":50,
          "SpotAllocationStrategy":"price-capacity-optimized"
        }
      },
      "MinSize":1,
      "MaxSize":5,
      "DesiredCapacity":4,
      ...
    }
  ]
}
```

2. start-instance-refresh명령을 사용하여 인스턴스 교체 워크플로를 시작하고 업데이트를 적용합니다. 특정 인스턴스 유형을 사용하는 인스턴스를 교체하려는 경우, 원하는 구성에서 원하는 인스턴스 유형만 포함하는 혼합 인스턴스 정책을 지정해야 합니다. 새 인스턴스 유형을 대신 추가할지 여부를 선택할 수 있습니다.

   다음 예 명령은 원하지 않는 인스턴스 유형 *m5a.large*를 제외한 인스턴스 새로 고침을 시작합니다. 그룹의 인스턴스 유형이 나머지 세 가지 인스턴스 유형 중 하나와 일치하지 않으면 인스턴스가 교체됩니다. (인스턴스 새로 고침이 새 인스턴스를 프로비저닝할 인스턴스 유형을 선택하지 않고, 대신 할당 전략이 선택합니다.) 작업이 성공적으로 완료되면 업데이트된 혼합 인스턴스 정책이 Auto Scaling 그룹에 성공적으로 적용됩니다.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

   `config.json`의 콘텐츠

```
{
```

```
  "AutoScalingGroupName":"my-asg",
  "DesiredConfiguration":{
    "MixedInstancesPolicy":{
      "LaunchTemplate":{
        "LaunchTemplateSpecification":{
          "LaunchTemplateId":"lt-073693ed27example",
          "Version":"$Default"
        },
        "Overrides":[
          {
            "InstanceType":"c5.large"
          },
          {
            "InstanceType":"c5a.large"
          },
          {
            "InstanceType":"m5.large"
          }
        ]
      },
      "InstancesDistribution":{
        "OnDemandAllocationStrategy":"prioritized",
        "OnDemandBaseCapacity":1,
        "OnDemandPercentageAboveBaseCapacity":50,
        "SpotAllocationStrategy":"price-capacity-optimized"
        }
      }
    }
  },
  "Preferences":{
    "SkipMatching":true
  }
}
```

인스턴스 새로 고침에 체크포인트 추가

인스턴스 새로 고침을 사용하는 경우, 작업을 진행하면서 인스턴스에 대한 검증을 수행할 수 있도록 단계별로 인스턴스를 교체하도록 선택할 수 있습니다. 단계별 교체를 수행하려면 인스턴스 새로 고침이 일시 중지되는 시점인 체크포인트를 추가합니다. 체크포인트를 사용하면 Auto Scaling 그룹을 업데이트하는 방법을 더 강력하게 제어할 수 있습니다. 체크포인트는 애플리케이션이 안정적이고 예측 가능한 방식으로 작동하는지 확인하는 데 유용합니다.

내용

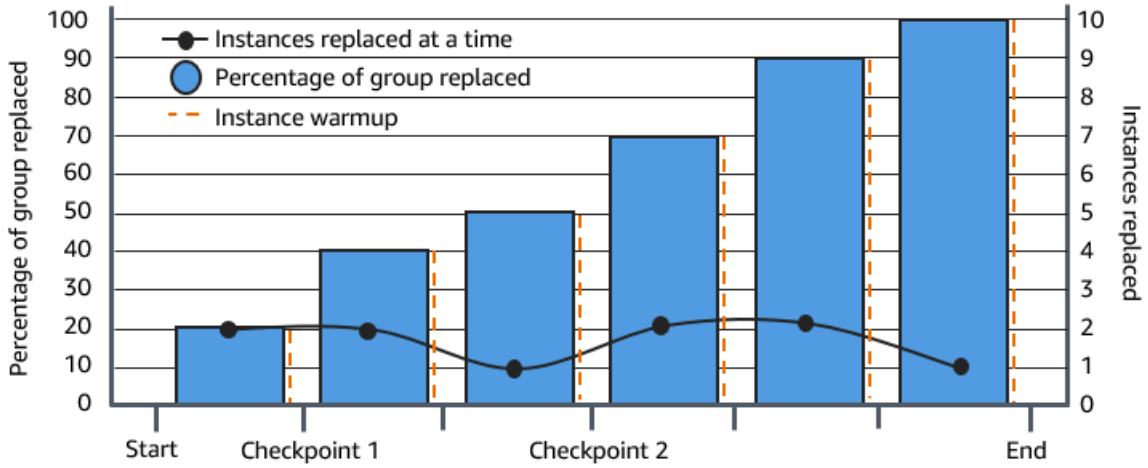
- 작동 방식
- 고려 사항
- 또는 를 사용하여 체크포인트를 활성화합니다. AWS Management ConsoleAWS CLI

작동 방식

인스턴스 새로 고침을 시작할 때 Auto Scaling 그룹에 있는 총 인스턴스 수의 백분율로 체크포인트를 지정합니다. 이 체크포인트는 Auto Scaling 그룹에서 새 인스턴스가 되어야 체크포인트에 도달한 것으로 간주되는 인스턴스의 최소 비율을 나타냅니다. 예를 들어 체크포인트가 다음과 `[20, 50, 100]` 같으면 새 인스턴스의 20% 가 첫 번째 체크포인트에 도달하고, 새 인스턴스가 50% 이면 두 번째 체크포인트에 도달하고, 모든 인스턴스가 새로 추가되면 최종 체크포인트에 도달합니다.

Amazon EC2 Auto Scaling은 그룹의 최소 정상 비율을 유지하면서 지정된 체크포인트 비율을 준수하도록 인스턴스 교체를 진행합니다. 체크포인트 백분율에 도달하기 위해 Amazon EC2 Auto Scaling은 때때로 최소 건전 백분율에서 허용하는 것보다 더 적게 교체하지만 절대 그보다 많이 교체하지는 않습니다.

인스턴스가 10개인 다음 Auto Scaling 그룹을 예로 들어 보겠습니다. 체크포인트 백분율은 `[20,50,100]`, 최소 건전 백분율은 80%, 최대 건전 백분율은 100%입니다. 최소 건전 백분율을 유지하기 위해 한 번에 2개의 인스턴스만 교체됩니다. 다음 다이어그램은 체크포인트에 도달하기 전에 인스턴스를 교체하는 프로세스를 요약한 것입니다.



위 예시에서는 새로 시작하는 각 인스턴스마다 인스턴스 준비 기간이 있습니다. 인스턴스를 대기 상태로 전환한 다음 인스턴스가 시작되거나 해지될 때 사용자 지정 작업을 수행하는 수명 주기 후크가 있을 수도 있습니다.

Amazon EC2 Auto Scaling은 100% 완료된 체크포인트를 제외하고 각 체크포인트에 대해 이벤트를 내보냅니다. Amazon SNS 등의 대상으로 이벤트를 전송하는 EventBridge 규칙을 추가할 수 있습니다. 이렇게 하면 필요한 검증을 실행할 수 있을 때 알림이 옵니다. 자세한 정보는 인스턴스 새로 고침 이벤트에 대한 EventBridge 규칙 생성을 참조하세요.

고려 사항

체크포인트를 사용할 때에는 다음 사항을 고려하세요.

- 체크포인트는 백분율을 기준으로 하기 때문에 교체할 인스턴스 수는 그룹 크기에 따라 달라집니다. 스케일 아웃 활동이 발생하고 그룹 규모가 커지면 진행 중인 작업이 다시 체크포인트에 도달할 수 있습니다. 이 경우, Amazon EC2 Auto Scaling은 다른 알림을 전송하고 계속하기 전에 체크포인트 간 대기 시간을 반복합니다.
- 특정 상황에서 체크포인트를 건너뛸 수 있습니다. 예컨대, Auto Scaling 그룹에 두 개의 인스턴스가 있고 체크포인트 백분율이 `[10,40,100]`이라고 가정해 보겠습니다. 첫 번째 인스턴스를 교체한 후 Amazon EC2 Auto Scaling은 그룹의 50%가 교체된 것으로 계산합니다. 50%가 처음 두 개의 체크포인트보다 높기 때문에 첫 번째 체크포인트(`10`)를 건너뛰고 두 번째 체크포인트(`40`)에 대한 알림을 전송합니다.
- 작업을 취소하면 추가 교체가 중지됩니다. 작업을 취소하거나 마지막 체크포인트에 도달하기 전에 작업이 실패하면 이미 교체된 인스턴스는 이전 구성으로 롤백되지 않습니다.
- 부분 새로 고침의 경우, 작업을 다시 실행하면 Amazon EC2 Auto Scaling이 마지막 체크포인트에서 다시 시작하지 않고 이전 인스턴스가 교체되는 경우에만 중지하지도 않습니다. 그러나 교체할 대상으로 새 인스턴스에 앞서 이전 인스턴스를 지정합니다.
- 체크포인트 비율이 그룹 내 인스턴스 수에 비해 너무 낮으면 실제 완료율이 해당 체크포인트의 백분율보다 높을 수 있습니다. 예컨대, 체크포인트의 백분율이 20%이고 그룹에 4개의 인스턴스가 있다고 가정해 보겠습니다. Amazon EC2 Auto Scaling이 4개의 인스턴스 중 하나를 교체하는 경우, 실제 교체된 백분율(25%)이 체크포인트의 백분율(20%)보다 높습니다.
- 체크포인트에 도달한 후 표시된 전체 완료율은 인스턴스 워밍업이 완료될 때까지 업데이트되지 않습니다. 예를 들어 체크포인트 백분율은 15분이고 `[20,50]` 최소 정상 백분율은 80% 입니다. Auto Scaling 그룹에는 10개의 인스턴스가 있으며 다음과 같이 인스턴스를 대체합니다.
  - 0:00: 두 개의 이전 인스턴스가 새 인스턴스로 교체됩니다.
  - 0:10: 두 개의 새 인스턴스가 워밍업을 완료합니다.
  - 0:25: 두 개의 이전 인스턴스가 새 인스턴스로 교체됩니다. (최소 건전 백분율을 유지하기 위해 인스턴스가 두 개만 교체됩니다.)
  - 0:35: 두 개의 새 인스턴스가 워밍업을 완료합니다.
  - 0:35: 한 개의 이전 인스턴스가 새 인스턴스로 교체됩니다.

- 0:45: 한 개의 새 인스턴스가 워밍업을 완료합니다.

0:35에서 새 인스턴스 시작 작업이 중지됩니다. 새 인스턴스의 워밍업이 완료되지 않았기 때문에 완료율은 아직 완료된 교체 수(50%)를 정확하게 반영하지 않습니다. 새 인스턴스가 0:45 에 워밍업 기간을 완료한 후 완료율은 50% 로 표시됩니다.

또는 를 사용하여 체크포인트를 활성화합니다. AWS Management ConsoleAWS CLI

AWS Management Console OR를 사용하여 체크포인트를 AWS CLI 활성화할 수 있습니다.

체크포인트 활성화(콘솔)

인스턴스 새로 고침을 시작하기 전에 체크포인트를 활성화하여 증분 또는 단계적 접근 방식을 사용하여 인스턴스를 교체할 수 있습니다. 이렇게 하면 검증을 위한 추가 시간이 제공됩니다.

체크포인트를 사용하는 인스턴스 새로 고침을 시작하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 아래쪽에 분할 창이 열립니다.
3. 인스턴스 새로 고침(Active instance refresh) 탭의 활성 인스턴스 새로 고침(Active Instance refreshes)에서 인스턴스 새로 고침 시작(Start instance refresh)을 선택합니다.
4. 인스턴스 새로 고침 시작(Start instance refresh) 페이지에 최소 건전 백분율(Minimum healthy percentage) 및 인스턴스 워밍업(Instance warmup)을 입력합니다.
5. 체크포인트 활성화(Enable checkpoints) 확인란을 선택합니다.

   그러면 첫 번째 체크포인트에 대한 백분율 임계값을 정의할 수 있는 상자가 표시됩니다.
6. 그룹의 ____%를 새로 고칠 때까지 계속 진행(Proceed until ____ % of the group is refreshed)에 숫자(1~100)를 입력합니다. 이렇게 하면 첫 번째 체크포인트의 백분율이 설정됩니다.
7. 다른 체크포인트를 추가하려면Add checkpoint(체크포인트 추가)를 클릭하고 다음 체크포인트의 백분율을 정의합니다.
8. 체크포인트에 도달한 후 Amazon EC2 Auto Scaling이 대기하는 시간을 지정하려면 체크포인트 간에 **1 hour** 대기의 필드를 업데이트합니다. 시간 단위는 시, 분 또는 초가 될 수 있습니다.
9. 인스턴스 새로 고침 선택을 마치면 인스턴스 새로 고침 시작을 선택합니다.

체크포인트 활성화(AWS CLI)

를 사용하여 체크포인트를 활성화한 상태로 인스턴스 새로 고침을 AWS CLI시작하려면 다음 파라미터를 정의하는 구성 파일이 필요합니다.

- `CheckpointPercentages`: 교체할 인스턴스의 백분율에 대한 임계값을 지정합니다. 이러한 임계값은 체크포인트를 제공합니다. 교체 및 워밍업되는 인스턴스의 백분율이 지정된 임계값 중 하나에 도달하면 작업이 지정된 기간에 대기합니다. `CheckpointDelay`에서 대기할 시간(초)을 지정합니다. 지정된 기간이 경과하면 인스턴스 새로 고침은 다음 체크포인트에 도달할 때까지 계속됩니다(해당하는 경우).
- `CheckpointDelay`: 계속 진행하기 전 체크포인트에 도달한 후 대기하는 시간(초)을 지정합니다. 검증을 수행할 수 있는 충분한 시간을 제공하는 기간을 선택합니다.

`CheckpointPercentages` 어레이에 표시된 마지막 값은 성공적으로 교체해야 하는 Auto Scaling 그룹의 백분율을 설명합니다. 이 백분율이 성공적으로 교체되고 각 인스턴스가 초기화를 완료한 것으로 간주되면 작업이 `Successful`(으)로 전환됩니다.

체크포인트를 여러 개 생성하려면

체크포인트를 여러 개 만들려면 다음 예제 start-instance-refresh명령을 사용합니다. 이 예에서는 처음에 Auto Scaling 그룹의 1%를 새로 고치는 인스턴스 새로 고침을 구성합니다. 인스턴스 새로 고침은 10분을 기다린 후 다음 19퍼센트를 새로 고치고 10분을 더 기다립니다. 마지막으로 그룹의 나머지 인스턴스를 새로 고치고 작업을 해지합니다.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json`의 콘텐츠:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20,100],
      "CheckpointDelay": 600
    }
}
```

단일 체크포인트를 생성하려면

단일 체크포인트를 만들려면 다음 예제 start-instance-refresh명령을 사용합니다. 이 예에서는 처음에 Auto Scaling 그룹의 20%를 새로 고치는 인스턴스 새로 고침을 구성합니다. 인스터스 새로 고침은 10분을 기다린 후 그룹의 나머지 인스턴스를 새로 고치고 작업을 해지합니다.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json`의 콘텐츠:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [20,100],
      "CheckpointDelay": 600
    }
}
```

Auto Scaling 그룹을 부분적으로 새로 고치려면

Auto Scaling 그룹의 일부만 교체한 다음 완전히 중지하려면 다음 예제 start-instance-refresh명령을 사용하십시오. 이 예에서는 처음에 Auto Scaling 그룹의 1%를 새로 고치는 인스턴스 새로 고침을 구성합니다. 인스턴스 새로 고침은 10분을 기다린 후 다음 19%를 새로 고치고 작업을 해지합니다.

```
aws autoscaling start-instance-refresh --cli-input-json file://config.json
```

`config.json`의 콘텐츠:

```
{
    "AutoScalingGroupName": "my-asg",
    "Preferences": {
      "InstanceWarmup": 60,
      "MinHealthyPercentage": 80,
      "CheckpointPercentages": [1,20],
      "CheckpointDelay": 600
    }
}
```

최대 인스턴스 수명을 기준으로 Auto Scaling 인스턴스 교체

최대 인스턴스 수명은 인스턴스가 해지되고 교체되기 전에 서비스를 제공할 수 있는 최대 시간(초)을 지정합니다. 일반적인 사용 사례 중 하나로 내부 보안 정책 또는 외부 규정 준수 제어로 인해 일정에 따라 인스턴스를 교체할 것이 요구되는 경우가 있습니다.

최소 86,400초(1일)의 값을 지정해야 합니다. 이전에 설정한 값을 해제하려면 새 값 0을 지정합니다. 이 설정은 Auto Scaling 그룹의 현재 인스턴스와 향후 인스턴스에 모두 적용됩니다.

내용

- 고려 사항
- 최대 인스턴스 수명 설정
- 제한 사항

고려 사항

이 기능을 사용할 때 고려할 사항은 다음과 같습니다.

- 이전 인스턴스를 교체하고 새 인스턴스를 시작할 때마다 새 인스턴스는 현재 Auto Scaling 그룹과 연결된 시작 템플릿 또는 시작 구성을 사용합니다. 시작 템플릿 또는 시작 구성에서 다른 버전의 애플리케이션의 Amazon Machine Image (AMI) ID를 지정하는 경우 이 버전의 애플리케이션이 자동으로 배포됩니다.
- 최대 인스턴스 수명을 너무 낮게 설정하면 인스턴스가 원하는 것보다 빨리 교체될 수 있습니다. Amazon EC2 Auto Scaling은 일반적으로 한 번에 하나씩 인스턴스를 교체하며, 교체할 때마다 일시 중지합니다. 그러나 지정된 최대 인스턴스 수명으로도 각 인스턴스를 개별적으로 교체할 시간이 충분하지 않은 경우 Amazon EC2 Auto Scaling은 한 번에 두 개 이상의 인스턴스를 교체해야 합니다. Auto Scaling 그룹의 현재 용량의 최대 10%까지 여러 인스턴스를 한 번에 교체할 수 있습니다. 한 번에 너무 많은 인스턴스를 교체하지 않으려면 최대 인스턴스 수명을 더 길게 설정하거나 인스턴스 확장 보호를 사용하여 개별 인스턴스가 종료되는 것을 일시적으로 방지하십시오. 자세한 정보는 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다.을 참조하세요.
- 기본적으로, Amazon EC2 Auto Scaling은 인스턴스를 해지하기 위해 새 규모 조정 활동을 생성한 다음 해당 인스턴스를 해지합니다. 인스턴스를 해지하는 동안 다른 크기 조정 활동이 새 인스턴스를 시작합니다. 인스턴스 유지 관리 정책을 사용하여 해지하기 전에 실행되도록 이 동작을 변경할 수 있습니다. 자세한 정보는 인스턴스 유지 관리 정책을 참조하세요.

최대 인스턴스 수명 설정

콘솔에서 Auto Scaling 그룹을 생성할 때에는 최대 인스턴스 수명을 설정할 수 없습니다. 그러나 그룹이 생성된 후에는 최대 인스턴스 수명을 편집하여 설정할 수 있습니다.

그룹의 최대 인스턴스 수명을 설정하려면(콘솔)

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 아래쪽에 분할 창이 열리고 선택한 그룹에 대한 정보가 표시됩니다.
3. 세부 정보(Details) 탭에서 고급 구성(Advanced configurations), 편집(Edit)을 선택합니다.
4. 최대 인스턴스 수명(Maximum instance lifetime)에 인스턴스가 서비스될 수 있는 최대 시간(초)을 입력합니다.
5. 업데이트를 선택합니다.

활동(Activity) 탭의 활동 기록(Activity history)에서 그룹의 전체 인스턴스 교체 기록을 볼 수 있습니다.

그룹의 최대 인스턴스 수명을 설정하려면(AWS CLI)

를 사용하여 신규 또는 기존 Auto Scaling 그룹의 최대 인스턴스 수명을 설정할 수도 있습니다. AWS CLI

새 Auto Scaling 그룹의 경우 create-auto-scaling-group명령을 사용하십시오.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

다음은 `2592000`초(30일)의 최대 인스턴스 수명을 보여주는 예 `config.json` 파일입니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateName": "my-launch-template",
        "Version": "$Default"
    },
    "MinSize": 1,
    "MaxSize": 5,
    "MaxInstanceLifetime": 2592000,
```

```
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": []
}
```

기존 Auto Scaling 그룹의 경우 update-auto-scaling-group명령을 사용하십시오.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-existing-asg --
max-instance-lifetime 2592000
```

Auto Scaling 그룹의 최대 인스턴스 수명 확인

describe-auto-scaling-groups 명령을 사용합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

제한 사항

- 모든 인스턴스에 대해 최대 수명이 정확하지 않을 수 있음: 최대 기간이 끝날 때만 인스턴스가 교체된다는 보장이 없습니다. 경우에 따라, 최대 인스턴스 수명 파라미터가 업데이트된 후 Amazon EC2 Auto Scaling이 즉시 인스턴스 교체를 시작해야 할 수 있습니다. 이 동작의 이유는 모든 인스턴스를 동시에 교체하는 상황을 방지하는 것입니다.
- 인스턴스 스케일 인 보호 적용: Amazon EC2 Auto Scaling은 종료할 수 있는 인스턴스를 제어할 수 있도록 인스턴스 스케일 인 보호 기능을 제공합니다. 인스턴스에서 이 보호 기능을 활성화하면 Amazon EC2 Auto Scaling은 인스턴스가 최대 인스턴스 수명에 도달했더라도 인스턴스를 종료하지 않습니다.
- 시작 전에 인스턴스가 해지됨: Amazon EC2 Auto Scaling은 기본적으로 인스턴스를 해지한 다음에 새 인스턴스를 시작시키기 때문에 Auto Scaling 그룹에 인스턴스가 하나만 있는 경우, 최대 인스턴스 수명 기능으로 인해 중단이 발생할 수 있습니다. 이 동작을 해지하기 전에 실행되도록 변경하려면 인스턴스 유지 관리 정책을 참조하세요.

스케일링을 통해 애플리케이션의 컴퓨팅 용량을 늘리거나 줄입니다.

조정은 애플리케이션의 컴퓨팅 용량을 늘리거나 줄이는 기능입니다. 확장은 Auto Scaling 그룹에 Amazon EC2 인스턴스를 시작하거나 종료하도록 지시하는 이벤트 또는 조정 작업으로 시작됩니다.

Amazon EC2 Auto Scaling은 애플리케이션 요구 사항에 가장 적합하도록 크기 조정을 조정할 수 있는 다양한 방법을 제공합니다. 그러므로 애플리케이션을 충분히 이해하는 것이 중요합니다. 다음 사항에 유의하세요.

- Amazon EC2 Auto Scaling은 애플리케이션 아키텍처에서 어떤 역할을 해야 합니까? 자동 조정을 주로 용량을 늘리거나 줄이는 수단으로 생각하는 것이 일반적이지만, 이는 안정적인 서버 수를 유지하려는 경우에도 유용합니다.
- 나에게 중요한 비용 제약 조건은 무엇인가요? Amazon EC2 Auto Scaling은 EC2 인스턴스를 사용하기 때문에 사용한 리소스에 대해서만 비용을 지불하면 됩니다. 비용 제약 조건을 알면 애플리케이션의 스케일 아웃 시기와 규모를 결정하는 데 도움이 될 수 있습니다.
- 애플리케이션에 어떤 지표가 중요한가요? Amazon은 Auto Scaling 그룹에서 사용할 수 있는 다양한 지표를 CloudWatch 지원합니다.

내용

- 스케일링 방법 선택
- Auto Scaling 그룹에 대한 스케일링 제한 설정
- Auto Scaling 그룹의 기본 인스턴스 워밍업 설정
- Amazon EC2 Auto Scaling을 위한 수동 스케일링
- Amazon EC2 Auto Scaling을 위한 스케일링 예약
- Amazon EC2 Auto Scaling을 위한 동적 스케일링
- Amazon EC2 Auto Scaling을 위한 예측 스케일링
- 축소 시 해지할 Auto Scaling 인스턴스 제어
- Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개

스케일링 방법 선택

Amazon EC2 Auto Scaling은 Auto Scaling 그룹을 확장할 수 있는 여러 가지 방법을 제공합니다.

일정한 수의 인스턴스 유지

Auto Scaling 그룹의 기본값은 연결된 스케일링 정책이나 예약된 작업이 없어 고정된 크기를 유지하도록 하는 것입니다. Auto Scaling 그룹을 생성하면 그 그룹은 원하는 용량을 충족하도록 충분한 인스턴스를 시작하여 시작합니다. 그룹에 연결된 스케일링 조건이 없는 경우, 인스턴스가 비건전 상태가 되더라도 원하는 용량을 계속 유지합니다. Amazon EC2 Auto Scaling은 Auto Scaling 그룹 내 각 인스턴스의 상태를 모니터링합니다. 인스턴스가 비건전 상태가 된 것을 발견하면 새 인스턴스로 교체합니다. 이 프로세스에 대한 자세한 설명은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인에서 확인할 수 있습니다.

수동 조정

수동 스케일링은 Auto Scaling 그룹을 스케일링하는 가장 기본적인 방법입니다. Auto Scaling 그룹의 원하는 용량을 업데이트하거나 Auto Scaling 그룹에서 인스턴스를 종료할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 수동 스케일링 단원을 참조하십시오.

일정에 근거하여 조정

일정에 따른 크기 조정이란 날짜 및 시간의 함수에 따라 조정 작업이 자동으로 수행된다는 것을 의미합니다. 그룹의 인스턴스 수를 늘려야 할지 또는 줄여야 할지 정확히 아는 경우에 유용합니다. 그 이유는 예측 가능한 일정에 따라 수요가 증가하기 때문입니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 스케일링 예약 단원을 참조하십시오.

수요에 따라 동적으로 규모를 조정할 수 있습니다.

동적 조정을 사용해 리소스를 조정하는 더 향상된 방법을 사용하면 수요 변화에 맞춰 Auto Scaling 그룹의 크기를 동적으로 조정하는 조정 정책을 정의할 수 있습니다. 예를 들어, 현재 두 개의 인스턴스에서 실행되는 웹 애플리케이션이 있고 애플리케이션의 부하가 변경될 때 Auto Scaling 그룹의 CPU 사용률을 약 50% 로 유지하려고 한다고 가정해 보겠습니다. 이 방법은 트래픽이 언제 변경될지 모르는 상황에서 트래픽 변화에 따라 규모를 조정하는 데 유용합니다. 대신 응답하도록 스케일링 정책을 구성할 수 있습니다. 트래픽 변화에 따라 규모를 조정하는 데 사용할 수 있는 여러 정책 유형 (또는 이들의 조합) 이 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 동적 스케일링 단원을 참조하십시오.

사전 예방적 규모 조정

또한 예측적 확장과 동적 규모 조정 (각각 사전 및 사후 접근 방식) 을 결합하여 용량을 더 빠르게 확장할 수 있습니다. EC2 예측 조정을 사용하면 트래픽 흐름의 일별 및 주별 패턴에 앞서 Auto Scaling 그룹의 EC2 인스턴스 수를 늘릴 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 예측 스케일링 단원을 참조하십시오.

Auto Scaling 그룹에 대한 스케일링 제한 설정

스케일링 한도는 Auto Scaling 그룹의 원하는 최소 및 최대 그룹 크기를 나타냅니다. 최소 및 최대 크기에 대해 별도로 제한을 설정합니다.

그룹의 원하는 용량은 최소 및 최대 크기 제한 범위 내의 수치로 조정할 수 있습니다. 원하는 용량은 최소 그룹 크기보다 크거나 같아야 하며 최대 그룹 크기보다 작거나 같아야 합니다.

- Desired capacity(원하는 용량): 생성 시의 초기 Auto Scaling 그룹 용량을 나타냅니다. Auto Scaling 그룹은 원하는 용량을 유지하려고 시도합니다. 이는 원하는 용량을 제공하도록 지정된 수의 인스턴스를 시작하는 것으로 시작되며, Auto Scaling 그룹에 연결된 조정 정책이나 예약된 작업이 없는 경우, 이 인스턴스 수를 유지합니다.
- 최소 용량(Minimum capacity): 최소 그룹 크기를 나타냅니다. 조정 정책이 설정되면 정책은 그룹이 원하는 용량을 최소 용량보다 작게 줄일 수 없습니다.
- 최대 용량(Maximum capacity): 최대 그룹 크기를 나타냅니다. 조정 정책이 설정되면 정책은 그룹이 원하는 용량을 최대 용량보다 크게 늘일 수 없습니다.

최소 및 최대 크기 제한은 다음 시나리오에서도 적용됩니다.

- 원하는 용량을 업데이트하여 Auto Scaling 그룹을 수동으로 조정하는 경우
- 원하는 용량을 업데이트하는 예약된 작업이 실행되는 경우 그룹에 대해 새 최소 및 최대 크기 제한을 지정하지 않고 예약된 작업을 실행하면, 그룹의 현재 최소 및 최대 크기 제한이 적용됩니다.

Auto Scaling 그룹은 항상 원하는 용량을 유지하려고 시도합니다. 인스턴스가 예기치 않게 해지되는 경우(예: 스팟 인스턴스 중단, 건전성 체크 불합격 또는 사용자 작업으로 인해) 이 그룹은 원하는 용량을 유지하기 위해 새 인스턴스를 자동으로 시작합니다.

콘솔에서 이러한 설정을 관리하려면

1. 에서 Amazon EC2 콘솔을 엽니다 https://console.aws.amazon.com/ec2/.
2. 탐색 창의 Auto Scaling에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.
3. Auto Scaling 그룹 페이지에서 Auto Scaling 그룹 옆의 확인란을 선택합니다.

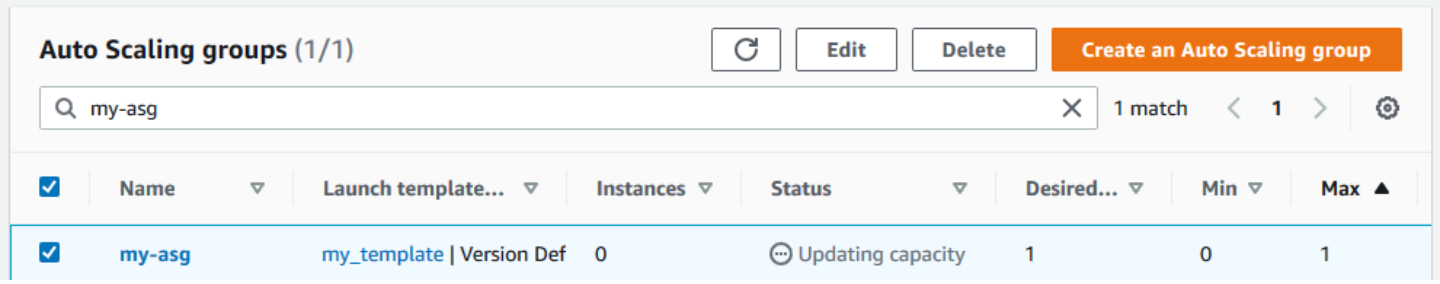
   페이지 하단에 분할 창이 열립니다.
4. 아래 창의 세부 정보 탭에서 그룹의 원하는 용량, 최소 및 최대 용량에 대한 현재 설정을 확인하거나 변경합니다. 자세한 내용은 기존 Auto Scaling 그룹의 원하는 용량 변경 단원을 참조하십시오.

세부 정보 창 위에는 Auto Scaling 그룹의 현재 인스턴스 수, 원하는 용량, 최소 및 최대 용량, 그리고 상태 열 같은 정보가 나와 있습니다. Auto Scaling 그룹에서 인스턴스 가중치를 사용하는 경우 원하는 용량에 포함된 용량 단위 수를 확인할 수도 있습니다.

목록에서 열을 추가하거나 제거하려면 페이지 상단의 설정 아이콘을 선택합니다. 그런 다음 Auto Scaling groups attributes(Auto Scaling 그룹 속성)에서 각 열을 켜거나 끄고 Confirm(확인)을 선택합니다.

변경한 후 Auto Scaling 그룹의 크기 확인

인스턴스(Instances) 열에는 현재 실행 중인 인스턴스 수가 표시됩니다. 인스턴스가 시작되거나 해지되는 동안에는 다음 이미지에 표시된 것처럼 상태(Status) 열에 용량 업데이트 중(Updating capacity) 상태가 표시됩니다.



몇 분 동안 기다렸다가 보기를 새로 고쳐 최신 상태를 확인합니다. 크기 조정 활동이 완료되면 인스턴스(Instances) 열에 업데이트된 값이 표시됩니다.

Instance management(인스턴스 관리) 탭의 Instances(인스턴스)에서 현재 실행 중인 인스턴스의 개수와 상태를 확인할 수 있습니다.

Auto Scaling 그룹의 기본 인스턴스 워밍업 설정

CloudWatch Auto Scaling 인스턴스 전반에서 네트워크 CPU I/O와 같은 사용 데이터를 수집하고 집계합니다. 이러한 지표를 사용하여 선택한 지표의 값이 증가 및 감소함에 따라 Auto Scaling 그룹의 인스턴스 수를 조정하는 조정 정책을 생성합니다.

인스턴스가 `InService` 상태에 도달한 후 집계된 지표에 사용량 데이터를 제공하기 전에 기다리는 시간을 지정할 수 있습니다. 이렇게 지정된 시간을 기본 인스턴스 워밍업이라고 합니다. 이렇게 하면 아직 애플리케이션 트래픽을 처리하지 않고 일시적으로 컴퓨팅 리소스 사용량이 높을 수 있는 개별 인스턴스에 대한 지표의 영향을 받아 동적 규모 조정이 영향을 받지 않습니다.

대상 추적 및 단계 조정 정책의 성능을 최적화하려면 기본 인스턴스 웜업을 활성화하고 구성하는 것이 좋습니다. 기본적으로 활성화되거나 구성되어 있지 않습니다.

기본 인스턴스 웜업을 활성화할 때, Auto Scaling 그룹이 인스턴스 유지 관리 정책을 사용하도록 설정되거나 인스턴스 새로 고침을 사용하여 인스턴스를 교체하는 경우 초기화가 완료되기 전에 인스턴스가 최소 정상 백분율에 포함되는 것을 방지할 수 있다는 점을 염두에 두십시오.

내용

- 스케일링 수행 고려 사항
- 기본 인스턴스 워밍업 시간을 선택합니다.
- 그룹에 대한 기본 인스턴스 워밍업 활성화
- 그룹의 기본 인스턴스 워밍업 시간을 확인하십시오.
- 이전에 설정한 인스턴스 워밍업 시간의 조정 정책을 찾아보세요.
- 스케일링 정책에 대해 이전에 설정한 인스턴스 워밍업을 지웁니다.

스케일링 수행 고려 사항

대부분의 애플리케이션에서는 기능별로 워밍업 시간을 다르게 설정하는 대신 모든 기능에 적용되는 기본 인스턴스 워밍업 시간을 하나로 설정하는 것이 좋습니다. 예를 들어 기본 인스턴스 웜업을 설정하지 않은 경우 인스턴스 새로 고침 기능은 상태 확인 유예 기간을 기본 워밍업 시간으로 사용합니다. 대상 추적 및 단계 조정 정책이 있는 경우 기본 휴지 시간에 설정된 값을 기본 워밍업 시간으로 사용합니다. 예측 조정 정책이 있는 경우에는 기본 워밍업 시간이 없습니다.

인스턴스가 워밍업되는 동안에는 워밍업하지 않는 인스턴스의 지표 값이 정책의 경보 상한 임계값 (또는 대상 추적 조정 정책의 목표 사용률) 보다 큰 경우에만 동적 조정 정책이 축소됩니다. 수요가 감소하면 동적 크기 조정은 애플리케이션의 가용성을 보호하기 위해 더욱 보수적으로 조정됩니다. 이렇게 하면 새 인스턴스의 워밍업이 완료될 때까지 동적 조정 활동의 규모가 차단됩니다.

Amazon EC2 Auto Scaling은 규모를 축소하는 동안 그룹에 추가할 인스턴스 수를 결정할 때 워밍업 중인 인스턴스를 그룹 용량의 일부로 간주합니다. 따라서 비슷한 용량을 추가해야 하는 경보 위반이 여러 번 발생하면 단일 조정 작업이 발생합니다. 과도한 규모 확대를 하지 않으면서 지속적으로 규모를 확장하려는 의도입니다.

기본 인스턴스 워밍업이 활성화되지 않은 경우, 메트릭을 전송하여 CloudWatch 현재 용량으로 계산하기 전에 인스턴스가 대기하는 시간은 인스턴스마다 달라집니다. 따라서 조정 정책이 실제 발생하는 워크로드와 비교할 때 예상치 못한 성능을 보일 가능성이 있습니다.

반복되는 on-and-off 워크로드 패턴을 가진 애플리케이션을 예로 들어 보겠습니다. 예측 스케일링 정책은 인스턴스 수를 늘릴지 여부에 대한 반복적인 결정을 내리는 데 사용됩니다. 예측 조정 정책에는 기본 준비 시간이 없으므로 인스턴스가 집계된 지표에 즉시 기여하기 시작합니다. 이러한 인스턴스가 시

작 시 리소스 사용량이 많은 경우, 인스턴스를 추가하면 집계된 지표이 급증할 수 있습니다. 사용량이 안정화되는 데 걸리는 시간에 따라 이러한 지표를 사용하는 동적 스케일링 정책에 영향을 미칠 수 있습니다. 동적 스케일링 정책의 경보 최고 임계값을 위반하면 그룹 크기가 다시 증가합니다. 새 인스턴스가 워밍업되는 동안에는 규모 조정이 차단됩니다.

기본 인스턴스 워밍업 시간을 선택합니다.

기본 인스턴스 워밍업 설정의 핵심은 인스턴스가 초기화를 완료하고 리소스 소비가 `InService` 상태에 도달한 후 안정화되는 데 필요한 시간을 결정하는 것입니다. 인스턴스 워밍업 시간을 선택할 때는 정상적인 트래픽을 위한 사용량 데이터 수집과 시작 시 일시적 사용량 급증과 관련된 데이터 수집을 최소화하는 것 사이에서 최적의 균형을 유지하세요.

Elastic Load Balancing 로드 밸런서에 Auto Scaling 그룹이 연결되어 있다고 가정해 보겠습니다. 새 인스턴스 실행이 완료되면 로드 밸런서에 등록되어 `InService` 상태가 되기 전에 로드 밸런서에 등록됩니다. 인스턴스가 `InService` 상태가 된 후에도 리소스 소비가 일시적으로 급증할 수 있으며 안정화하는 데 시간이 필요할 수 있습니다. 예컨대, 큰 자산을 다운로드하고 캐시해야 하는 애플리케이션 서버의 리소스 소비는 다운로드할 큰 자산이 없는 경량 웹 서버보다 안정화하는 데 더 오래 걸립니다. 인스턴스 워밍업은 리소스 소비가 안정화되는 데 필요한 시간 지연을 제공합니다.

> **Important**
>
> 워밍업 시간에 얼마나 많은 시간이 필요한지 잘 모르겠다면 300초부터 시작해도 됩니다. 그런 다음 애플리케이션에 가장 적합한 확장 성능을 얻을 때까지 점진적으로 줄이거나 늘리십시오. 제대로 하려면 이 작업을 몇 번 반복해야 할 수도 있습니다. 또는 자체 워밍업 시간 (`EstimatedInstanceWarmup`) 이 있는 조정 정책이 있는 경우 이 값을 사용하여 시작할 수 있습니다. 자세한 내용은 이전에 설정한 인스턴스 워밍업 시간의 조정 정책을 찾아보세요. 단원을 참조하십시오.

시작 시 실행할 구성 작업 또는 스크립트가 있는 사용 사례에 대해 수명 주기 후크를 사용하는 것을 고려하십시오. 수명 주기 후크는 초기화가 완료될 때까지 새 인스턴스가 서비스 상태가 되는 것을 지연시킬 수 있습니다. 이 기능은 완료하는 데 시간이 걸리는 부트스트랩 스크립트가 있는 경우, 특히 유용합니다. 수명 주기 후크를 추가하면 기본 인스턴스 워밍업 값을 줄일 수 있습니다. 수명 주기 후크 사용에 대한 자세한 설명은 Amazon EC2 Auto Scaling 수명 주기 후크 섹션을 참조하세요.

그룹에 대한 기본 인스턴스 워밍업 활성화

Auto Scaling 그룹을 생성할 때 기본 인스턴스 워밍업을 활성화할 수 있습니다. 기존 그룹에 대해 기본 인스턴스 워밍업을 활성화할 수도 있습니다.

기본 인스턴스 웜업 기능을 활성화하면 다음 기능의 웜업 파라미터 값을 더 이상 지정할 필요가 없습니다.

- 인스턴스 새로 고침
- 대상 추적 스케일링
- 단계적 조정

Console

새 그룹에 대한 기본 인스턴스 워밍업 활성화(콘솔)

Auto Scaling 그룹을 생성할 때 고급 옵션 구성(Configure advanced options) 페이지의 추가 설정(Additional settings)에서 기본 인스턴스 워밍업 활성화(Enable default instance warmup) 옵션을 선택합니다. 애플리케이션에 필요한 워밍업 시간을 선택합니다.

AWS CLI

새 그룹에 대한 기본 인스턴스 워밍업 활성화(AWS CLI)

Auto Scaling 그룹의 기본 인스턴스 워밍업을 활성화하려면 `--default-instance-warmup` 옵션을 추가하고 0에서 3600 사이의 값을 초 단위로 지정합니다. 활성화된 후 `-1` 값은 이 설정을 끕니다.

다음 create-auto-scaling-group명령을 실행하면 다음과 같은 이름을 가진 Auto Scaling 그룹이 생성됩니다.*my-asg* 값이 다음과 같은 기본 인스턴스 웜업을 활성화합니다.*120* 초.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120 ...
```

> **Tip**
> 이 명령에서 오류가 발생하는 경우 AWS CLI 로컬에서 최신 버전으로 업데이트했는지 확인하십시오.

Console

기존 그룹에 대한 기본 인스턴스 워밍업 활성화(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 화면 상단의 탐색 모음에서 Auto Scaling 그룹을 생성한 AWS 리전 을 선택합니다.
3. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
4. 세부 정보(Details) 탭에서 고급 구성(Advanced configurations), 편집(Edit)을 선택합니다.
5. 기본 인스턴스 워밍업의 경우 애플리케이션에 필요한 워밍업 시간을 선택합니다.
6. 업데이트를 선택합니다.

AWS CLI

기존 그룹에 대한 기본 인스턴스 워밍업 활성화(AWS CLI)

다음 예제에서는 update-auto-scaling-group명령을 사용하여 값이 다음과 같은 기본 인스턴스 웜업을 활성화합니다.*120* 이름이 지정된 기존 Auto Scaling 그룹의 경우 초 *my-asg*.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
default-instance-warmup 120
```

> **Tip**
> 이 명령에서 오류가 발생하는 경우 AWS CLI 로컬에서 최신 버전으로 업데이트했는지 확인하십시오.

그룹의 기본 인스턴스 워밍업 시간을 확인하십시오.

다음 절차를 사용하여 Auto Scaling 그룹의 기본 인스턴스 준비 시간을 확인하십시오. AWS CLI

Auto Scaling 그룹의 기본 인스턴스 준비 시간을 확인하려면

다음 describe-auto-scaling-groups 명령을 사용합니다. Replace *my-asg* Auto Scaling 그룹의 이름을 사용합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

다음은 응답의 예입니다.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            ...
            "DefaultInstanceWarmup": 120
        }
    ]
}
```

이전에 설정한 인스턴스 워밍업 시간의 조정 정책을 찾아보세요.

정책에 자체 웜 업 시간이 있는지 확인하려면 를 `EstimatedInstanceWarmup` 사용하여 다음 describe-policy 명령을 실행하십시오. AWS CLI Replace *my-asg* Auto Scaling 그룹의 이름을 사용합니다.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
  --query 'ScalingPolicies[?EstimatedInstanceWarmup!=`null`]'
```

출력의 예제는 다음과 같습니다.

```
[
  {
    "AutoScalingGroupName":"my-asg",
    "PolicyName":"cpu50-target-tracking-scaling-policy",
    "PolicyARN":"arn",
    "PolicyType":"TargetTrackingScaling",
    "StepAdjustments":[],
    "EstimatedInstanceWarmup":120,
    "Alarms":[{
        "AlarmARN": "arn:aws:cloudwatch:us-west-2:123456789012:alarm:TargetTracking-my-
asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
        "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
      },
      {
```

```
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
    }],
    "TargetTrackingConfiguration":{
      "PredefinedMetricSpecification":{
        "PredefinedMetricType":"ASGAverageCPUUtilization"
      },
      "TargetValue":50.0,
      "DisableScaleIn":false
    },
    "Enabled":true
  },

    ... additional policies ...

]
```

스케일링 정책에 대해 이전에 설정한 인스턴스 워밍업을 지웁니다.

기본 인스턴스 워밍업을 활성화한 후 자체 워밍업 시간이 남아 있는 조정 정책을 업데이트하여 이전에 설정한 값을 지우십시오. 그렇지 않으면 기본 인스턴스 워밍업을 재정의합니다.

콘솔 AWS CLI, 또는 를 사용하여 조정 정책을 업데이트할 수 있습니다. AWS SDKs 이 섹션에서는 콘솔의 단계를 다룹니다. AWS CLI AWS SDKsOR를 사용하는 경우 기존 정책 구성을 보존하되 `EstimatedInstanceWarmup` 속성은 제거해야 합니다. 기존 조정 정책을 업데이트하면 프로그래밍 방식으로 PutScalingPolicy호출할 때 지정한 것으로 정책이 대체됩니다. 원래 값은 유지되지 않습니다.

스케일링 정책에 대해 이전에 설정한 인스턴스 워밍업을 지우기(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. Auto Scaling 탭의 동적 스케일링 정책에서 관심 있는 정책을 선택한 다음 작업, 편집을 선택합니다.
4. 인스턴스 워밍업의 경우, 기본 인스턴스 웜업 값을 대신 사용하려면 인스턴스 워밍업 값을 지우십시오.

5. 업데이트를 선택합니다.

Amazon EC2 Auto Scaling을 위한 수동 스케일링

언제든지 Auto Scaling 그룹의 EC2 인스턴스 수를 수동으로 조정할 수 있습니다. 인스턴스 수를 수동으로 변경하는 이 프로세스를 수동 조정이라고 합니다. 수동 크기 조정은 Auto Scaling의 대안이며, 특히 용량을 한 번에 변경하려는 경우 더욱 그렇습니다.

그룹을 수동으로 조정하면 Amazon EC2 Auto Scaling은 사용자가 정의한 조정 정책 및 예약된 작업에 따라 정상적인 Auto Scaling 활동을 재개합니다. 기본 인스턴스 워밍업이 활성화된 그룹의 경우 모든 새 인스턴스는 워밍업 기간을 거쳐 Auto Scaling에 사용되는 지표에 기여하기 시작합니다. 이 워밍업 기간은 그룹을 새 용량으로 안정화하는 데 도움이 됩니다. 자세한 내용은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 단원을 참조하십시오.

때로는 그룹을 수동으로 조정하기 전에 조정 정책과 예약된 작업을 일시적으로 비활성화해야 할 수도 있습니다. 이렇게 하면 수동 조정 작업과 자동 조정 활동 간에 충돌이 발생하는 것을 방지할 수 있습니다. 자세한 내용은 스케일링 활동을 끕니다. 단원을 참조하십시오.

목차

- 기존 Auto Scaling 그룹의 원하는 용량 변경
- Auto Scaling 그룹에서 인스턴스 해지 (AWS CLI)

기존 Auto Scaling 그룹의 원하는 용량 변경

Auto Scaling 그룹의 원하는 용량을 변경하면 Amazon EC2 Auto Scaling에서 원하는 새 크기에 도달하도록 인스턴스를 시작하고 종료하는 프로세스를 관리합니다.

Console

Auto Scaling 그룹의 크기 변경

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 표시됩니다.
3. 세부 정보(Details) 탭에서 그룹 세부 정보(Group details), 편집(Edit)을 선택합니다.

4. 원하는 용량의 경우 원하는 용량을 늘리거나 줄이십시오. 예를 들어, 그룹 크기를 1씩 늘리려면 현재 값이 다음과 1 같으면 를 입력합니다2.

   새 희망 용량 값이 최소 희망 용량 및 최대 희망 용량보다 큰 경우, 최대 희망 용량은 자동으로 새 희망 용량 값으로 증가합니다.

5. 완료되면 업데이트(Update)를 선택합니다.

지정한 그룹 크기로 인해 동일한 양의 인스턴스가 시작되었는지 확인하십시오. 예를 들어 그룹 크기를 하나 늘린 경우 Auto Scaling 그룹에서 인스턴스 하나를 추가로 시작했는지 확인하십시오.

Auto Scaling 그룹의 크기가 변경되었는지 확인

1. 활동 탭의 활동 기록에서 Auto Scaling 그룹과 관련된 활동의 진행 상황을 볼 수 있습니다. 상태 열에는 인스턴스의 현재 상태가 표시됩니다. 인스턴스가 시작되는 동안 상태 열에 `Not yet in service`가 표시됩니다. 인스턴스가 시작되면 상태가 `Successful`로 변경됩니다. 새로 고침 아이콘을 사용하여 인스턴스의 현재 상태를 확인할 수도 있습니다. 자세한 내용은 Auto Scaling 그룹에 대한 크기 조정 활동 확인 단원을 참조하십시오.
2. 인스턴스 관리 탭의 인스턴스에서 인스턴스의 상태를 볼 수 있습니다. 인스턴스를 시작하는 데 약간 시간이 걸립니다.
   - 수명 주기 열에는 인스턴스의 상태가 표시됩니다. 처음에는 인스턴스가 `Pending` 상태로 되어 있습니다. 인스턴스가 트래픽을 수신할 준비가 되면 상태가 `InService`로 전환됩니다.
   - 상태 열에는 인스턴스의 Amazon EC2 Auto Scaling 상태 확인 결과가 표시됩니다.

AWS CLI

다음 예에서는 최소 크기가 1이고 최대 크기가 5인 Auto Scaling 그룹을 생성한 것으로 가정합니다. 따라서 그룹에는 현재 실행 중인 인스턴스가 1개입니다.

Auto Scaling 그룹의 크기 변경

다음 예와 같이 set-desired-capacity명령을 사용하여 Auto Scaling 그룹의 크기를 변경합니다.

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
  --desired-capacity 2
```

Auto Scaling 그룹의 기본 냉각 기간을 지키기로 한 경우, 다음 예에 표시된 것처럼 `--honor-cooldown` 옵션을 지정해야 합니다. 자세한 설명은 Amazon EC2 Auto Scaling의 스케일링 쿨다운 섹션을 참조하세요.

```
aws autoscaling set-desired-capacity --auto-scaling-group-name my-asg \
  --desired-capacity 2 --honor-cooldown
```

Auto Scaling 그룹의 크기 확인

다음 예와 같이 describe-auto-scaling-groups명령을 사용하여 Auto Scaling 그룹의 크기가 변경되었는지 확인합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

다음은 시작된 그룹 및 인스턴스에 대한 세부 정보를 제공하는 예제 출력입니다.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-050555ad16a3f9c7f"
            },
            "MinSize": 1,
            "MaxSize": 5,
            "DesiredCapacity": 2,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 300,
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
```

```
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-05b4f7d5be44822a6",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "Pending"
                },
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-050555ad16a3f9c7f"
                    },
                    "InstanceId": "i-0c20ac468fa3049e8",
                    "InstanceType": "t3.micro",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService"
                }
            ],
            "CreatedTime": "2019-03-18T23:30:42.611Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-c87f2be0",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN": "arn",
            "TrafficSources": []
        }
    ]
}
```

`DesiredCapacity`에 새 값이 표시됩니다. Auto Scaling 그룹에서 추가 인스턴스를 시작했습니다.

Auto Scaling 그룹에서 인스턴스 해지 (AWS CLI)

Auto Scaling 그룹에서 수동으로 스케일링하고 싶지만 특정 인스턴스를 해지하고 싶은 경우가 있을 수 있습니다. 다음 예와 같이 terminate-instance-in-auto-scaling-group 명령을 사용하고 종료하려는 인스턴스의 ID와 `--should-decrement-desired-capacity` 옵션을 지정하여 Auto Scaling 그룹에서 수동으로 규모를 조정할 수 있습니다.

```
aws autoscaling terminate-instance-in-auto-scaling-group \
  --instance-id i-026e4c9f62c3e448c --should-decrement-desired-capacity
```

다음은 조정 활동에 대한 세부 정보를 제공하는 예제 출력입니다.

```
{
    "Activities": [
        {
            "ActivityId": "b8d62b03-10d8-9df4-7377-e464ab6bd0cb",
            "AutoScalingGroupName": "my-asg",
            "Description": "Terminating EC2 instance: i-026e4c9f62c3e448c",
            "Cause": "At 2023-09-23T06:39:59Z instance i-026e4c9f62c3e448c was taken
 out of service in response to a user request, shrinking the capacity from 1 to 0.",
            "StartTime": "2023-09-23T06:39:59.015000+00:00",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":\"us-
west-2c\"}"
        }
    ]
}
```

콘솔에서는 이 옵션을 사용할 수 없습니다. 하지만 Amazon EC2 콘솔의 인스턴스 페이지를 사용하여 Auto Scaling 그룹의 인스턴스를 종료할 수 있습니다. 이렇게 하면 Amazon EC2 Auto Scaling에서 인스턴스가 더 이상 실행되지 않음을 감지하고 상태 확인 프로세스의 일환으로 인스턴스를 자동으로 교체합니다. 인스턴스를 종료한 후 새 인스턴스가 시작되기까지 1~2분 정도 걸립니다. 인스턴스를 종료하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 종료를 참조하십시오.

그룹에서 인스턴스를 종료하여 가용 영역 전체에 균등하게 분배되지 않는 경우, 프로세스를 일시 중단하지 않는 한 Amazon EC2 Auto Scaling은 그룹을 재조정하여 균등한 배포를 재설정합니다. `AZRebalance` 자세한 내용은 Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개 단원을 참조하십시오.

Amazon EC2 Auto Scaling을 위한 스케일링 예약

스케줄링된 스케일링을 사용하면 예측 가능한 부하 변동에 따라 애플리케이션에 대한 자동 스케일링을 설정할 수 있습니다. 특정 시간에 그룹의 원하는 용량을 늘리거나 줄이는 예약된 작업을 생성할 수 있습니다.

예를 들어, 주중에는 부하가 증가하고 주말에는 부하가 감소하는 주간 트래픽 패턴이 정기적으로 발생하는 경우를 예로 들 수 있습니다. Amazon EC2 Auto Scaling에서 다음 패턴에 맞게 조정 일정을 구성할 수 있습니다.

- 수요일 아침에는 예정된 작업 하나가 Auto Scaling 그룹에 이전에 설정된 희망 용량을 늘려 용량을 증가시킵니다.
- 금요일 저녁에 예정된 또 다른 작업은 Auto Scaling 그룹에서 이전에 설정한 원하는 용량을 줄임으로써 용량을 줄입니다.

이러한 예약된 규모 조정 작업을 통해 비용과 성능을 최적화할 수 있습니다. 애플리케이션은 주중에 트래픽이 최고조에 달할 때를 처리할 수 있을 만큼 충분한 용량을 갖추고 있지만, 다른 시간에 불필요한 용량을 오버프로비저닝하지는 않습니다.

스케줄 조정 정책과 규모 조정 정책을 함께 사용하면 두 가지 규모 조정 방식의 이점을 모두 누릴 수 있습니다. 예약된 작업이 실행된 후 조정 정책은 계속해서 용량을 추가로 조정할지를 결정할 수 있습니다. 이를 통해 애플리케이션의 로드를 처리할 수 있는 충분한 용량을 보유하도록 보장합니다. 애플리케이션이 수요에 맞게 조정되는 동안 현재 용량은 예약된 작업에서 설정한 최소 및 최대 용량 이내여야 합니다.

내용

- 예약된 조정 작동 방식
- 반복되는 일정
- 시간대
- 고려 사항
- 제한 사항
- 예약된 작업 생성
- 예약된 작업 세부 정보 보기
- 예약된 작업 삭제

예약된 조정 작동 방식

예약 조정을 사용하려면 Amazon EC2 Auto Scaling에서 특정 시간에 조정 활동을 수행하도록 지시하는 예약 작업을 생성하십시오. 스케줄링된 작업을 생성할 때는 Auto Scaling 그룹, 조정 활동이 발생하는 시기, 원하는 새 용량, 그리고 선택적으로 새로운 최소 용량과 새로운 최대 용량을 지정합니다. 규모를 한 번만 조정하거나 반복되는 일정으로 조정하도록 예약된 작업을 생성할 수 있습니다.

지정된 시간에 Amazon EC2 Auto Scaling은 현재 용량을 지정된 원하는 용량과 비교하여 새 용량 값을 기준으로 조정합니다.

- 현재 용량이 지정된 원하는 용량보다 적을 경우 Amazon EC2 Auto Scaling은 원하는 지정된 용량까지 인스턴스를 확장하거나 추가합니다.
- 현재 용량이 지정된 원하는 용량보다 큰 경우 Amazon EC2 Auto Scaling은 지정된 원하는 용량까지 인스턴스를 확장하거나 제거합니다.

예약된 작업에 따라 지정된 날짜 및 시간에 그룹의 희망 용량, 최소 용량, 최대 용량이 설정됩니다. 이러한 용량 중 하나만 한 번에 원하는 용량 (예: 원하는 용량) 에 대해서만 스케줄링된 작업을 생성할 수 있습니다. 그러나 작업에서 지정한 원하는 용량이 이러한 제한을 벗어나지 않도록 최소 및 최대 용량을 포함해야 하는 경우도 있습니다.

반복되는 일정

AWS CLI 또는 SDK an을 사용하여 반복 일정을 만들려면 cron 표현식과 예약된 작업이 반복되는 시기를 설명하는 시간대를 지정하십시오. 선택적으로 시작 시간, 해지 시간 또는 두 가지 모두에 대한 날짜 및 시간을 지정할 수 있습니다.

를 사용하여 반복 일정을 만들려면 예약된 작업의 반복 패턴 AWS Management Console, 시간대, 시작 시간 및 선택적 종료 시간을 지정하십시오. 모든 반복 패턴 옵션은 cron 표현식에 근거하여 합니다. 또는 사용자 지정 cron 표현식을 작성할 수 있습니다.

지원되는 cron 표현식 형식은 다음과 같이 공백으로 구분된 다섯 개의 필드로 구성됩니다. [Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]. 예컨대, cron 표현식 `30 6 * * 2`는 매주 화요일 오전 6:30에 발생하는 예약된 작업을 구성합니다. 별표는 필드의 모든 값을 일치시키기 위한 와일드카드로 사용됩니다. 크론 표현식의 다른 예는 을 참조하십시오. https://crontab.guru/examples.html 이 형식으로 자체 cron 표현식을 작성하는 방법에 대한 자세한 설명은 Crontab을 참조하세요.

시작 시간과 해지 시간은 신중하게 선택합니다. 다음 사항에 유의하세요.

- 시작 시간을 지정하는 경우 Amazon EC2 Auto Scaling은 이 시간에 작업을 수행한 다음 지정된 반복에 따라 작업을 수행합니다.
- 해지 시간을 지정하면 이 시간 이후에는 작업이 반복되지 않습니다. 예약된 작업은 해지 시간이 되면 해당 계정에 존속하지 않습니다.
- 반복 시간이 종료 시간과 정확히 일치하는 경우 Amazon EC2 Auto Scaling은 종료 시간에 예약된 작업을 수행하지 않습니다.
- AWS CLI 또는 SDK an을 사용할 UTC 때 시작 시간과 종료 시간을 설정해야 합니다.

시간대

기본적으로 설정하는 반복 일정은 협정 세계시 () UTC 를 기준으로 합니다. 현지 표준 시간대 또는 네트워크의 다른 부분에 대한 표준 시간대와 일치하도록 시간을 변경할 수 있습니다. 일광 절약 시간제 (DST) 를 준수하는 시간대를 지정하면 작업이 자동으로 조정됩니다. DST

유효한 값은 인터넷 지정 번호 기관 () IANA 시간대 데이터베이스의 표준 시간대 이름입니다. 예를 들어, 미국 동부 시간은 표준적으로 다음과 같이 식별됩니다. `America/New_York` 자세한 내용은 https://www.iana.org/time-zones 을 참조하십시오.

위치 기반 시간대 (예: `America/New_York` 자동 조정 대상) DST 그러나 와 같은 UTC 기준 시간대는 절대 `Etc/UTC` 시간이므로 조정할 수 없습니다. DST

예컨대, 해당 표준 시간대가 `America/New_York`인 반복되는 일정이 있습니다. 첫 번째 스케일링 작업은 DST 시작 전 `America/New_York` 시간대에 발생합니다. 다음 스케일링 작업은 DST 시작 후 `America/New_York` 시간대에 발생합니다. 첫 번째 작업은 현지 시간으로 오전 UTC 8시~5시에 시작하고, 두 번째 작업은 현지 시간으로 오전 UTC 8시~4시에 시작합니다.

를 사용하여 예약된 작업을 만들고 이를 준수하는 DST 시간대를 지정하면 반복 일정과 시작 및 종료 시간이 모두 자동으로 조정됩니다. AWS Management Console DST

고려 사항

예약된 작업을 만들 경우, 다음 사항에 유의해야 합니다.

- 그룹 전체가 아니라 동일한 그룹 내에서 예약된 작업의 실행 순서가 보장됩니다.
- 예약된 작업은 일반적으로 몇 초 내에 실행됩니다. 하지만 작업이 예약된 시작 시간에서 최대 2분까지 지연될 수 있습니다. 이것은 Auto Scaling 그룹 내의 예약된 작업은 지정된 순서대로 실행하기 때문이며 예약된 시작 시간이 서로 가까운 작업은 실행하는 데 더 많은 시간이 소요될 수 있습니다.

- `ScheduledActions` 프로세스를 일시 중지하여 Auto Scaling 그룹에 대해 예약된 조정을 일시적으로 끌 수 있습니다. 이렇게 하면 예약된 작업을 삭제할 필요 없이 활성 상태가 되는 것을 방지할 수 있습니다. 그런 다음 다시 사용하려는 경우, 예약된 조정을 재개할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개 단원을 참조하십시오.
- 예약된 작업을 생성한 후 이름을 제외한 모든 설정을 업데이트할 수 있습니다.

제한 사항

- 예약된 작업의 이름은 Auto Scaling 그룹별로 고유해야 합니다.
- 예약된 작업은 고유한 시간 값을 가져야 합니다. 다른 크기 조정 활동이 이미 예약된 경우, 한 번에 하나의 활동을 예약하려고 시도하면 호출이 거부되고 예약된 시작 시간에 예약된 작업이 이미 있음을 알리는 오류가 반환됩니다.
- Auto Scaling 그룹당 최대 125개의 예약된 작업을 만들 수 있습니다.

예약된 작업 생성

Auto Scaling 그룹에 예약된 작업을 생성하려면 다음 방법 중 하나를 사용하십시오.

Console

예약된 작업을 만들려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 자동 조정 탭의 Scheduled actions(예약된 작업)에서 Create scheduled action(예약된 작업 생성)을 선택합니다.
4. 예약된 작업의 이름을 입력합니다.
5. 원하는 용량,최소,최대에서 그룹에 새로 원하는 용량과 최소 및 최대 크기 한도를 선택합니다. 원하는 용량은 최소 그룹 크기보다 크거나 같아야 하며 최대 그룹 크기보다 작거나 같아야 합니다.
6. Recurrence(반복)에서 사용 가능한 옵션 중 하나를 선택합니다.

- 반복 일정에 따라 규모를 조정하려면 Amazon EC2 Auto Scaling에서 예약된 작업을 실행할 빈도를 선택하십시오.
  - Every(간격)로 시작하는 옵션을 선택하면 cron 식이 자동으로 생성됩니다.
  - Cron을 선택하는 경우, 작업을 수행하는 시기를 지정하는 Cron 식을 입력합니다.
- 한 번만 조정하려면 Once(한 번)를 선택합니다.

7. Time zone(표준 시간대)에서 시간대를 선택합니다. 기본값은 `Etc/UTC`입니다.

   나열된 모든 시간대는 시간대 데이터베이스에서 가져온 것입니다IANA. 자세한 설명은 https://en.wikipedia.org/wiki/List_of_tz_database_time_zones을 참조하세요.

8. Specific start time(특정 시작 시간)의 날짜 및 시간을 정의합니다.
   - 반복되는 일정을 선택한 경우, 시작 시간은 일련의 반복에서 첫 번째 예약된 작업이 실행되는 시점을 정의합니다.
   - 반복으로 Once(한 번)를 선택하면 시작 시간은 예약된 작업이 실행될 날짜와 시간을 정의합니다.
9. (옵션) 반복되는 일정의 경우Set End Time(해지 시간 설정)을 선택하한 다음 End by(해지 기한)의 날짜 및 시간을 선택하여 해지 시간을 지정할 수 있습니다.
10. 생성(Create)을 선택합니다. 콘솔에 Auto Scaling 그룹에 대해 예약된 작업이 표시됩니다.

AWS CLI

예약된 작업을 만들려면 다음 예제 명령 중 하나를 사용할 수 있습니다. 각각을 바꾸십시오.*user input placeholder* 자신의 정보로

예: 한 번만 규모를 조정하려면

다음 put-scheduled-update-group-action 명령을 `--start-time "YYYY-MM-DDThh:mm:ssZ"` 및 `--desired-capacity` 옵션과 함께 사용하십시오.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-one-time-action \
  --auto-scaling-group-name my-asg --start-time "2021-03-31T08:00:00Z" --desired-capacity 3
```

예: 반복 일정에 따라 규모 조정을 예약하려면

다음 put-scheduled-update-group-action 명령을 `--recurrence "cron expression"` 및 `--desired-capacity` 옵션과 함께 사용하십시오.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-recurring-action \
  --auto-scaling-group-name my-asg --recurrence "0 9 * * *" --desired-capacity 3
```

기본적으로 Amazon EC2 Auto Scaling은 UTC 시간대를 기준으로 지정된 반복 일정을 실행합니다. 다른 시간대를 지정하려면 다음 예와 같이 `--time-zone` 옵션과 IANA 시간대 이름을 포함하십시오.

```
--time-zone "America/New_York"
```

자세한 설명은 https://en.wikipedia.org/wiki/List_of_tz_database_time_zones을 참조하세요.

예약된 작업 세부 정보 보기

Auto Scaling 그룹에 예정된 예정된 작업의 세부 정보를 보려면 다음 방법 중 하나를 사용하십시오.

Console

예약된 작업 세부 정보를 보려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹을 선택합니다.
3. 자동 조정 탭의 예정된 작업 섹션에서 예정된 예정된 작업을 볼 수 있습니다.

참고로 콘솔에는 시작 시간과 종료 시간 값이 현지 시간으로 표시되며, UTC 오프셋은 지정된 날짜 및 시간에 적용됩니다. UTC오프셋은 현지 시간과의 시간 및 분 단위 UTC 차이입니다. Time zone(표준 시간대) 값은 요청한 시간대를 나타냅니다(예: `America/New_York`).

AWS CLI

다음 describe-scheduled-actions 명령을 사용합니다.

```
aws autoscaling describe-scheduled-actions --auto-scaling-group-name my-asg
```

이 명령이 성공하면 다음과 비슷한 출력이 반환됩니다.

```
{
  "ScheduledUpdateGroupActions": [
    {
      "AutoScalingGroupName": "my-asg",
      "ScheduledActionName": "my-recurring-action",
      "Recurrence": "30 0 1 1,6,12 *",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-action",
      "StartTime": "2020-12-01T00:30:00Z",
      "Time": "2020-12-01T00:30:00Z",
      "MinSize": 1,
      "MaxSize": 6,
      "DesiredCapacity": 4
    }
  ]
}
```

크기 조정 활동 확인

예약된 스케일링과 관련된 스케일링 작업을 확인하려면 Auto Scaling 그룹에 대한 크기 조정 활동 확인을 참조하세요.

예약된 작업 삭제

예약된 작업을 삭제하려면 다음 방법 중 하나를 사용하십시오.

Console

예약된 작업 삭제

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 오토 스케일링을 선택합니다.
3. 자동 조정(Automatic scaling) 탭의 예약된 작업(Scheduled actions)에서 예약된 작업을 선택합니다.
4. 작업(Actions), 삭제(Delete)를 선택합니다.
5. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.

AWS CLI

다음 delete-scheduled-action 명령을 사용합니다.

```
aws autoscaling delete-scheduled-action --auto-scaling-group-name my-asg \
  --scheduled-action-name my-recurring-action
```

Amazon EC2 Auto Scaling을 위한 동적 스케일링

동적 조정은 트래픽의 변화에 따라 Auto Scaling 그룹의 용량을 조정합니다.

Amazon EC2 Auto Scaling은 다음과 같은 유형의 동적 조정 정책을 지원합니다.

- 목표 추적 조정 —Amazon CloudWatch 지표와 목표 값을 기반으로 그룹의 현재 용량을 늘리거나 줄입니다. 이는 온도 조절기가 집안 온도를 유지하는 방식과 비슷하게 작동합니다. 사용자가 온도만 선택하면 나머지는 온도 조절기가 알아서 합니다.
- 단계별 조정(Step scaling) - 그룹의 현재 용량을 일련의 조정 조절에 따라 늘리고 줄이며 경보 위반의 크기에 따라 달라지는 단계 조절이라고 합니다.
- 단순 조정(Simple scaling) - 단일 조정 설정에 따라 그룹의 현재 용량을 늘리고 줄입니다. 각 조정 작업 간에는 냉각 기간이 발생합니다.

대상 추적 조정 정책을 사용하고 Auto Scaling 그룹의 용량 변화에 반비례하여 변경되는 지표를 선택하는 것이 좋습니다. 따라서 Auto Scaling 그룹 크기를 두 배로 늘리면 지표가 50% 감소합니다. 이렇게 하면 지표 데이터가 비례적 조정 이벤트를 정확하게 트리거할 수 있습니다. 평균 CPU 사용률 또는 대상당 평균 요청 수와 같은 지표도 포함됩니다.

대상 추적을 사용하면 Auto Scaling 그룹이 애플리케이션의 실제 부하에 정비례하여 조정됩니다. 즉, 대상 추적 정책은 로드 변화에 대응하여 즉각적인 용량 요건을 충족하는 것 외에도, 계절적 변동 등으로 인해 시간 경과에 따라 발생하는 로드 변화에도 맞추어 조정할 수 있습니다.

또한 대상 추적 정책을 사용하면 CloudWatch 경보와 규모 조정을 수동으로 정의할 필요가 없습니다. Amazon EC2 Auto Scaling은 사용자가 설정한 목표에 따라 이를 자동으로 처리합니다.

내용

- 동적 조정 정책 작동 방식
- 여러 동적 조정 정책
- Amazon EC2 Auto Scaling의 대상 추적 조정 정책

- Amazon EC2 Auto Scaling에 대한 단계적이고 간단한 규모 조정 정책
- Amazon EC2 Auto Scaling의 스케일링 쿨다운
- 아마존 기반 규모 조정 정책 SQS
- Auto Scaling 그룹에 대한 크기 조정 활동 확인
- Auto Scaling 그룹에 대한 조정 정책 비활성화
- Auto Scaling 그룹의 조정 정책 삭제
- 에 대한 조정 정책 예시 AWS CLI

동적 조정 정책 작동 방식

동적 조정 정책은 Amazon EC2 Auto Scaling에 특정 CloudWatch 지표를 추적하도록 지시하고, 관련 CloudWatch 경보가 켜졌을 때 취해야 할 조치를 정의합니다. ALARM 경보 상태를 호출하는 데 사용되는 지표는 Auto Scaling 그룹의 모든 인스턴스에서 보내는 지표를 집계한 것입니다. (예를 들어 한 인스턴스가 60% 이고 다른 인스턴스가 40% CPU 인 두 개의 인스턴스가 있는 Auto Scaling 그룹이 있다고 가정해 보겠습니다CPU. 평균적으로 50% CPU 입니다. 정책이 시행되면 Amazon EC2 Auto Scaling은 경보 임계값 위반 시 그룹의 원하는 용량을 늘리거나 줄입니다.

동적 조정 정책을 호출할 때 용량 계산 결과 그룹의 최소 및 최대 크기 범위를 벗어나는 숫자가 생성되는 경우 Amazon EC2 Auto Scaling은 새 용량이 최소 및 최대 크기 제한을 초과하지 않도록 합니다. 용량은 두 가지 방법 중 하나로 측정됩니다. 하나는 원하는 용량을 인스턴스 단위로 설정할 때 선택한 것과 동일한 단위를 사용하거나 용량 단위 (인스턴스 가중치가 적용된 경우) 를 사용하는 것입니다.

- 예 1: Auto Scaling 그룹 최대 용량 3, 현재 용량 2, 인스턴스 3개를 추가하는 조정 정책. Amazon EC2 Auto Scaling은 이 정책을 호출할 때 그룹이 최대 크기를 초과하지 않도록 인스턴스 1개만 그룹에 추가합니다.
- 예 2: Auto Scaling 그룹 최소 용량 2, 현재 용량 3, 인스턴스 2개를 제거하는 동적 조정 정책. 이 정책을 호출할 때 Amazon EC2 Auto Scaling은 그룹이 최소 크기보다 작아지는 것을 방지하기 위해 그룹에서 인스턴스 1개만 제거합니다.

원하는 용량이 최대 크기 제한에 도달하면 스케일 아웃이 중지됩니다. 수요가 감소하고 용량이 감소하면 Amazon EC2 Auto Scaling은 다시 규모를 확장할 수 있습니다.

단, 인스턴스 가중치를 사용하는 경우는 예외입니다. 이 경우 Amazon EC2 Auto Scaling은 최대 인스턴스 무게까지만 최대 크기 한도 이상으로 확장할 수 있습니다. 그 의도는 최대한 새로운 원하는 용량에 근접하면서도 그룹에 지정된 할당 전략을 고수하는 것입니다. 할당 전략에 따라 시작할 인스턴스 유

형이 결정됩니다. 가중치는 인스턴스 유형에 따라 각 인스턴스가 그룹의 원하는 용량에 기여하는 용량 유닛 수를 결정합니다.

- 예 3: Auto Scaling 그룹 최대 용량 12, 현재 용량 10, 용량 유닛 5를 추가하는 동적 조정 정책. 인스턴스 유형에는 1, 4 또는 6의 세 가지 가중치 중 하나가 할당됩니다. 정책을 호출할 때 Amazon EC2 Auto Scaling은 할당 전략에 따라 가중치가 6인 인스턴스 유형을 시작하도록 선택합니다. 이 스케일 아웃 이벤트의 결과는 원하는 용량이 12이고 현재 용량이 16인 그룹입니다.

여러 동적 조정 정책

대부분의 경우에는 Auto Scaling 그룹이 자동으로 스케일 아웃 및 축소되도록 구성하는 데 대상 추적 조정 정책으로 충분합니다. 대상 추적 조정 정책을 사용하면 원하는 결과를 선택하고, 그러한 결과를 얻기 위해 필요에 따라 Auto Scaling 그룹이 인스턴스를 추가 및 제거하도록 할 수 있습니다.

고급 조정 구성의 경우, Auto Scaling 그룹에 하나 이상의 조정 정책이 있을 수 있습니다. 예컨대, 하나 이상의 대상 추적 조정 정책, 하나 이상의 단계별 조정 정책, 또는 둘 모두를 정의할 수 있습니다. 이로 인해 유연성이 늘어나고 여러 시나리오를 처리할 수 있습니다.

여러 동적 조정 정책이 함께 작동하는 방식을 설명하기 위해 Auto Scaling 그룹과 Amazon SQS 대기열을 사용하여 단일 EC2 인스턴스로 요청을 보내는 애플리케이션을 생각해 보십시오. 애플리케이션에서 최적의 수준으로 작업이 수행되도록 보장하기 위해, Auto Scaling 그룹이 스케일 아웃되어야 하는 시기를 제어하는 두 개의 정책이 마련되어 있습니다. 하나는 사용자 지정 지표를 사용하여 대기열의 SQS 메시지 수에 따라 용량을 추가 및 제거하는 대상 추적 정책입니다. 다른 하나는 지정된 기간 동안 인스턴스 사용률이 90% 를 초과할 경우 Amazon CloudWatch `CPUUtilization` 지표를 사용하여 용량을 추가하는 단계별 조정 정책입니다.

동시에 여러 정책이 실행되는 경우, 각 정책이 동시에 Auto Scaling 그룹이 스케일 아웃(또는 축소)하도록 지시할 수 있습니다. 예를 들어, 사용자 지정 `CPUUtilization` 측정치가 급증하여 SQS 사용자 지정 측정치 경보의 임계값을 위반하는 동시에 측정치가 급증하여 CloudWatch 경보 임계값을 위반할 수 있습니다.

이러한 상황이 발생하면 Amazon EC2 Auto Scaling은 스케일 아웃과 스케일 인 모두에 가장 큰 용량을 제공하는 정책을 선택합니다. 예를 들어, 대기열에 대한 정책에서는 인스턴스 하나를 `CPUUtilization` 시작하고 SQS 대기열에 대한 정책에서는 두 개의 인스턴스를 시작한다고 가정해 보겠습니다. 두 정책의 확장 기준이 동시에 충족되는 경우 Amazon EC2 Auto Scaling은 대기열 정책을 우선시합니다SQS. 이렇게 하면 Auto Scaling 그룹에서 인스턴스 두 개가 시작됩니다.

가장 큰 용량을 제공하는 정책에 우선순위를 부여하는 접근법은 다른 스케일 아웃 기준을 사용하는 정책에도 적용됩니다. 예를 들어, 한 정책에서 인스턴스 3개가 종료되고 다른 정책이 인스턴스 수를 25%

줄이며 확장 시 그룹에 8개의 인스턴스가 있는 경우 Amazon EC2 Auto Scaling은 그룹에 가장 많은 수의 인스턴스를 제공하는 정책을 우선시합니다. 따라서 Auto Scaling 그룹은 인스턴스 두 개를 해지합니다(8의 25% = 2). Amazon EC2 Auto Scaling에서 너무 많은 인스턴스를 제거하는 것을 방지하기 위한 것입니다.

그러나 대상 추적 조정 정책과 단계별 조정 정책을 함께 사용하는 경우, 정책 간 충돌로 인해 바람직하지 않은 동작이 발생할 수 있으므로 주의해야 합니다. 예를 들어, 대상 추적 정책이 규모를 축소할 준비가 되기 전에 단계 조정 정책이 활동 축소를 시작하는 경우 활동 축소가 차단되지 않습니다. 활동 축소가 완료되면 대상 추적 정책이 그룹에 다시 규모를 축소하도록 지시할 수 있습니다.

Amazon EC2 Auto Scaling의 대상 추적 조정 정책

대상 추적 조정 정책은 대상 지표 값을 기반으로 Auto Scaling 그룹의 용량을 자동으로 조정합니다. 이를 통해 애플리케이션은 수동 개입 없이 최적의 성능과 비용 효율성을 유지할 수 있습니다.

대상 추적을 사용할 때는 애플리케이션의 이상적인 평균 사용률 또는 처리량 수준을 나타내는 지표와 목표 값을 선택합니다. Amazon EC2 Auto Scaling은 지표가 목표에서 벗어날 때 조정 이벤트를 호출하는 CloudWatch 경보를 생성하고 관리합니다. 예를 들어, 이는 온도 조절기가 목표 온도를 유지하는 방법과 유사합니다.

예를 들어, 현재 두 개의 인스턴스에서 실행되는 애플리케이션이 있고 애플리케이션의 부하가 변경될 때 Auto Scaling 그룹의 CPU 사용률을 약 50% 로 유지하려고 한다고 가정해 보겠습니다. 이로 인해 과도한 유휴 리소스를 유지하지 않고도 트래픽 급증을 처리할 수 있는 추가 용량을 확보할 수 있습니다.

평균 CPU 사용률 50% 를 목표로 하는 목표 추적 조정 정책을 생성하여 이러한 요구를 충족할 수 있습니다. 그러면 Auto Scaling 그룹이 50% CPU 를 초과할 경우 증가된 부하를 처리하기 위해 용량을 확장하거나 늘립니다. 사용률이 낮은 기간 동안 비용을 최적화하기 위해 50% 미만으로 CPU 떨어지면 용량을 축소하거나 축소합니다.

주제

- 다수의 대상 추적 조정 정책
- 지표 선택
- 목표 값 정의
- 인스턴스 워밍업 시간을 정의합니다.
- 고려 사항
- 대상 추적 조정 정책 생성
- 지표 수학을 사용하여 목표 추적 조정 정책을 생성하십시오.

다수의 대상 추적 조정 정책

조정 성능을 최적화하기 위해, 각각 다른 지표를 사용한다는 전제하에 다수의 대상 추적 조정 정책을 함께 사용할 수 있습니다. 예컨대, 사용률과 처리량(throughput)은 서로 영향을 줄 수 있습니다. 이러한 지표 중 하나가 변경될 때마다 일반적으로 다른 지표도 영향을 받게 됩니다. 따라서 여러 지표를 사용하면 Auto Scaling 그룹이 겪고 있는 부하에 대한 추가 정보가 제공됩니다. 이를 통해 Amazon EC2 Auto Scaling에서 그룹에 추가할 용량을 결정할 때 정보에 입각한 결정을 내릴 수 있습니다.

Amazon EC2 Auto Scaling의 목적은 항상 가용성에 우선 순위를 두는 것입니다. 대상 추적 정책 중 규모를 확장할 준비가 되면 Auto Scaling 그룹을 축소합니다. 모든 대상 추적 정책 (부분 축소가 활성화된 상태) 이 축소될 준비가 된 경우에만 축소됩니다.

지표 선택

사용자 지정 또는 사전 정의된 지표를 사용하여 대상 추적 크기 조정 정책을 생성할 수 있습니다.

사전 정의된 지표 유형을 사용하여 대상 추적 크기 조정 정책을 생성하는 경우, 다음의 사전 정의된 지표 목록에서 지표를 하나 선택합니다:

- `ASGAverageCPUUtilization`—Auto Scaling 그룹의 평균 CPU 사용률입니다.
- `ASGAverageNetworkIn` - 모든 네트워크 인터페이스에서 단일 인스턴스가 받은 평균 바이트 수
- `ASGAverageNetworkOut` - 모든 네트워크 인터페이스에서 단일 인스턴스가 보낸 평균 바이트 수
- `ALBRequestCountPerTarget` - 대상당 평균 Application Load Balancer 요청 수

> **Important**
>
> CPU사용률 지표, 네트워크 I/O, 대상당 Application Load Balancer 요청 수에 대한 기타 중요한 정보는 Amazon 사용 설명서의 인스턴스에 사용할 수 있는 CloudWatch 지표 나열 항목과 애플리케이션 로드 밸런서 사용 EC2 설명서의 Application Load Balancer 주제에 대한 CloudWatch 지표에서 각각 확인할 수 있습니다.

사용자 지정 CloudWatch 지표를 CloudWatch 지정하여 사용 가능한 다른 측정치를 선택하거나 자체 측정치를 선택할 수 있습니다. 사용자 지정된 지표 사양이 포함된 대상 추적 정책을 SDK 만들려면 AWS CLI 또는 an을 사용해야 합니다. 를 사용하여 대상 추적 조정 정책에 대한 사용자 지정 지표 사양을 지정하는 예는 을 AWS CLI참조하십시오에 대한 조정 정책 예시 AWS CLI.

지표를 선택하는 경우, 다음 사항에 유의하세요.

- 사용률 변화에 따라 더 빠르게 조정할 수 있도록 1분 간격으로 제공되는 지표만 사용할 것을 권장합니다. 대상 추적은 사전 정의된 모든 지표와 사용자 지정 지표에 대해 1분 단위로 집계된 지표를 평가하지만, 기본 지표는 데이터를 게시하는 빈도가 낮을 수 있습니다. 예를 들어 모든 Amazon EC2 지표는 기본적으로 5분 간격으로 전송되지만 1분으로 구성할 수 있습니다 (세부 모니터링이라고 함). 이 선택은 개별 서비스에 따라 달라집니다. 대부분의 경우, 가능한 가장 짧은 간격을 사용하려고 합니다. 자세한 모니터링 활성화에 대한 자세한 설명은 Auto Scaling 인스턴스에 대한 모니터링 구성을 참조하세요.
- 모든 사용자 지정 지표를 대상 추적에 사용할 수 있는 것은 아닙니다. 지표는 유효한 사용량 수치로서 인스턴스의 사용량을 설명해야 합니다. 지표 값은 Auto Scaling 그룹의 인스턴스 수에 비례하여 증가하거나 감소합니다. 즉, 지표 데이터를 사용하여 인스턴스 수에 따라 비례적으로 스케일 아웃하거나 축소할 수 있습니다. 예를 들어 Auto Scaling 그룹의 부하가 인스턴스 전체에 분산되면 Auto Scaling 그룹의 CPU 사용률 (즉, 지표 차원이 `CPUUtilization` 있는 Amazon EC2 지표`AutoScalingGroupName`) 이 작동합니다.
- 다음 지표는 대상 추적에 사용할 수 없습니다.
  - Auto Scaling 그룹에 대한 로드 밸런서가 수신하는 요청 수(즉, Elastic Load Balancing 지표 `RequestCount`). 로드 밸런서가 수신하는 요청 수는 Auto Scaling 그룹의 사용량에 따라 변경되지 않습니다.
  - 로드 밸런서 요청 지연 시간(즉, Elastic Load Balancing 지표 `Latency`). 요청 지연 시간은 사용량 증가에 따라 늘어날 수는 있지만 반드시 비례하지는 않습니다.
  - CloudWatch 아마존 SQS 큐 메트릭`ApproximateNumberOfMessagesVisible`. 대기열의 메시지 수는 대기열의 메시지를 처리하는 Auto Scaling 그룹의 크기에 비례하여 변경되지 않을 수 있습니다. 하지만 Auto Scaling 그룹의 EC2 인스턴스당 대기열에 있는 메시지 수를 측정하는 사용자 지정 지표는 작동할 수 있습니다. 자세한 내용은 아마존 기반 규모 조정 정책 SQS 단원을 참조하십시오.
- `ALBRequestCountPerTarget` 지표를 사용하려면 `ResourceLabel` 파라미터를 지정하여 지표와 연관된 로드 밸런서 대상 그룹을 식별해야 합니다. 를 사용하여 대상 추적 조정 정책의 `ResourceLabel` 파라미터를 지정하는 예는 을 AWS CLI참조하십시오에 대한 조정 정책 예시 AWS CLI.
- 지표가 실제 0 값을 내보내는 경우 CloudWatch (예:`ALBRequestCountPerTarget`) 일정 기간 동안 애플리케이션에 대한 트래픽이 없을 때 Auto Scaling 그룹은 0으로 확장할 수 있습니다. 해당 그룹에 요청이 라우팅되지 않은 경우, Auto Scaling 그룹을 인스턴스 0개로 스케일 인하려면, 그룹의 최소 용량을 0으로 설정해야 합니다.
- 조정 정책에 사용할 새 지표를 게시하는 대신 지표 수학을 사용하여 기존 지표를 결합할 수 있습니다. 자세한 설명은 지표 수학을 사용하여 목표 추적 조정 정책을 생성하십시오. 섹션을 참조하세요.

목표 값 정의

대상 추적 조정 정책을 생성할 경우, 목표 값을 지정해야 합니다. 목표 값은 Auto Scaling 그룹의 최적 평균 사용률 또는 처리량(throughput)을 나타냅니다. 리소스를 비용 효율적으로 사용하려면 예상치 못한 트래픽 증가에 대비하여, 적절한 버퍼를 두고 목표 값을 가능한 한 높게 설정합니다. 애플리케이션이 정상적인 트래픽 흐름을 위해 최적으로 스케일 아웃되면 실제 지표 값은 목표 값과 같거나 그보다 조금 낮아야 합니다.

조정 정책이 Application Load Balancer 대상당 요청 수, 네트워크 I/O 또는 기타 수 지표와 같은 처리량(throughput)에 근거하여 하는 경우, 대상 값은 1분 동안 단일 인스턴스의 최적 평균 처리량(throughput)을 나타냅니다.

인스턴스 워밍업 시간을 정의합니다.

새로 시작된 인스턴스가 워밍업되는 데 걸리는 시간(초)을 선택적으로 지정할 수 있습니다. 지정된 워밍업 시간이 만료될 때까지 인스턴스는 Auto Scaling 그룹의 집계된 EC2 인스턴스 측정치에 포함되지 않습니다.

인스턴스가 워밍업 기간에 있는 동안에는 워밍업을 하지 않는 인스턴스의 지표 값이 정책의 목표 사용률보다 큰 경우에만 조정 정책이 확장됩니다.

그룹이 다시 스케일 아웃되면 아직 워밍업 중인 인스턴스가 다음 스케일 아웃 활동에 대한 원하는 용량의 일부로 계산됩니다. 계속적이지만 과도하지는 않게 스케일 아웃하기 위한 목적입니다.

확장 활동이 진행 중인 동안에는 인스턴스 워밍업이 완료될 때까지 조정 정책에 의해 시작된 모든 규모 조정 활동이 차단됩니다. 인스턴스 워밍업이 끝나면 스케일 인 이벤트가 발생하면 원하는 새 용량을 계산할 때 현재 종료 중인 모든 인스턴스가 그룹의 현재 용량에 포함됩니다. 따라서 Auto Scaling 그룹에서 인스턴스를 필요 이상으로 빼지 않습니다.

기본값

값이 설정되지 않은 경우 조정 정책은 그룹에 정의된 기본 인스턴스 워밍업 값인 기본값을 사용합니다. 기본 인스턴스 워밍업이 null인 경우 기본 재사용 대기시간 값으로 돌아갑니다. 웜업 시간이 변경될 때 모든 조정 정책을 더 쉽게 업데이트할 수 있도록 기본 인스턴스 워밍업을 사용하는 것이 좋습니다.

고려 사항

대상 추적 조정 정책과 관련한 작업을 수행할 때는 다음 고려 사항이 적용됩니다.

- 대상 추적 조정 정책과 함께 사용되는 CloudWatch 경보를 생성, 편집 또는 삭제하지 마십시오. Amazon EC2 Auto Scaling은 대상 추적 조정 정책과 관련된 CloudWatch 경보를 생성 및 관리하며 더 이상 필요하지 않을 때 이를 삭제합니다.
- 대상 추적 조정 정책은 트래픽이 감소할 때 보다 점진적으로 스케일 인하여 트래픽 수준이 변동하는 기간 동안 가용성을 우선으로 보장합니다. 워크로드가 완료될 때 Auto Scaling 그룹을 즉시 스케일 인하려는 경우, 정책의 스케일 인 부분을 비활성화할 수 있습니다. 이 기능은 사용률이 낮을 때 사용자의 요건에 가장 적합한 스케일 인 방법을 사용할 수 있는 유연성을 제공합니다. 스케일 인이 최대한 빨리 이루어지도록 하려면 냉각 기간이 추가되는 것을 방지하는 단순 조정 정책을 사용하지 않는 것이 좋습니다.
- 지표에 데이터 포인트가 누락된 경우 CloudWatch 경보 상태가 로 변경됩니다. `INSUFFICIENT_DATA` 이 경우 Amazon EC2 Auto Scaling은 새 데이터 포인트를 찾을 때까지 그룹을 확장할 수 없습니다.
- 설계상 지표가 드물게 보고되는 경우, 지표 수학이 유용할 수 있습니다. 예컨대, 가장 최근 값을 사용하려면 `m1`이 지표에 있는 `FILL(m1,REPEAT)` 함수를 사용하세요.
- 대상 값과 실제 지표 데이터 포인트 사이에는 차이가 발생할 수 있습니다. 추가하거나 제거할 인스턴스 수를 결정할 때마다 항상 반올림 또는 내림을 통해 어림짐작으로 동작하기 때문입니다. 이는 인스턴스를 부족하게 추가하거나 너무 많이 제거하는 일을 방지하기 위해서입니다. 하지만 인스턴스가 줄어서 Auto Scaling 그룹이 작아지는 경우에는 그룹의 사용량이 목푯값에서 멀어질 수도 있습니다. 예를 들어 CPU 사용률의 목표 값을 50% 로 설정했는데 Auto Scaling 그룹이 목표를 초과했다고 가정해 보겠습니다. 1.5개의 인스턴스를 추가하면 CPU 사용률이 거의 50% 까지 낮아질 것이라고 판단할 수 있습니다. 하지만 1.5개의 인스턴스를 추가할 수 없기 때문에 반올림을 통해 인스턴스 2개를 추가합니다. 이렇게 하면 CPU 사용률이 50% 미만으로 낮아질 수 있지만 애플리케이션에 이를 지원하는 데 필요한 리소스가 충분하다는 것을 보장할 수 있습니다. 마찬가지로, 1.5개의 인스턴스를 제거하면 CPU 사용률이 50% 이상으로 증가한다고 판단되면 인스턴스 하나만 제거됩니다.

  인스턴스가 더 많고 규모가 큰 Auto Scaling 그룹의 경우, 활용도가 더 많은 수의 인스턴스에 분산되므로 인스턴스를 추가 또는 제거하면 대상 값과 실제 지표 데이터 포인트 간 차이가 줄어듭니다.
- 대상 추적 조정 정책은 지정한 지표가 목푯값을 초과할 때 한해서 Auto Scaling 그룹을 스케일 아웃하게 되어 있습니다. 대상 추적 조정 정책에서는 지정한 지표가 목푯값보다 작을 때 Auto Scaling 그룹을 스케일 아웃할 수 없습니다.

대상 추적 조정 정책 생성

Auto Scaling 그룹에 대한 대상 추적 조정 정책을 생성하려면 다음 방법 중 하나를 사용하십시오.

시작하기 전에 기본 측정치를 1분 간격으로 사용할 수 있는지 확인하십시오 (Amazon EC2 지표의 기본 5분 간격과 비교).

Console

새 Auto Scaling 그룹을 위한 대상 추적 조정 정책 생성

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.
3. 1, 2, 3단계에서 원하는 옵션을 선택하고 4단계: 그룹 크기 및 조정 정책 구성으로 진행합니다.
4. 스케일링 항목에서 specify the range that you want to scale between by updating the 원하는 최소 용량과 원하는 최대 용량을 업데이트하여 조정할 범위를 지정합니다. 이 두 설정을 사용하면 Auto Scaling 그룹이 크기를 동적으로 조정할 수 있습니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.
5. 자동 조정에서 대상 추적 조정 정책을 선택합니다.
6. 정책을 정의하려면 다음을 수행하십시오:
   a. 정책의 이름을 지정합니다.
   b. 지표 유형(Metric type)에서 지표를 선택합니다.

      대상당 Application Load Balancer 요청 수(Application Load Balancer request count per target)를 선택한 경우, 대상 그룹(Target group)에서 대상 그룹을 선택합니다.
   c. 지표의 대상 값(Target value)을 지정합니다.
   d. (선택 사항) 인스턴스 워밍업의 경우 필요에 따라 인스턴스 워밍업 값을 업데이트하십시오.
   e. (옵션) 축소 정책을 비활성화하여 스케일 아웃 정책만 생성(Disable scale in to create only a scale-out policy)을 선택합니다. 이렇게 하면 원할 경우, 유형이 다를 때마다 축소 정책을 별도로 생성할 수 있습니다.
7. 계속해서 Auto Scaling 그룹을 생성합니다. Auto Scaling 그룹이 생성된 후에 조정 정책이 생성됩니다.

기존 Auto Scaling 그룹을 위한 대상 추적 조정 정책 생성

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.

2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.

3. 스케일링 제한이 적절하게 설정되었는지 확인합니다. 예컨대, 그룹의 원하는 용량이 이미 최대치에 있는 경우, 스케일 아웃하려면 새로운 최대값을 지정해야 합니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.

4. 자동 크기 조정(Automatic scaling) 탭의 동적 크기 조정 정책(Dynamic scaling policies)에서 동적 크기 조정 정책 생성(Create dynamic scaling policy)을 선택합니다.

5. 정책을 정의하려면 다음을 수행하십시오:

   a. 정책 유형의 경우, 기본값 대상 추적 조정을 유지합니다.

   b. 정책의 이름을 지정합니다.

   c. 지표 유형(Metric type)에서 지표를 선택합니다. 지표 유형은 하나만 선택할 수 있습니다. 둘 이상의 지표를 사용하려면 여러 정책을 생성합니다.

      대상당 Application Load Balancer 요청 수(Application Load Balancer request count per target)를 선택한 경우, 대상 그룹(Target group)에서 대상 그룹을 선택합니다.

   d. 지표의 대상 값(Target value)을 지정합니다.

   e. (선택 사항) 인스턴스 워밍업의 경우 필요에 따라 인스턴스 워밍업 값을 업데이트하십시오.

   f. (옵션) 축소 정책을 비활성화하여 스케일 아웃 정책만 생성(Disable scale in to create only a scale-out policy)을 선택합니다. 이렇게 하면 원할 경우, 유형이 다를 때마다 축소 정책을 별도로 생성할 수 있습니다.

6. 생성(Create)을 선택합니다.

AWS CLI

대상 추적 조정 정책을 만들려면 다음 예제를 사용하면 시작하는 데 도움이 됩니다. 각각 교체하십시오.*user input placeholder* 자신의 정보로

> **Note**
>
> 더 많은 예제는 에 대한 조정 정책 예시 AWS CLI를 참조합니다.

대상 추적 조정 정책(AWS CLI)을 생성하려면

1. 다음 `cat` 명령을 사용하여 조정 정책의 목표 값과 사전 정의된 지표 사양을 홈 `config.json` 디렉터리의 이름이 지정된 JSON 파일에 저장합니다. 다음은 평균 CPU 사용률을 50% 로 유지하는 대상 추적 구성의 예입니다.

```
$ cat ~/config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification":
    {
      "PredefinedMetricType": "ASGAverageCPUUtilization"
    }
}
```

   자세한 내용은 Amazon EC2 Auto Scaling API 레퍼런스를 참조하십시오 PredefinedMetricSpecification.

2. 이전 단계에서 생성한 `config.json` 파일과 함께 put-scaling-policy명령을 사용하여 조정 정책을 생성합니다.

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

   성공하면 이 명령은 사용자 대신 생성된 두 CloudWatch 경보의 ARNs 및 이름을 반환합니다.

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/cpu50-target-tracking-
scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
```

```
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
        }
    ]
}
```

지표 수학을 사용하여 목표 추적 조정 정책을 생성하십시오.

지표 수학을 사용하면 여러 CloudWatch 지표를 쿼리하고 수학 식을 사용하여 이러한 지표를 기반으로 새 시계열을 만들 수 있습니다. CloudWatch 콘솔에서 결과 시계열을 시각화하고 대시보드에 추가할 수 있습니다. 지표 수학에 대한 자세한 내용은 Amazon 사용 CloudWatch 설명서의 지표 수학 사용을 참조하십시오.

다음은 지표 수학 표현식에 적용되는 고려 사항입니다.

- 사용 가능한 모든 CloudWatch 지표를 쿼리할 수 있습니다. 각 지표는 지표 이름, 네임스페이스, 0개 이상의 측정기준으로 이루어진 고유한 조합입니다.
- 모든 산술 연산자 (+ - */^), 통계 함수 (예: AVG orSUM) 또는 지원하는 기타 함수를 사용할 수 있습니다. CloudWatch
- 수학 표현식의 공식에서 지표 및 다른 수학 표현식의 결과를 모두 사용할 수 있습니다.
- 지표 사양에 사용된 표현식은 결국 단일 시계열을 반환해야 합니다.
- CloudWatch 콘솔이나 를 사용하여 메트릭 수학 표현식이 유효한지 확인할 수 있습니다. CloudWatch GetMetricDataAPI

> **Note**
>
> AWS CLI AWS CloudFormation, 또는 an을 사용하는 경우에만 메트릭 수학을 사용하여 목표 추적 조정 정책을 생성할 수 SDK 있습니다. 이 기능은 콘솔에서는 아직 사용할 수 없습니다.

예: 인스턴스당 Amazon SQS 대기열 백로그

인스턴스당 Amazon SQS 대기열 백로그를 계산하려면 대기열에서 검색할 수 있는 대략적인 메시지 수를 가져와서 이 수를 Auto Scaling 그룹의 실행 용량 (상태 내 인스턴스 수) 으로 나눕니다. `InService` 자세한 내용은 아마존 기반 규모 조정 정책 SQS 단원을 참조하십시오.

표현식의 로직은 다음과 같습니다.

```
sum of (number of messages in the queue)/(number of InService instances)
```

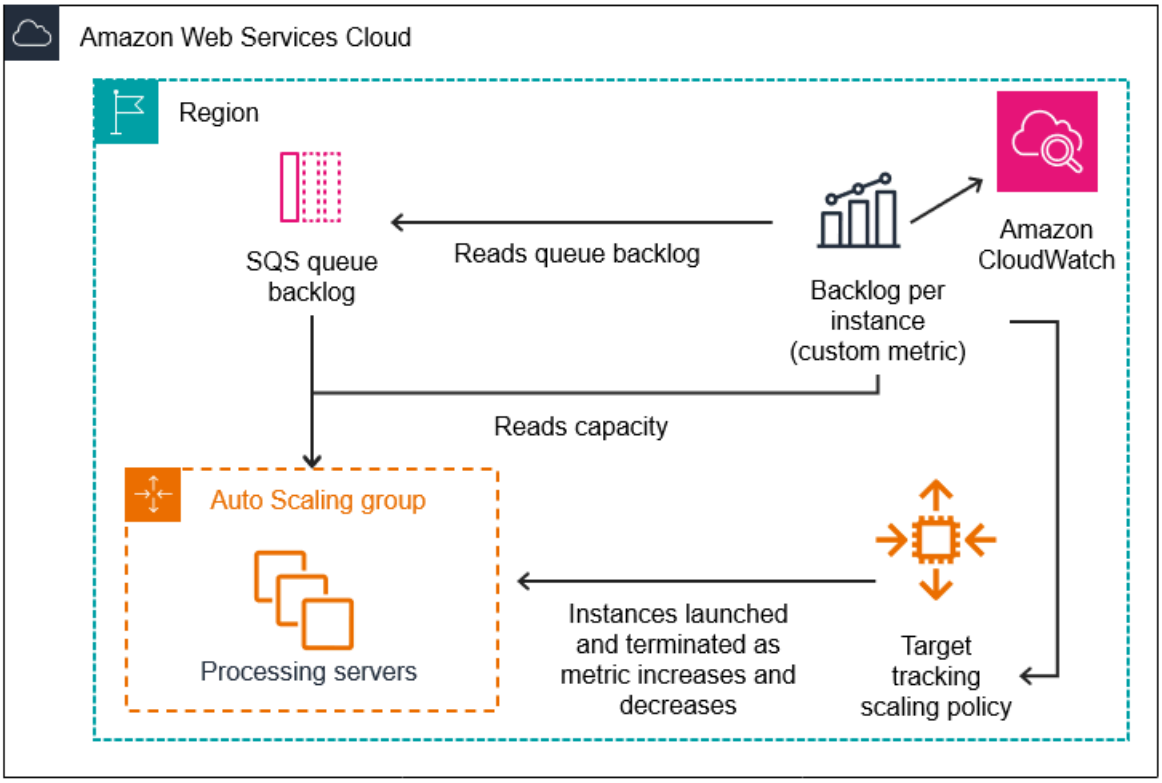
그러면 CloudWatch 측정치 정보는 다음과 같습니다.

ID	CloudWatch 지표	통계	기간
m1	ApproximateNumberOfMessagesVisible	Sum	1분
m2	GroupInServiceInstances	평균	1분

지표 수식 ID와 표현식은 다음과 같습니다.

ID	표현식
e1	`(m1)/(m2)`

다음 다이어그램은 이 지표의 아키텍처를 보여줍니다.



이 지표 수학을 사용하여 대상 추적 조정 정책 생성(AWS CLI)

1. 메트릭 수학 표현식을 사용자 지정된 지표 사양의 일부로 라는 JSON 파일에 저장합니다. `config.json`

   다음 표가 시작하는 데 도움이 될 수 있습니다. 각각 바꾸기 *`user input placeholder`* 자신의 정보로

```
{
    "CustomizedMetricSpecification": {
        "Metrics": [
            {
                "Label": "Get the queue size (the number of messages waiting to be
 processed)",
                "Id": "m1",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "ApproximateNumberOfMessagesVisible",
                        "Namespace": "AWS/SQS",
                        "Dimensions": [
                            {
                                "Name": "QueueName",
```

```
                                "Value": "my-queue"
                            }
                        ]
                    },
                    "Stat": "Sum"
                },
                "ReturnData": false
            },
            {
                "Label": "Get the group size (the number of InService instances)",
                "Id": "m2",
                "MetricStat": {
                    "Metric": {
                        "MetricName": "GroupInServiceInstances",
                        "Namespace": "AWS/AutoScaling",
                        "Dimensions": [
                            {
                                "Name": "AutoScalingGroupName",
                                "Value": "my-asg"
                            }
                        ]
                    },
                    "Stat": "Average"
                },
                "ReturnData": false
            },
            {
                "Label": "Calculate the backlog per instance",
                "Id": "e1",
                "Expression": "m1 / m2",
                "ReturnData": true
            }
        ]
    },
    "TargetValue": 100
}
```

자세한 내용은 Amazon EC2 Auto Scaling API 레퍼런스를 참조하십시오 TargetTrackingConfiguration.

> **Note**
>
> 다음은 지표에 대한 CloudWatch 지표 이름, 네임스페이스, 차원 및 통계를 찾는 데 도움이 되는 몇 가지 추가 리소스입니다.
>
> - AWS 서비스에 사용할 수 있는 지표에 대한 자세한 내용은 Amazon CloudWatch User Guide의 CloudWatch 지표를 게시하는AWS 서비스를 참조하십시오.
> - 가 포함된 지표의 정확한 지표 이름, 네임스페이스 및 차원 (해당하는 경우) 을 가져오려면 목록 CloudWatch 지표를 참조하십시오. AWS CLI

2. 이 정책을 생성하려면 다음 예와 같이 JSON 파일을 입력으로 사용하여 put-scaling-policy명령을 실행합니다.

```
aws autoscaling put-scaling-policy --policy-name sqs-backlog-target-tracking-
scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

성공하면 이 명령은 정책의 Amazon 리소스 이름 (ARN) 과 사용자를 대신하여 생성된 두 개의 CloudWatch 경보 ARNs 중 하나를 반환합니다.

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:228f02c2-c665-4bfd-
aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/sqs-backlog-target-
tracking-scaling-policy",
    "Alarms": [
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
            "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
        },
        {
            "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
            "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
```

```
        }
    ]
}
```

> **Note**
>
> 이 명령에서 오류가 발생하는 경우 AWS CLI 로컬에서 최신 버전으로 업데이트했는지 확인하십시오.

Amazon EC2 Auto Scaling에 대한 단계적이고 간단한 규모 조정 정책

단계별 조정 및 단순 조정 정책은 경보를 기반으로 CloudWatch Auto Scaling 그룹의 용량을 사전 정의된 증분으로 조정합니다. 경보 임계값 위반 시 스케일 아웃(용량 증가) 및 스케일 인(용량 감소)을 처리하도록 별도의 조정 정책을 정의할 수 있습니다.

단계별 조정 및 단순 조정을 사용하면 조정 프로세스를 호출하는 CloudWatch 경보를 생성하고 관리할 수 있습니다. 경보가 위반되면 Amazon EC2 Auto Scaling은 해당 경보와 관련된 조정 정책을 시작합니다.

대상 추적 조정 정책을 사용하여 평균 CPU 사용률 또는 대상당 평균 요청 수와 같은 지표를 기반으로 규모를 조정하는 것이 좋습니다. 용량이 증가할 때 감소하고 용량이 감소할 때 증가하는 지표를 사용하여 비례적으로 스케일 아웃하거나 대상 추적을 사용하여 인스턴스 수를 늘릴 수 있습니다. 이를 통해 Amazon EC2 Auto Scaling이 애플리케이션의 수요 곡선을 거의 따라갈 수 있습니다. 자세한 내용은 대상 추적 조정 정책 단원을 참조하십시오.

목차

- 단계 조정 정책 작동 방식
- 단계별 조정 정책에 대한 단계별 조정
- 조정 조절 유형
- 인스턴스 워밍업
- 고려 사항
- 스케일 아웃을 위한 단계별 조정 정책을 생성하세요.
- 확장을 위한 단계별 조정 정책을 생성하십시오.
- 단순 조정 정책

단계 조정 정책 작동 방식

단계별 스케일링을 사용하려면 먼저 Auto Scaling 그룹의 지표를 모니터링하는 CloudWatch 경보를 생성합니다. 경보 위반을 결정하는 지표, 임계값, 평가 기간 수를 정의합니다. 그런 다음 경보 임계값 위반 시 그룹을 조정하는 방법을 정의하는 단계별 조정 정책을 생성하십시오.

정책에 단계 조정을 추가합니다. 경보의 위반 규모에 따라 다양한 단계 조정을 정의할 수 있습니다. 예:

- 알람 지표가 60% 에 도달하면 인스턴스를 10개씩 축소합니다.
- 알람 지표가 75% 에 도달하면 인스턴스를 30개까지 확장할 수 있습니다.
- 알람 지표가 85% 에 도달하면 40개 인스턴스까지 확장할 수 있습니다.

지정된 평가 기간 동안 경보 임계값이 위반되면 Amazon EC2 Auto Scaling은 정책에 정의된 단계 조정을 적용합니다. 경보 상태가 OK로 돌아갈 때까지 추가 경보 위반에 대해 조정을 계속할 수 있습니다.

각 인스턴스에는 조정 활동이 짧은 기간 동안 발생하는 변경에 너무 민감하지 않도록 준비 기간이 있습니다. 조정 정책의 워밍업 기간을 선택적으로 구성할 수 있습니다. 하지만 워밍업 시간이 변경될 때 모든 조정 정책을 더 쉽게 업데이트할 수 있도록 기본 인스턴스 웜업을 사용하는 것이 좋습니다. 자세한 내용은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 단원을 참조하십시오.

단순 조정 정책은 각 조정 활동 사이에 휴지 기간이 있는 단일 조정 조정을 기반으로 한다는 점을 제외하면 단계별 조정 정책과 비슷합니다. 자세한 내용은 단순 조정 정책 단원을 참조하십시오.

단계별 조정 정책에 대한 단계별 조정

단계별 조정 정책을 생성할 때 경보 위반의 크기에 따라 인스턴스 수를 동적으로 자동 조정되도록 하나 이상의 단계별 조정을 지정합니다. 각 단계별 조정은 다음을 지정합니다.

- 지표 값의 하한값입니다.
- 지표 값의 상한값입니다.
- 조정 유형에 근거하여 축소하거나 스케일 아웃하는 양입니다.

CloudWatch 경보와 관련된 지표의 통계를 기반으로 지표 데이터 포인트를 집계합니다. CloudWatch 경보를 위반하면 적절한 조정 정책이 호출됩니다. Amazon EC2 Auto Scaling은 CloudWatch (원시 지표 데이터와 반대) 의 가장 최근 지표 데이터 요소에 집계 유형을 적용합니다. 이 집계된 지표 값을 단계별 조정으로 정의된 상한값 및 하한값과 비교하여 어느 단계의 조정을 수행할 것인지 결정합니다.

위반 임계값과 연계하여 상한값과 하한값을 지정합니다. 예를 들어 지표가 50% 를 초과하는 경우에 대비하여 CloudWatch 경보와 확장 정책을 만들었다고 가정해 보겠습니다. 그런 다음 지표가

50% 미만일 때를 대비한 두 번째 경보와 스케일 인 정책을 만들었습니다. 각 정책에 대해 조정 유형 `PercentChangeInCapacity` (또는 콘솔의 그룹 비율) 을 사용하여 일련의 단계 조정을 수행했습니다.

예: 스케일 아웃 정책에 대한 단계 조절

하한	상한	조절
0	10	0
10	20	10
20	null	30

예: 축소 정책에 대한 단계 조절

하한	상한	조절
-10	0	0
-20	-10	-10
null	-20	-30

이렇게 하면 다음과 같은 조정 구성이 생성됩니다.

```
Metric value

-infinity          30%    40%          60%     70%             infinity
---------------------------------------------------------------------------
          -30%      | -10% | Unchanged  | +10%  |       +30%
---------------------------------------------------------------------------
```

이제 현재 용량과 원하는 용량이 모두 10인 Auto Scaling 그룹에서 이 조정 구성을 사용한다고 가정해 보겠습니다. 다음은 그룹의 원하는 용량과 현재 용량과 관련된 조정 구성의 동작을 요약한 내용입니다.

- 집계된 지표 값이 40보다 크고 60보다 작으면 현재 용량과 원하는 용량이 유지됩니다.
- 지표 값이 60이 되면 스케일 아웃 정책의 2단계 조절(10개의 인스턴스에 10% 합산)에 따라 그룹의 원하는 용량을 1개의 인스턴스만큼 늘려 11개의 인스턴스로 만듭니다. 새 인스턴스가 실행되고 지정

된 웜업 시간이 만료되면 그룹의 현재 용량은 11개의 인스턴스로 늘어납니다. 용량이 이렇게 증가한 후에도 지표 값이 70으로 증가하면 그룹의 원하는 용량이 14개의 인스턴스까지 증가합니다(인스턴스가 3개씩 증가). 이는 스케일 아웃 정책의 세 번째 단계 조절에 근거하여 합니다(11개 인스턴스의 30%인 3.3개 인스턴스가 3개로 반올림되어 추가).

- 지표 값이 40이 될 경우, 축소 정책의 2단계 조절에 따라(14개의 인스턴스에 10%인 1.4개의 인스턴스를 빼되 1개의 인스턴스로 반내림) 그룹의 원하는 용량을 1개의 인스턴스만큼 더 줄여 13개의 인스턴스로 만듭니다. 이렇게 용량이 감소한 후에도 지표 값이 30으로 떨어지면 그룹의 원하는 용량이 10개의 인스턴스로 감소합니다(인스턴스가 3개씩 감소). 이는 스케일 인 정책의 세 번째 단계 조절에 근거하여 합니다(13개 인스턴스 30%인 3.9개 인스턴스가 3개로 반내림되어 제거).

조정 정책에 대한 단계 조절을 지정할 때는 다음 사항에 유의하세요.

- 를 사용하는 경우 상한과 하한을 절대값으로 지정합니다. AWS Management Console AWS CLI 또는 SDK an을 사용하는 경우 위반 임계값을 기준으로 상한과 하한을 지정합니다.
- 단계 조절의 범위는 중복되거나 격차가 있어서는 안 됩니다.
- 1단계 조절에만 null 하한값(negative infinity)이 포함될 수 있습니다. 1단계 조절에 음의 하한값이 포함될 경우, null 하한값으로 단계 조절을 해야 합니다.
- 1단계 조절에만 null 상한값(positive infinity)이 포함될 수 있습니다. 1단계 조절에 양의 상한값이 포함될 경우, null 상한값으로 단계 조절을 해야 합니다.
- 상한 및 하한값은 동일한 단계 조절에서 null이 될 수 없습니다.
- 지표 값이 위반 임계값을 초과할 경우, 하한값은 포함되고 상한값은 제외됩니다. 지표 값이 위반 임계값 미만일 경우, 하한값은 제외되고 상한값은 포함됩니다.

조정 조절 유형

선택한 조정 조절 유형에 따라 최적의 조정 작업을 수행하는 조정 정책을 정의할 수 있습니다. 조절 유형을 Auto Scaling 그룹의 현재 용량의 백분율이나 용량 단위로 지정할 수 있습니다. 인스턴스 가중치 기능을 사용하지 않는 한 일반적으로 용량 단위는 인스턴스 1개를 의미합니다.

Amazon EC2 Auto Scaling은 단계별 조정 및 단순 조정을 위해 다음과 같은 조정 유형을 지원합니다.

- `ChangeInCapacity` - 그룹의 현재 용량을 지정된 값만큼 늘리거나 줄입니다. 양의 값은 용량을 늘리고, 음의 조절 값은 용량을 줄입니다. 예: 그룹의 현재 용량이 3개의 인스턴스이고 조절이 5개인 경우, 이 정책이 수행되면 용량 단위가 총 8개가 되도록 용량에 5개의 용량 단위를 추가합니다.

- `ExactCapacity` - 그룹의 현재 용량을 지정된 값으로 변경합니다. 이 조절 유형에는 음이 아닌 값을 지정합니다. 예: 그룹의 현재 용량이 3개의 인스턴스이고 조절이 5개인 경우, 이 정책이 수행되면 용량을 5개의 용량 단위로 변경합니다.
- `PercentChangeInCapacity` - 그룹의 현재 용량을 지정된 퍼센트만큼 늘리거나 줄입니다. 양의 값은 용량을 늘리고, 음의 값은 용량을 줄입니다. 예: 현재 용량이 10개이고 조절이 10%인 경우, 이 정책이 수행되면 용량 단위가 총 11개가 되도록 용량에 1개의 용량 단위를 추가합니다.

> **Note**
>
> 결과 값이 정수가 아닌 경우, 다음과 같이 반올림(반내림)합니다.
>
> - 1보다 큰 값은 반내림합니다. 예컨대, `12.7`은 `12`로 반내림합니다.
> - 0과 1 사이의 값은 1로 반올림합니다. 예컨대, `.67`은 `1`로 반올림합니다.
> - 0과 -1 사이의 값은 1로 반내림합니다. 예컨대, `-.58`은 `-1`으로 반올림합니다.
> - -1보다 작은 값은 반올림합니다. 예컨대, `-6.67`은 `-6`으로 반올림합니다.

`PercentChangeInCapacity`에서는 `MinAdjustmentMagnitude` 파라미터를 사용하여 조정할 최소 인스턴스 수를 지정할 수도 있습니다. 예컨대, 25퍼센트를 추가하는 정책을 생성하고 최소 증분으로 2개의 인스턴스를 지정한다고 가정해 보세요. Auto Scaling 그룹에 4개 인스턴스가 있고 조정 정책이 실행되는 경우, 4개 인스턴스의 25%는 1개 인스턴스입니다. 그러나 최소 증분을 2로 지정했기 때문에 2개의 인스턴스가 추가됩니다.

인스턴스 가중치를 사용하면 `MinAdjustmentMagnitude` 파라미터를 0이 아닌 값으로 설정하는 효과가 변경됩니다. 값은 용량 유닛입니다. 조정할 최소 인스턴스 수를 설정하려면 이 파라미터를 가장 큰 인스턴스 가중치 이상의 값으로 설정합니다.

인스턴스 가중치를 사용하는 경우 Auto Scaling 그룹의 현재 용량이 필요에 따라 원하는 용량을 초과할 수 있다는 점을 염두에 두십시오. 절대 감소 수 또는 백분율에서 감소하는 양이 현재 용량과 원하는 용량 간의 차이보다 작으면 조정 작업이 수행되지 않습니다. 임계값 경보가 위반되면 조정 정책의 결과를 확인할 때 이러한 동작을 고려해야 합니다. 예컨대, 원하는 용량이 30개이고 현재 용량이 32개라고 가정해 보겠습니다. 경보가 위반되면 조정 정책이 원하는 용량을 1씩 줄이게 되고 조정 작업이 수행되지 않습니다.

인스턴스 워밍업

단계별 조정의 경우, 새로 시작된 인스턴스가 워밍업되는 데 걸리는 시간(초)을 선택적으로 지정할 수 있습니다. 지정된 워밍업 시간이 만료될 때까지 인스턴스는 Auto Scaling 그룹의 집계된 EC2 인스턴스 측정치에 포함되지 않습니다.

인스턴스가 워밍업 기간에 있는 동안에는 워밍업을 하지 않는 인스턴스의 지표 값이 정책의 경보 상한 임계값보다 큰 경우에만 조정 정책이 확장됩니다.

그룹이 다시 스케일 아웃되면 아직 워밍업 중인 인스턴스가 다음 스케일 아웃 활동에 대한 원하는 용량의 일부로 계산됩니다. 따라서 여러 개의 경보 위반이 동일한 단계 조절 범위에 해당하는 경우, 단일 크기 조정 활동이 이루어집니다. 계속적이지만 과도하지는 않게 스케일 아웃하기 위한 목적입니다.

예컨대, 두 단계로 구성된 정책을 만든다고 가정해 보겠습니다. 첫 번째 단계는 지표이 60%에 도달하면 10%를 추가하고, 두 번째 단계는 지표이 70%에 도달하면 30%를 추가합니다. Auto Scaling 그룹의 원하는 용량과 현재 용량은 10입니다. 집계된 지표 값이 60 미만인 동안에는 원하는 및 현재의 용량은 변하지 않습니다. 지표이 60에 도달하여 인스턴스 1개(10개 인스턴스의 10%)가 추가되었다고 가정합니다. 그런 다음 새 인스턴스가 아직 워밍업 중인 동안 지표는 62가 됩니다. 스케일링 정책은 현재 용량(여전히 10)을 기준으로 새로 원하는 용량을 계산합니다. 하지만 그룹의 필요한 용량이 이미 11개 인스턴스로 증가하였으므로 조정 정책이 원하는 용량을 더 이상 늘리지 않습니다. 지표가 70이 되고 새 인스턴스가 여전히 워밍업 중이면 3개의 인스턴스(10개 인스턴스의 30퍼센트)를 추가해야 합니다. 하지만 그룹의 필요한 용량이 이미 11이므로 새로운 필요 용량 13개 인스턴스에 대해 2개의 인스턴스만 추가합니다.

스케일 아웃 활동이 진행 중인 동안에는 인스턴스 워밍업이 완료될 때까지 조정 정책에 의해 시작된 모든 축소 활동이 차단됩니다. 인스턴스 워밍업이 완료된 후 스케일 인 이벤트가 발생하면 현재 해지 중인 인스턴스는 새로 원하는 용량을 계산할 때 그룹의 현재 용량에 포함됩니다. 따라서 Auto Scaling 그룹에서 인스턴스를 필요 이상으로 빼지 않습니다. 예컨대, 인스턴스가 이미 해지되고 있는 동안 원하는 용량을 1씩 감소시킨 동일한 단계별 조정 범위에서 경보가 위반되는 경우, 스케일링 작업이 수행되지 않습니다.

기본값

값이 설정되지 않은 경우 조정 정책은 그룹에 정의된 기본 인스턴스 웜업 값인 기본값을 사용합니다. 기본 인스턴스 워밍업이 null인 경우 기본 재사용 대기시간 값으로 돌아갑니다.

고려 사항

단계별 및 단순 조정 정책과 관련한 작업을 수행할 때는 다음 고려 사항이 적용됩니다:

- 단계별 조정을 사용할 수 있을 만큼 애플리케이션의 단계별 조정을 정확하게 예측할 수 있는지 생각해 보세요. 조정 지표가 스케일 아웃 가능한 대상의 용량에 비례하여 증가하거나 감소하는 경우, 대상 추적 조정 정책을 대신 사용하는 것이 좋습니다. 단계별 조정을 고급 구성에 대한 추가 정책으로 사용할 수도 있습니다. 예컨대, 사용률이 일정 수준에 도달할 때 더 공격적인 대응을 구성할 수 있습니다.
- 플래핑을 방지하려면 스케일 아웃 임계값과 스케일 인 임계값 사이에서 적절한 마진을 선택해야 합니다. 플래핑은 스케일 인과 스케일 아웃의 무한 루프입니다. 즉, 스케일 아웃 작업을 진행하면 지표 값이 변경되고 반대 방향으로 다른 스케일 아웃 작업이 시작됩니다.

스케일 아웃을 위한 단계별 조정 정책을 생성하세요.

Auto Scaling 그룹의 스케일 아웃을 위한 단계별 조정 정책을 생성하려면 다음 방법 중 하나를 사용하십시오.

Console

1단계: 지표 상한 임계값에 대한 CloudWatch 경보 생성

1. 에서 CloudWatch 콘솔을 엽니다 https://console.aws.amazon.com/cloudwatch/.
2. 필요한 경우, 지역을 변경합니다. 탐색 모음에서 Auto Scaling 그룹이 상주하는 지역을 선택합니다.
3. 탐색 창에서 경보(Alarms), 모든 경보(All alarms)를 선택한 다음 경보 생성(Create alarm)을 선택합니다.
4. 지표 선택을 선택하세요.
5. 모든 지표 탭에서 Auto Scaling 그룹별을 선택하고 EC2검색 필드에 Auto Scaling 그룹 이름을 입력합니다. 그런 다음 `CPUUtilization`을 선택하고 지표 선택(Select metric)을 선택합니다. 지표에 대한 그래프와 기타 정보가 표시된 지표 및 조건 지정(Specify metric and conditions) 페이지가 나타납니다.
6. 기간(Period)에 예컨대, 1분과 같은 경보에 대한 평가 기간을 선택합니다. 경보를 평가할 때 각 기간이 하나의 데이터 포인트로 집계됩니다.

   > **Note**
   > 기간이 짧을수록 경보가 더 민감해집니다.

7. 조건(Conditions)에서 다음을 수행하십시오:

- 임계값 유형(Threshold type)에서 정적(Static)을 선택합니다.
- **`CPUUtilization`**Whenever is의 경우 지표 값을 경보 위반 임계값보다 크거나 같게 할지 여부를 지정합니다. 그런 다음 기준(than)에 경보를 위반하려는 임계값을 입력합니다.

> **Important**
>
> 스케일 아웃 정책(지표 상한)과 함께 사용할 경보의 경우, 임계값보다 작은 항목이나 작거나 같은 항목을 선택하지 않았는지 확인합니다.

8. 추가 구성(Additional configuration)에서 다음을 수행하십시오:
   - 경보에 대한 데이터 포인트(Datapoints to alarm)에서 지표 값이 경보에 대한 임계값 조건을 충족해야 하는 데이터 포인트(평가 기간)를 입력합니다. 예컨대, 5분의 시간이 두 차례 연속되면 경보 상태를 호출하기까지 10분이 걸립니다.
   - 누락 데이터 처리(Missing data treatment)에서 누락 데이터를 불량으로 처리(임계값 위반)(Treat missing data as bad (breaching threshold))를 선택합니다. 자세한 내용은 Amazon CloudWatch User Guide의 CloudWatch 경보가 누락된 데이터를 처리하는 방법 구성을 참조하십시오.
9. Next(다음)를 선택합니다.

   작업 구성 페이지가 표시됩니다.
10. 알림에서 Amazon SNS 주제를 선택하여 경보가 `ALARM` 상태, `OK` 상태 또는 `INSUFFICIENT_DATA` 상태에 있을 때 알립니다.

    경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

    경보에서 알림을 보내지 않게 하려면 제거(Remove)를 선택합니다.
11. 작업 구성(Configure actions) 페이지의 다른 섹션은 비워둘 수 있습니다. 다른 섹션을 비워 두면 조정 정책에 연결하지 않고도 경보가 생성됩니다. 그런 다음 Amazon EC2 Auto Scaling 콘솔에서 조정 정책과 경보를 연결할 수 있습니다.
12. Next(다음)를 선택합니다.
13. 경보의 이름(예: `Step-Scaling-AlarmHigh-AddCapacity`)과 설명(옵션)을 입력하고 다음(Next)을 선택합니다.
14. 경보 생성(Create alarm)을 선택하십시오.

다음 절차를 사용하여 CloudWatch 경보를 생성한 후 중단한 부분부터 계속 진행하십시오.

2단계: 스케일 아웃을 위한 단계별 조정 정책 생성

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 스케일링 제한이 적절하게 설정되었는지 확인합니다. 예컨대, 그룹의 원하는 용량이 이미 최대치에 있는 경우, 스케일 아웃하려면 새로운 최대값을 지정해야 합니다. 자세한 설명은 Auto Scaling 그룹에 대한 스케일링 제한 설정 섹션을 참조하세요.
4. 자동 크기 조정(Automatic scaling) 탭의 동적 크기 조정 정책(Dynamic scaling policies)에서 동적 크기 조정 정책 생성(Create dynamic scaling policy)을 선택합니다.
5. 정책 유형에서 Step scaling을 선택한 다음 정책의 이름을 지정합니다.
6. CloudWatch 경보의 경우 경보를 선택합니다. 아직 알람을 생성하지 않은 경우 알람 생성을 선택하고 이전 절차의 4단계부터 14단계까지 완료하여 알람을 생성합니다. CloudWatch
7. 다음 작업을 수행(Take the action)을 사용하여 이 정책을 실행할 때 적용할 현재 그룹 크기 변경을 지정합니다. 특정 인스턴스 개수 또는 기존 그룹 크기의 백분율을 추가하거나 그룹을 정확한 크기로 설정할 수 있습니다.

   예를 들어 그룹 용량을 30% 늘리는 스케일 아웃 정책을 생성하려면 을 선택하고 다음 필드에를 입력한 `30` 다음 선택하십시오Add. `percent of group` 기본적으로 이 단계별 조정의 하한값은 경보 임계값이고 상한값은 양(+)의 무한대입니다.
8. 다른 단계를 추가하려면 단계 추가(Add step)를 선택한 다음 조정할 양과 경보 임계값에 상대적인 단계 하한 및 상한을 정의합니다.
9. 조정할 최소 인스턴스 수를 설정하려면 최소 1 단위 용량의 증분으로 용량 단위 추가(Add capacity units in increments of at least 1 capacity units)의 숫자 필드를 업데이트합니다.
10. (선택 사항) 인스턴스 웜업의 경우 필요에 따라 인스턴스 웜업 값을 업데이트합니다.
11. 생성(Create)을 선택합니다.

AWS CLI

스케일 아웃 (용량 증가) 을 위한 단계별 조정 정책을 생성하려면 다음 예제 명령을 사용할 수 있습니다. 각각 교체하십시오.*`user input placeholder`* 자신의 정보로

를 사용할 때는 먼저 지표 값이 증가할 때 규모를 축소하는 방법에 대한 지침을 Amazon EC2 Auto Scaling에 제공하는 단계별 조정 정책을 생성합니다. AWS CLI그런 다음 관찰할 지표를 식별하고, 측정치 상한 임계값 및 경보에 대한 기타 세부 정보를 정의하고, 경보를 조정 정책과 연결하여 경보를 생성합니다.

1단계: 스케일 아웃을 위한 정책 생성

다음 put-scaling-policy명령을 사용하여 다음 단계 조정 (CloudWatch 경보 임계값을 60% 로 가정)에 따라 그룹 용량을 늘리는 조정 유형을 `PercentChangeInCapacity` 포함하는 단계 조정 정책을 생성합니다. `my-step-scale-out-policy`

- 지표 값이 60% 이상이고 75% 미만인 경우, 인스턴스 수를 10% 늘립니다.
- 지표 값이 75% 이상이고 85% 미만인 경우, 인스턴스 수를 20% 늘립니다.
- 지표 값이 85% 이상이면 인스턴스 수를 30% 증가시킵니다.

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=0.0,MetricIntervalUpperBound=15.0,ScalingAdjustment=10 \

 MetricIntervalLowerBound=15.0,MetricIntervalUpperBound=25.0,ScalingAdjustment=20 \
                     MetricIntervalLowerBound=25.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. 정책에 대한 CloudWatch 경보를 만들려면 이 정보가 필요합니다.

```
{
    "PolicyARN":
 "arn:aws:autoscaling:region:123456789012:scalingPolicy:4ee9e543-86b5-4121-b53b-
aa4c23b5bbcc:autoScalingGroupName/my-asg:policyName/my-step-scale-in-policy
}
```

2단계: 지표 상한 임계값에 대한 CloudWatch 경보 생성

다음 CloudWatch put-metric-alarm명령을 사용하여 2분 이상의 연속 평가 기간 동안 평균 CPU 임계값 60% 를 기준으로 Auto Scaling 그룹의 크기를 늘리는 경보를 생성합니다. 사용자 지정 지표를 사용하려면 `--metric-name`에 지표 이름을 지정하고, `--namespace`에 네임스페이스를 지정합니다.

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmHigh-AddCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 60 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --alarm-actions PolicyARN
```

확장을 위한 단계별 조정 정책을 생성하십시오.

Auto Scaling 그룹의 축소를 위한 단계별 조정 정책을 생성하려면 다음 방법 중 하나를 사용하십시오.

Console

1단계: 지표 하한 임계값에 대한 CloudWatch 경보 생성

1. 에서 CloudWatch 콘솔을 엽니다 https://console.aws.amazon.com/cloudwatch/.
2. 필요한 경우, 지역을 변경합니다. 탐색 모음에서 Auto Scaling 그룹이 상주하는 지역을 선택합니다.
3. 탐색 창에서 경보(Alarms), 모든 경보(All alarms)를 선택한 다음 경보 생성(Create alarm)을 선택합니다.
4. 지표 선택을 선택하세요.
5. 모든 지표 탭에서 Auto Scaling 그룹별을 선택하고 EC2검색 필드에 Auto Scaling 그룹 이름을 입력합니다. 그런 다음 `CPUUtilization`을 선택하고 지표 선택(Select metric)을 선택합니다. 지표에 대한 그래프와 기타 정보가 표시된 지표 및 조건 지정(Specify metric and conditions) 페이지가 나타납니다.
6. 기간(Period)에 예컨대, 1분과 같은 경보에 대한 평가 기간을 선택합니다. 경보를 평가할 때 각 기간이 하나의 데이터 포인트로 집계됩니다.

   > **Note**
   > 기간이 짧을수록 경보가 더 민감해집니다.

7. 조건(Conditions)에서 다음을 수행하십시오:

- 임계값 유형(Threshold type)에서 정적(Static)을 선택합니다.
- **`CPUUtilization`**Whenever is의 경우 지표 값을 경보 위반 임계값보다 작게 할지 또는 같게 할지를 지정합니다. 그런 다음 기준(than)에 경보를 위반하려는 임계값을 입력합니다.

> **Important**
>
> 정책 규모 (지표 하한) 와 함께 경보를 사용하려면 임계값보다 크거나 같음을 선택하지 마십시오.

8. 추가 구성(Additional configuration)에서 다음을 수행하십시오:
   - 경보에 대한 데이터 포인트(Datapoints to alarm)에서 지표 값이 경보에 대한 임계값 조건을 충족해야 하는 데이터 포인트(평가 기간)를 입력합니다. 예컨대, 5분의 시간이 두 차례 연속되면 경보 상태를 호출하기까지 10분이 걸립니다.
   - 누락 데이터 처리(Missing data treatment)에서 누락 데이터를 불량으로 처리(임계값 위반)(Treat missing data as bad (breaching threshold))를 선택합니다. 자세한 내용은 Amazon CloudWatch User Guide의 CloudWatch 경보가 누락된 데이터를 처리하는 방법 구성을 참조하십시오.
9. Next(다음)를 선택합니다.

   작업 구성 페이지가 표시됩니다.
10. 알림에서 Amazon SNS 주제를 선택하여 경보가 `ALARM` 상태, `OK` 상태 또는 `INSUFFICIENT_DATA` 상태에 있을 때 알립니다.

    경보가 동일한 경보 상태 또는 다른 경보 상태에 대해 여러 개의 알림을 보내도록 설정하려면 알림 추가를 선택합니다.

    경보에서 알림을 보내지 않게 하려면 제거(Remove)를 선택합니다.
11. 작업 구성(Configure actions) 페이지의 다른 섹션은 비워둘 수 있습니다. 다른 섹션을 비워 두면 조정 정책에 연결하지 않고도 경보가 생성됩니다. 그런 다음 Amazon EC2 Auto Scaling 콘솔에서 조정 정책과 경보를 연결할 수 있습니다.
12. Next(다음)를 선택합니다.
13. 경보의 이름(예: `Step-Scaling-AlarmLow-RemoveCapacity`)과 설명(옵션)을 입력하고 다음(Next)을 선택합니다.
14. 경보 생성(Create alarm)을 선택하십시오.

다음 절차를 사용하여 CloudWatch 경보를 생성한 후 중단한 부분부터 계속 진행하십시오.

2단계: 확장을 위한 단계별 규모 조정 정책 생성

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 스케일링 제한이 적절하게 설정되었는지 확인합니다. 예를 들어 그룹의 원하는 용량이 이미 최소 용량인 경우 규모를 늘리려면 최소 용량을 새로 지정해야 합니다. 자세한 내용은 Auto Scaling 그룹에 대한 스케일링 제한 설정 단원을 참조하십시오.
4. 자동 크기 조정(Automatic scaling) 탭의 동적 크기 조정 정책(Dynamic scaling policies)에서 동적 크기 조정 정책 생성(Create dynamic scaling policy)을 선택합니다.
5. 정책 유형에서 Step scaling을 선택한 다음 정책의 이름을 지정합니다.
6. CloudWatch 경보의 경우 경보를 선택합니다. 아직 알람을 생성하지 않은 경우 알람 생성을 선택하고 이전 절차의 4단계부터 14단계까지 완료하여 알람을 생성합니다. CloudWatch
7. 다음 작업을 수행(Take the action)을 사용하여 이 정책을 실행할 때 적용할 현재 그룹 크기 변경을 지정합니다. 특정 인스턴스 개수 또는 기존 그룹 크기의 백분율을 제거하거나 그룹을 정확한 크기로 설정할 수 있습니다.

   예를 들어 그룹 용량을 두 인스턴스만큼 줄이는 스케일 인 정책을 생성하려면 을 선택하고 `Remove` 다음 필드에 를 입력한 `2` 다음 선택하십시오`capacity units`. 기본적으로 이 단계별 조정의 상한값은 경보 임계값이고 하한값은 음(-)의 무한대입니다.
8. 다른 단계를 추가하려면 단계 추가(Add step)를 선택한 다음 조정할 양과 경보 임계값에 상대적인 단계 하한 및 상한을 정의합니다.
9. 생성(Create)을 선택합니다.

AWS CLI

스케일 인 (용량 감소) 을 위한 단계별 조정 정책을 생성하려면 다음 예제 명령을 사용할 수 있습니다. 각각 교체하십시오.*`user input placeholder`* 자신의 정보로

를 사용할 때는 먼저 지표 값이 감소할 때 규모를 조정하는 방법에 대한 지침을 Amazon EC2 Auto Scaling에 제공하는 단계별 조정 정책을 생성합니다. AWS CLI그런 다음 관찰할 지표를 식별하고, 지표의 하한 임계값 및 경보에 대한 기타 세부 정보를 정의하고, 경보를 조정 정책과 연결하여 경보를 생성합니다.

1단계: 규모 확대를 위한 정책 생성

다음 put-scaling-policy명령을 사용하여 관련 CloudWatch 경보가 메트릭 하한 임계값을 위반할 경우 그룹 용량을 인스턴스 2개까지 줄이는 조정 유형이 포함된 단계 조정 정책을 생성하십시오. `my-step-scale-in-policy` `ChangeInCapacity`

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. 정책에 대한 CloudWatch 경보를 생성하는 데 필요합니다.

```
{
    "PolicyARN": "arn:aws:autoscaling:region:123456789012:scalingPolicy:ac542982-cbeb-4294-891c-a5a941dfa787:autoScalingGroupName/my-asg:policyName/my-step-scale-out-policy
}
```

2단계: 지표 하한 임계값에 대한 CloudWatch 경보 생성

다음 CloudWatch put-metric-alarm명령을 사용하여 최소 2분의 연속 평가 기간 동안 평균 CPU 임계값인 40% 를 기준으로 Auto Scaling 그룹의 크기를 줄이는 경보를 생성합니다. 사용자 지정 지표를 사용하려면 `--metric-name`에 지표 이름을 지정하고, `--namespace`에 네임스페이스를 지정합니다.

```
aws cloudwatch put-metric-alarm --alarm-name Step-Scaling-AlarmLow-RemoveCapacity \
  --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average \
  --period 120 --evaluation-periods 2 --threshold 40 \
  --comparison-operator LessThanOrEqualToThreshold \
  --dimensions "Name=AutoScalingGroupName,Value=my-asg" \
  --alarm-actions PolicyARN
```

단순 조정 정책

다음 예는 CLI 명령을 사용하여 간단한 조정 정책을 생성하는 방법을 보여줍니다. 이러한 지침은 사용하려는 모든 고객을 위한 참고 자료로 이 문서에 남아 있지만, 대신 대상 추적 또는 단계 조정 정책을 사용하는 것이 좋습니다.

단계별 조정 정책과 마찬가지로 단순 조정 정책도 조정 정책에 대한 CloudWatch 경보를 생성해야 합니다. 생성하는 정책에서 인스턴스 추가 또는 제거 여부와 인스턴스 수를 정의하거나 그룹을 정확한 크기로 설정해야 합니다.

단계별 조정 정책과 단순 조정 정책의 주요 차이점 중 하나는 단계별 조정 정책을 통해 얻을 수 있는 단계 조정입니다. 단계별 조정을 사용하면 지정한 단계 조정에 따라 그룹 크기를 더 크게 또는 더 작게 변경할 수 있습니다.

또한 단순 조정 정책은 진행 중인 조정 활동 또는 상태 점검 교체가 완료되고 휴지 기간이 종료될 때까지 기다려야 하며 추가 경보에 응답하기 전에 휴지 기간이 종료되어야 합니다. 반면, 단계별 크기 조정에서는 조정 활동 또는 상태 점검 교체가 진행 중인 동안에도 정책이 추가 경보에 계속 응답합니다. 즉, Amazon EC2 Auto Scaling은 경보 메시지를 수신할 때 모든 경보 위반을 평가합니다. 따라서 조정 조정이 한 번뿐이더라도 단계 조정 정책을 대신 사용하는 것이 좋습니다.

Amazon EC2 Auto Scaling은 원래 단순 규모 조정 정책만 지원했습니다. 목표 추적 및 단계별 조정 정책이 도입되기 전에 조정 정책을 생성한 경우 해당 정책은 단순 조정 정책으로 취급됩니다.

스케일 아웃을 위한 간단한 규모 조정 정책을 생성하세요.

다음 put-scaling-policy명령을 사용하여 관련 CloudWatch 경보가 지표의 `PercentChangeInCapacity` 상한 임계값을 위반할 경우 그룹 용량을 30% 늘리는 조정 유형이 포함된 단순 조정 정책을 생성합니다. `my-simple-scale-out-policy`

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. 정책에 대한 CloudWatch 경보를 생성하는 데 필요합니다.

규모 확대를 위한 간단한 규모 조정 정책을 생성하세요.

다음 put-scaling-policy명령을 사용하여 관련 CloudWatch 경보가 지표의 하한 임계값을 위반할 경우 그룹 용량을 한 인스턴스씩 줄이는 조정 유형을 `ChangeInCapacity` 포함하는 단순 조정 정책을 생성합니다. `my-simple-scale-in-policy`

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. 정책에 대한 CloudWatch 경보를 생성하는 데 필요합니다.

Amazon EC2 Auto Scaling의 스케일링 쿨다운

> **Important**
>
> 모범 사례로 간단한 조정 정책 및 조정 휴지를 사용하지 않는 것이 좋습니다. 목표 추적 조정 정책 또는 단계별 조정 정책이 성능 조정에 더 좋습니다. 조정 지표의 값이 감소하거나 증가함에 따라 Auto Scaling 그룹 크기를 비례적으로 변경하는 조정 정책의 경우, 단순 조정 또는 단계별 조정보다 대상 추적을 사용하는 것이 좋습니다.

Auto Scaling 그룹에 대한 간단한 스케일링 정책을 만들 때는 스케일링 쿨다운을 동시에 구성하는 것이 좋습니다.

Auto Scaling 그룹은 인스턴스를 시작하거나 해지한 후 단순 조정 정책에 의해 시작된 추가 조정 활동이 시작되기 전에 냉각 기간이 끝날 때까지 기다립니다. 냉각 기간의 목적은 Auto Scaling 그룹이 안정되도록 하고 이전 조정 활동의 효과가 가시화되기 전에 추가 인스턴스가 시작되거나 해지되는 것을 방지하는 것입니다.

예를 들어 CPU 사용률에 대한 간단한 조정 정책에서 두 개의 인스턴스를 시작하도록 권장한다고 가정해 보겠습니다. Amazon EC2 Auto Scaling은 두 개의 인스턴스를 시작한 다음 휴지 기간이 끝날 때까지 조정 활동을 일시 중지합니다. 냉각 기간이 끝나면 단순 조정 정책에 의해 시작된 모든 조정 활동이 재개될 수 있습니다. CPU사용률이 경보 상한 임계값을 다시 초과하면 Auto Scaling 그룹이 다시 축소되고 휴지 기간이 다시 적용됩니다. 그러나 두 개의 인스턴스로 지표 값을 다시 낮추기에 충분하면 그룹은 현재 크기로 유지됩니다.

내용

- 고려 사항
- 수명 주기 후크는 추가적 지연을 야기할 수 있습니다.
- 기본 냉각 기간 변경
- 특정 단순 조정 정책에 대한 냉각 기간 설정

고려 사항

단순 조정 정책 및 조정 휴지 작업 시 다음 고려 사항이 적용됩니다.

- 대상 추적 및 단계별 조정 정책은 냉각 기간이 끝날 때까지 기다리지 않고 즉시 스케일 아웃 활동을 시작할 수 있습니다. 대신 Auto Scaling 그룹에서 인스턴스를 시작할 때마다 개별 인스턴스에 준비 기간이 있습니다. 자세한 내용은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 단원을 참조하십시오.
- 예약된 작업이 예약된 시간에 시작되면 냉각 기간이 끝날 때까지 기다리지 않고 즉시 조정 활동을 시작할 수도 있습니다.
- 인스턴스가 비정상 상태가 되면 Amazon EC2 Auto Scaling은 휴지 기간이 끝날 때까지 기다리지 않고 비정상 인스턴스를 교체합니다.
- 여러 인스턴스가 시작되거나 해지될 때 냉각 기간(기본 휴지 또는 조정 정책별 휴지)은 마지막 인스턴스가 시작 또는 해지될 때부터 적용됩니다.
- Auto Scaling 그룹을 수동으로 스케일 아웃할 때 기본값은 냉각 기간이 끝날 때까지 기다리지 않는 것입니다. 하지만 AWS CLI 또는 an을 사용하여 수동으로 확장할 때는 이 동작을 무시하고 기본 휴지 시간을 적용할 수 있습니다. SDK
- 기본적으로 Elastic Load Balancing은 등록 취소(Connection Draining) 프로세스를 완료하기 위해 300초 동안 기다립니다. 그룹이 Elastic Load Balancing 로드 밸런서 뒤에 있는 경우, 냉각 기간을 시작하기 전에 해지 인스턴스가 등록 취소될 때까지 기다립니다.

수명 주기 후크는 추가적 지연을 야기할 수 있습니다.

수명 주기 후크가 호출되면 수명 주기 작업을 완료한 후 또는 제한 시간이 끝난 후 냉각 기간이 시작됩니다. 예컨대, Auto Scaling 그룹에 인스턴스 시작을 위한 수명 주기 후크가 있다고 가정해 봅시다. 애플리케이션에 수요가 증가하면 그룹은 인스턴스를 시작하여 용량을 추가합니다. 수명 주기 후크가 있기 때문에 인스턴스는 대기 상태가 되고 단순 조정 정책으로 인한 크기 조정 활동은 일시 중지됩니다. 인스턴스가 `InService` 상태로 들어가면 냉각 기간이 시작됩니다. 냉각 기간이 끝나면 단순 조정 정책 활동이 재개됩니다.

Elastic Load Balancing이 활성화되면 확장을 위해 종료하도록 선택한 인스턴스가 연결 드레이닝을 시작할 때 (등록 취소 지연) 휴지 기간이 시작됩니다. 휴지 기간은 연결 드레이닝이 완료되거나 라이프사이클 후크가 해당 작업을 완료할 때까지 기다리지 않습니다. 즉, 단순한 규모 조정 정책으로 인한 모든 조정 활동은 규모 조정 이벤트의 결과가 그룹 용량에 반영되는 즉시 재개될 수 있습니다. 그러지 않고 Connection Draining, 수명 주기 후크 및 냉각 기간, 이렇게 세 가지 활동이 모두 완료될 때까지 대기하면 Auto Scaling 그룹에서 크기 조정을 일시 중지하는 데 필요한 시간이 크게 늘어납니다.

기본 냉각 기간 변경

Amazon EC2 Auto Scaling 콘솔에서 Auto Scaling 그룹을 처음 생성할 때는 기본 휴지 시간을 설정할 수 없습니다. 기본적으로 냉각 기간은 300초(5분)로 설정됩니다. 필요한 경우, 그룹이 생성된 후 이를 업데이트할 수 있습니다.

기본 냉각 기간 변경(콘솔)

Auto Scaling 그룹을 생성한 후 Details(세부 정보) 탭에서 Advanced configurations(고급 구성), Edit(편집)을 선택합니다. 기본 냉각 기간(Default cooldown)에서 인스턴스 시작 시간 또는 기타 애플리케이션 요건에 따라 원하는 시간을 선택합니다.

기본 냉각 기간 변경(AWS CLI)

다음 명령을 사용하여 신규 또는 기존 Auto Scaling 그룹의 기본 냉각 기간을 변경합니다. 기본 냉각 기간이 정의되지 않은 경우, 기본값인 300초가 사용됩니다.

- create-auto-scaling-group
- update-auto-scaling-group

기본 재사용 대기시간 값을 확인하려면 명령을 사용하십시오. describe-auto-scaling-groups

특정 단순 조정 정책에 대한 냉각 기간 설정

기본적으로 모든 단순 조정 정책은 Auto Scaling 그룹에 대해 정의된 기본 냉각 기간을 사용합니다. 특정 단순 조정 정책에 대한 냉각 기간을 설정하려면 정책을 생성하거나 업데이트할 때 선택적 휴지 파라미터를 사용합니다. 정책에 대한 냉각 기간이 지정되면 기본 냉각 기간이 재정의됩니다.

조정 정책별 휴지 기간의 일반적인 용도 중 하나는 정책 축소를 사용하는 것입니다. 이 정책은 인스턴스를 종료하므로 Amazon EC2 Auto Scaling은 추가 인스턴스를 종료할지 여부를 결정하는 데 더 적은 시간이 소요됩니다. 인스턴스를 해지하는 것은 인스턴스를 시작하는 것보다 훨씬 빠른 작업이어야 합니다. 따라서 기본 냉각 기간인 300초가 너무 깁니다. 이 경우 규모 조정 정책별 휴지 기간을 두고 축소 정책 값을 낮추면 그룹을 더 빠르게 확장할 수 있어 비용을 절감하는 데 도움이 될 수 있습니다.

콘솔에서 단순 조정 정책을 생성하거나 업데이트하려면 그룹을 생성한 후 자동 조정(Automatic scaling) 탭을 선택합니다. 를 사용하여 간단한 조정 정책을 생성하거나 업데이트하려면 명령을 사용하십시오 AWS CLI. put-scaling-policy 자세한 내용은 단계별 조정 및 단순 조정 정책 단원을 참조하십시오.

아마존 기반 규모 조정 정책 SQS

> **Important**
>
> 다음 정보와 단계는 사용자 지정 지표로 게시하기 전에 SQS `ApproximateNumberOfMessages` queue 속성을 사용하여 인스턴스당 Amazon 대기열 백로그를 계산하는 방법을 보여줍니다. CloudWatch 하지만 이제 지표 수학을 사용하여 자체 지표를 게시하는 데 드는 비용과 노력을 줄일 수 있습니다. 자세한 내용은 지표 수학을 사용하여 목표 추적 조정 정책을 생성하십시오. 단원을 참조하십시오.

이 섹션에서는 Amazon 심플 큐 서비스 (AmazonSQS) 대기열의 시스템 부하 변화에 따라 Auto Scaling 그룹을 조정하는 방법을 보여줍니다. Amazon을 사용하는 방법에 대해 자세히 SQS 알아보려면 Amazon 심플 큐 서비스 개발자 안내서를 참조하십시오.

Amazon SQS 대기열의 활동에 따라 규모를 조정하는 방안을 고려할 수 있는 몇 가지 시나리오가 있습니다. 사용자가 이미지를 업로드하고 그러한 이미지를 온라인으로 사용하는 웹 앱을 예로 들어 보겠습니다. 이 시나리오에서 각 이미지를 게시하려면 먼저 크기를 변경하고 인코딩해야 합니다. 앱은 Auto Scaling 그룹의 EC2 인스턴스에서 실행되며 일반적인 업로드 속도를 처리하도록 구성되어 있습니다. 인스턴스 수준을 항상 최신으로 유지하기 위해 비건전 인스턴스는 해지되고 교체됩니다. 앱은 이미지의 원시 비트맵 데이터를 처리를 위해 SQS 대기열에 넣습니다. 그리고 이미지를 처리한 후, 사용자가 볼 수 있는 처리된 이미지를 게시합니다. 이 시나리오의 아키텍처는 이미지 업로드 수가 항상 일정할 경우, 잘 작동합니다. 그러나 시간이 지남에 따라 업로드 수가 변경되는 경우, 동적 조정을 사용하여 Auto Scaling 그룹 용량의 조정을 고려할 수 있습니다.

내용

- 올바른 지표에 대상 추적 사용
- 제한 사항 및 사전 요건
- Amazon을 기반으로 스케일링을 구성합니다. SQS
- Amazon SQS 및 인스턴스 스케일 인 보호

올바른 지표에 대상 추적 사용

사용자 지정 Amazon SQS 대기열 지표를 기반으로 하는 대상 추적 조정 정책을 사용하는 경우 동적 규모 조정을 애플리케이션의 수요 곡선에 더 효과적으로 맞출 수 있습니다. 대상 추적에 대한 지표 선택에 대한 자세한 설명은 지표 선택 섹션을 참조하세요.

`ApproximateNumberOfMessagesVisible`대상 추적과 같은 CloudWatch Amazon SQS 지표를 사용할 때의 문제는 대기열의 메시지 수가 대기열의 메시지를 처리하는 Auto Scaling 그룹의 크기에 비례하여 변경되지 않을 수 있다는 것입니다. SQS대기열에 있는 메시지 수가 필요한 인스턴스의 수만을 정의하는 것은 아니기 때문입니다. 하지만 Auto Scaling 그룹의 인스턴스 수는 메시지를 처리하는 데 걸리는 시간이나 허용되는 지연 시간(대기열 지연) 등과 같은 여러 요인의 영향을 받을 수 있습니다.

해결 방법은 인스턴스당 백로그 지표와 목표값을 사용하여 인스턴스당 허용 백로그를 유지하는 것입니다. 이 값은 다음과 같이 계산할 수 있습니다.

- 인스턴스당 백로그: 인스턴스당 백로그를 계산하려면 `ApproximateNumberOfMessages` queue 속성으로 시작하여 대기열의 길이 (SQS대기열에서 검색할 수 있는 메시지 수) 를 결정합니다. 이 값을 플릿의 실행 용량(Auto Scaling 그룹에서 상태가 `InService`인 인스턴스 수)으로 나눠서 인스턴스당 백로그를 산출합니다.
- 인스턴스당 허용되는 백로그: 목표값을 계산하려면 먼저 애플리케이션에서 허용할 수 있는 지연 시간을 확인하세요. 그런 다음, 허용 가능한 지연 시간 값을 EC2 인스턴스가 메시지를 처리하는 데 걸리는 평균 시간으로 나눕니다.

예컨대, 현재 인스턴스 10개로 구성된 Auto Scaling 그룹이 있고, 대기열 (`ApproximateNumberOfMessages`)에 표시되는 메시지 수가 1,500개라고 가정해 보겠습니다. 각 메시지의 평균 처리 시간이 0.1초이고 가장 긴 허용 지연 시간이 10초인 경우, 인스턴스당 허용 가능한 백로그는 10/0.1인 100개의 메시지입니다. 즉, 목표 추적 정책의 목표값은 100입니다. 인스턴스당 백로그가 목표 값에 도달하면 스케일 아웃 이벤트가 발생합니다. 인스턴스당 백로그가 이미 150개의 메시지(1,500개의 메시지/10개의 인스턴스)이므로 그룹이 스케일 아웃되고 5개의 인스턴스를 추가하여 목표값 백분율을 유지합니다.

다음 절차에서는 사용자 지정 지표를 게시하고 이러한 계산에 따라 Auto Scaling 그룹을 조정하도록 구성하는 대상 추적 조정 정책을 생성하는 방법을 보여줍니다.

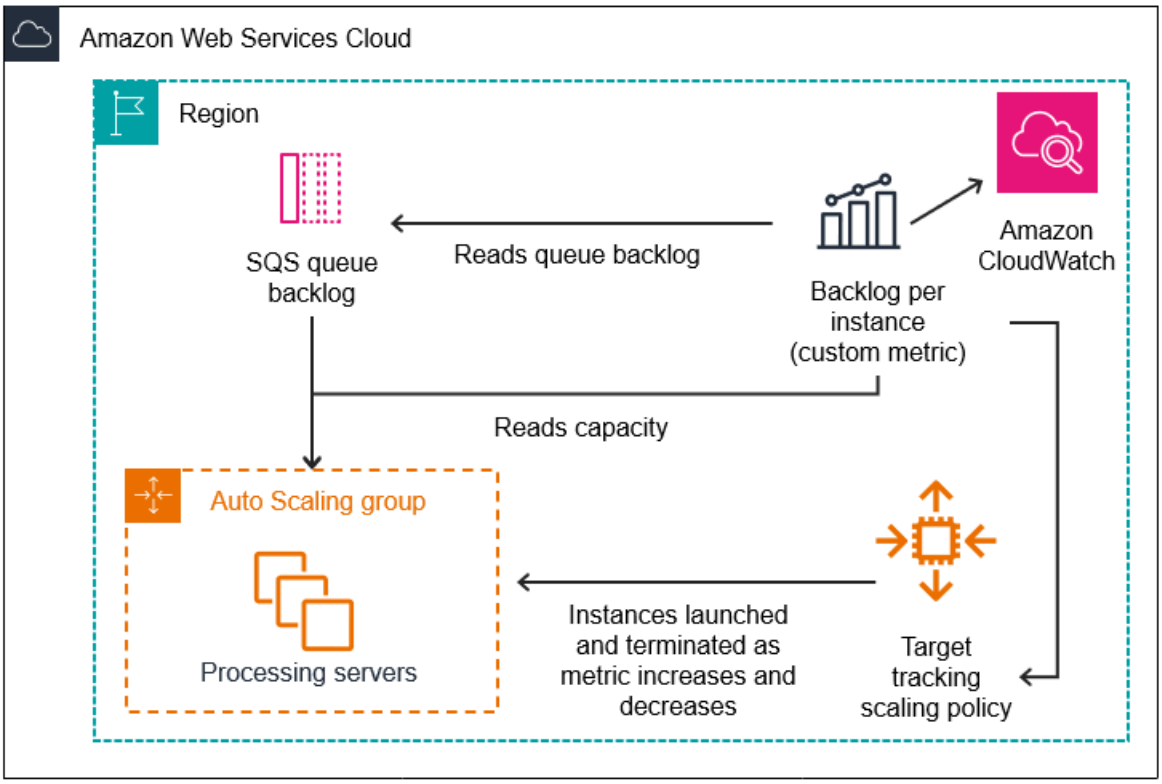
> **Important**
>
> 비용을 절감하기 위해서는 지표 수학을 사용해야 한다는 점을 기억하세요. 자세한 설명은 지표 수학을 사용하여 목표 추적 조정 정책을 생성하십시오. 섹션을 참조하세요.

이러한 구성을 위해 세 가지 방법을 사용할 수 있습니다.

- SQS대기열의 메시지를 처리하기 위해 EC2 인스턴스를 관리하는 Auto Scaling 그룹입니다.

- Auto Scaling 그룹의 EC2 인스턴스당 대기열에 CloudWatch 있는 메시지 수를 측정하는 사용자 지정 지표로, Amazon에 전송합니다.
- 사용자 지정 지표와 설정된 목표 값을 기반으로 Auto Scaling 그룹이 확장되도록 구성하는 대상 추적 정책입니다. CloudWatch 경보는 조정 정책을 호출합니다.

다음 다이어그램은 이 구성의 아키텍처를 보여 줍니다.



제한 사항 및 사전 요건

이 구성을 사용하려면 다음과 같은 제한 사항을 숙지해야 합니다.

- AWS CLI 또는 an을 사용하여 사용자 지정 지표를 SDK 게시해야 합니다. CloudWatch 그런 다음 를 사용하여 지표를 모니터링할 수 AWS Management Console있습니다.
- Amazon EC2 Auto Scaling 콘솔은 사용자 지정 지표를 사용하는 대상 추적 조정 정책을 지원하지 않습니다. 조정 정책에 대한 사용자 지정 지표를 SDK 지정하려면 AWS CLI 또는 an을 사용해야 합니다.

다음 섹션에서는 수행해야 하는 AWS CLI 작업에 를 사용하도록 안내합니다. 예를 들어, 큐의 현재 사용을 반영하는 지표 데이터를 가져오려면 SQS get-queue-attributes명령을 사용합니다. 이 (가) CLI 설치 및 구성되어 있는지 확인하십시오.

시작하기 전에 사용할 Amazon SQS 대기열이 있어야 합니다. 다음 섹션에서는 대기열 (표준 또는 FIFO), Auto Scaling 그룹 및 해당 대기열을 사용하는 애플리케이션을 실행하는 EC2 인스턴스가 이미 있다고 가정합니다. Amazon에 대한 자세한 내용은 Amazon SQS 심플 큐 서비스 개발자 안내서를 참조하십시오.

Amazon을 기반으로 스케일링을 구성합니다. SQS

이 섹션에서는 Amazon Amazon을 기반으로 규모 조정을 구성하는 방법을 설명합니다SQS.

Tasks

- 1단계: CloudWatch 사용자 지정 지표 생성
- 2단계: 대상 추적 조정 정책 생성
- 3단계: 조정 정책 테스트

1단계: CloudWatch 사용자 지정 지표 생성

사용자 지정 지표는 지표 이름과 선택한 네임스페이스를 사용하여 정의됩니다. 사용자 지정 지표를 위한 네임스페이스는 `AWS/`로 시작할 수 없습니다. 사용자 지정 지표 게시에 대한 자세한 내용은 Amazon CloudWatch User Guide의 사용자 지정 지표 게시 주제를 참조하십시오.

이 절차에 따라 먼저 AWS 계정에서 정보를 읽고 사용자 지정 지표를 생성하십시오. 그런 다음, 앞 섹션에서 권장한 인스턴스당 백로그 지표를 계산합니다. 마지막으로, 이 수치를 1분 단위로 CloudWatch 세분화하여 게시하십시오. 가능하면 시스템 로드 변화에 빠르게 대응할 수 있도록 1분 단위로 지표를 조정하는 것이 좋습니다.

CloudWatch 사용자 지정 지표를 만들려면 ()AWS CLI

1. SQSget-queue-attributes명령을 사용하여 대기열에서 대기 중인 메시지 수를 가져옵니다 (`ApproximateNumberOfMessages`).

```
aws sqs get-queue-attributes --queue-url https://
sqs.region.amazonaws.com/123456789/MyQueue \
  --attribute-names ApproximateNumberOfMessages
```

2. describe-auto-scaling-groups명령을 사용하여 그룹의 실행 용량 (`InService`수명 주기 상태의 인스턴스 수) 을 가져옵니다. 이 명령은 Auto Scaling 그룹의 인스턴스와 해당 수명 주기 상태를 반환합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

3. 대기열에서 검색할 수 있는 대략적인 메시지 수를 그룹의 실행 용량으로 나누어 인스턴스당 백로그를 계산합니다.
4. 1분마다 실행되는 스크립트를 생성하여 인스턴스당 백로그 값을 검색하고 이를 CloudWatch 사용자 지정 지표에 게시하십시오. 사용자 지정 지표를 게시할 때 지표의 이름, 네임스페이스, 단위, 값 및 0개 이상의 차원을 지정합니다. 차원은 차원 이름과 차원 값으로 구성됩니다.

   사용자 지정 지표를 게시하려면 자리 표시자 값을 다음으로 바꾸십시오.*italics* 선호하는 지표 이름, 지표 값, 네임스페이스 ("`AWS`“로 시작하지 않는 한) 및 차원 (선택 사항) 을 입력하고 다음 명령을 실행합니다. put-metric-data

```
aws cloudwatch put-metric-data --metric-name MyBacklogPerInstance --
namespace MyNamespace \
  --unit None --value 20 --
dimensions MyOptionalMetricDimensionName=MyOptionalMetricDimensionValue
```

애플리케이션에서 원하는 지표를 내보낸 후 데이터가 로 전송됩니다. CloudWatch 지표는 CloudWatch 콘솔에서 볼 수 있습니다. 에 AWS Management Console 로그인하고 CloudWatch 페이지로 이동하면 액세스할 수 있습니다. 그런 다음 지표 페이지로 이동하거나 검색 상자에서 검색하여 지표를 볼 수 있습니다. 지표 보기에 대한 자세한 내용은 Amazon 사용 CloudWatch 설명서의 사용 가능한 지표 보기를 참조하십시오.

2단계: 대상 추적 조정 정책 생성

이제 생성한 지표를 대상 추적 스케일링 정책에 추가할 수 있습니다.

대상 추적 조정 정책(AWS CLI)을 생성하려면

1. 다음 `cat` 명령을 사용하여 조정 정책의 목표 값과 사용자 지정된 지표 사양을 홈 `config.json` 디렉터리에 이름이 지정된 JSON 파일에 저장합니다. 각각 바꾸십시오.*user input placeholder* 자신의 정보로 `TargetValue`에, 인스턴스당 허용 백로그 지표를 계산하여 입력합니다. 이 값을 계산하려면 이전 섹션에서 설명한 대로 보통 지연 시간 값을 확인하여, 이 값을 메시지를 처리하는 데 걸리는 평균 시간으로 나눕니다.

1단계에서 만든 지표에 대해 어떤 차원도 지정하지 않았다면 사용자 지정 지표 사양에 어떤 차원도 포함하지 마세요.

```
$ cat ~/config.json
{
   "TargetValue":100,
   "CustomizedMetricSpecification":{
      "MetricName":"MyBacklogPerInstance",
      "Namespace":"MyNamespace",
      "Dimensions":[
         {
            "Name":"MyOptionalMetricDimensionName",
            "Value":"MyOptionalMetricDimensionValue"
         }
      ],
      "Statistic":"Average",
      "Unit":"None"
   }
}
```

2. 이전 단계에서 생성한 `config.json` 파일과 함께 put-scaling-policy명령을 사용하여 조정 정책을 생성합니다.

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://~/config.json
```

이 정책은 스케일 아웃과 축소에 대해 한 개씩 두 개의 경보를 생성합니다. 또한 등록된 정책의 Amazon Resource Name (ARN) 을 반환하며 CloudWatch, 이 이름을 CloudWatch 사용하여 지표 임계값이 위반될 때마다 크기 조정을 호출합니다.

3단계: 조정 정책 테스트

설정을 마친 후, 조정 정책이 작동하는지 확인합니다. SQS대기열의 메시지 수를 늘리고 Auto Scaling 그룹에서 추가 EC2 인스턴스를 시작했는지 확인하여 테스트할 수 있습니다. SQS대기열의 메시지 수를 줄인 다음 Auto Scaling 그룹이 인스턴스를 종료했는지 확인하여 테스트할 수도 있습니다. EC2

스케일 아웃 기능을 테스트하려면

1. Amazon SQS 표준 대기열 생성 및 메시지 전송 또는 Amazon SQS FIFO 대기열 생성 및 메시지 전송의 단계에 따라 대기열에 메시지를 추가합니다. 인스턴스당 백로그 지표가 목표값을 초과하도록 대기열의 메시지 수를 늘려야 합니다.

   변경 사항에 따라 경보를 호출하기까지 몇 분 정도 걸릴 수 있습니다.
2. describe-auto-scaling-groups명령을 사용하여 그룹에서 인스턴스를 시작했는지 확인합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

스케일 인 함수를 테스트하려면

1. 메시지 수신 및 삭제 (콘솔) 의 단계에 따라 대기열에서 메시지를 삭제합니다. 인스턴스당 백로그 지표가 목표값 미만이 되도록 대기열의 메시지 수를 줄여야 합니다.

   변경 사항에 따라 경보를 호출하기까지 몇 분 정도 걸릴 수 있습니다.
2. describe-auto-scaling-groups명령을 사용하여 그룹에서 인스턴스가 종료되었는지 확인합니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

Amazon SQS 및 인스턴스 스케일 인 보호

인스턴스가 종료될 때 처리되지 않은 메시지는 SQS 대기열로 반환되어 아직 실행 중인 다른 인스턴스에서 처리할 수 있습니다. 오래 실행 중인 작업을 수행하는 애플리케이션의 경우, 인스턴스 스케일 인 방비를 선택적으로 사용하여 Auto Scaling 그룹이 축소될 때 어떤 대기열 작업자를 해지할지 제어할 수 있습니다.

다음 유사 코드는 오래 실행되는 대기열 기반 작업자 프로세스가 축소 시 해지되지 않도록 방지하는 한 가지 방법을 보여 줍니다.

```
while (true)
{
  SetInstanceProtection(False);
  Work = GetNextWorkUnit();
  SetInstanceProtection(True);
  ProcessWorkUnit(Work);
  SetInstanceProtection(False);
```

```
}
```

자세한 설명은 인스턴스 종료를 정상적으로 처리하도록 애플리케이션을 설계하십시오. 섹션을 참조하세요.

Auto Scaling 그룹에 대한 크기 조정 활동 확인

Amazon EC2 콘솔의 Amazon EC2 Auto Scaling 섹션에서 Auto Scaling 그룹의 활동 기록을 통해 현재 진행 중인 조정 활동의 현재 상태를 확인할 수 있습니다. 크기 조정 활동이 완료되면 성공 여부를 확인할 수 있습니다. 이는 Auto Scaling 그룹을 생성하거나 기존 그룹에 조정 조건을 추가할 때 특히 유용합니다.

Auto Scaling 그룹에 대상 추적, 단계별 또는 단순 조정 정책을 추가하면 Amazon EC2 Auto Scaling에서 즉시 해당 지표에 대한 정책 평가를 시작합니다. 메트릭 경보는 지표가 지정된 평가 기간 수의 임계값을 위반하면 ALARM 상태가 됩니다. 이는 크기 조정 정책이 생성된 직후 크기 조정 활동으로 이어질 수 있음을 의미합니다. Amazon EC2 Auto Scaling이 조정 정책에 따라 원하는 용량을 조정한 후 계정의 조정 활동을 확인할 수 있습니다. Amazon EC2 Auto Scaling으로부터 조정 활동에 대해 알려주는 이메일 알림을 받으려면 의 아마존 EC2 오토 스케일링에 대한 아마존 SNS 알림 옵션 지침을 따르십시오.

> **Tip**
> 다음 절차에서는 Auto Scaling 그룹에 대한 Activity history(활동 기록) 및 Instances(인스턴스) 섹션을 살펴봅니다. 두 섹션 모두에 명명된 열이 이미 표시되어 있어야 합니다. 숨겨진 열을 표시하거나 표시되는 행 수를 변경하려면 각 섹션의 오른쪽 상단에 있는 톱니바퀴 아이콘을 선택하여 기본 설정 모달을 열고 필요에 따라 설정을 업데이트한 다음 확인(Confirm)을 선택합니다.

Auto Scaling 그룹의 크기 조정 활동 보기(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 화면 상단의 탐색 모음에서 Auto Scaling 그룹이 들어있는 지역을 선택합니다.
3. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
4. Activity(활동) 탭에서 Activity history(활동 기록)의 Status(상태) 열에 Auto Scaling 그룹이 성공적으로 인스턴스를 시작 또는 해지했는지와 크기 조정 활동이 아직 진행 중인지 여부가 표시됩니다.

5. (옵션) 크기 조정 활동이 많은 경우, 활동 기록 상단 가장자리의 > 아이콘을 선택하여 크기 조정 활동의 다음 페이지를 볼 수 있습니다.
6. 인스턴스 관리 탭의 인스턴스에서 수명 주기 열은 인스턴스의 상태를 포함합니다. 인스턴스가 시작되고 모든 수명 주기 후크가 완료되면 해당 수명 주기 상태가 `InService`로 변경됩니다. Health status 열에는 EC2 인스턴스의 인스턴스 상태 점검 결과가 표시됩니다.

Auto Scaling 그룹의 크기 조정 활동 보기(AWS CLI)

다음 describe-scaling-activities 명령을 사용합니다.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

출력의 예제는 다음과 같습니다.

크기 조정 활동은 시작 시간으로 정렬됩니다. 아직 진행 중인 활동이 먼저 설명됩니다.

```
{
  "Activities": [
    {
      "ActivityId": "5e3a1f47-2309-415c-bfd8-35aa06300799",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-06c4794c2499af1df",
      "Cause": "At 2020-02-11T18:34:10Z a monitor alarm TargetTracking-my-asg-AlarmLow-
b9376cab-18a7-4385-920c-dfa3f7783f82 in state ALARM triggered policy my-target-
tracking-policy changing the desired capacity from 3 to 2.  At 2020-02-11T18:34:31Z
 an instance was taken out of service in response to a difference between desired and
 actual capacity, shrinking the capacity from 3 to 2.  At 2020-02-11T18:34:31Z instance
 i-06c4794c2499af1df was selected for termination.",
      "StartTime": "2020-02-11T18:34:31.268Z",
      "EndTime": "2020-02-11T18:34:53Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-west-2a
\"...}",
      "AutoScalingGroupARN": "arn"
    },
...
  ]
}
```

출력의 필드에 대한 설명은 Amazon EC2 Auto Scaling API 참조의 활동을 참조하십시오.

삭제된 그룹에 대한 조정 활동 검색, 발생할 수 있는 오류 유형 및 이를 처리하는 방법에 대한 자세한 설명은 Amazon EC2 Auto Scaling의 문제 해결을 참조하세요.

Auto Scaling 그룹에 대한 조정 정책 비활성화

이 항목에서는 Auto Scaling 그룹에 포함된 인스턴스 수를 변경하지 않도록 조정 정책을 일시적으로 비활성화하는 방법을 설명합니다. 조정 정책을 비활성화하면 구성 세부 정보가 보존되므로 정책을 신속하게 다시 활성화할 수 있습니다. 이는 필요하지 않을 때 일시적으로 정책을 삭제했다가 나중에 다시 생성하는 것보다 쉽습니다.

조정 정책을 비활성화하면 조정 정책이 비활성화된 동안 위반된 지표 경보에 대해 Auto Scaling 그룹이 스케일 아웃 또는 축소되지 않습니다. 그러나 진행 중인 크기 조정 활동은 중지되지 않습니다.

비활성화된 조정 정책도 Auto Scaling 그룹에 추가할 수 있는 조정 정책 수 할당량에 포함됩니다.

조정 정책을 비활성화하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 자동 크기 조정(Automatic scaling) 탭의 동적 조정 정책(Dynamic scaling policies)에서 원하는 조정 정책의 오른쪽 상단에 있는 확인란을 선택합니다.
4. 동적 조정 정책(Dynamic scaling policies) 섹션의 맨 위로 스크롤하고 작업(Actions), 사용 중지(Disable)를 선택합니다.

조정 정책을 다시 활성화할 준비가 되면 이러한 단계를 반복한 다음 작업, 활성화를 선택합니다. 조정 정책을 다시 활성화한 후 현재 상태에 있는 경보가 있는 경우 Auto Scaling 그룹에서 즉시 조정 작업을 시작할 수 있습니다. ALARM

조정 정책을 비활성화하려면(AWS CLI)

다음과 같이 put-scaling-policy명령을 `--no-enabled` 옵션과 함께 사용하십시오. 정책을 생성할 때 지정하는 대로 명령에서 모든 옵션을 지정합니다.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
   --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
   --estimated-instance-warmup 360 \
```

```
  --target-tracking-configuration '{ "TargetValue": 70,
 "PredefinedMetricSpecification": { "PredefinedMetricType":
 "ASGAverageCPUUtilization" } }' \
  --no-enabled
```

조정 정책을 다시 활성화하려면(AWS CLI)

다음과 같이 put-scaling-policy명령을 `--enabled` 옵션과 함께 사용합니다. 정책을 생성할 때 지정하는 대로 명령에서 모든 옵션을 지정합니다.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
   --policy-name my-scaling-policy --policy-type TargetTrackingScaling \
   --estimated-instance-warmup 360 \
   --target-tracking-configuration '{ "TargetValue": 70,
 "PredefinedMetricSpecification": { "PredefinedMetricType":
 "ASGAverageCPUUtilization" } }' \
   --enabled
```

조정 정책을 설명하려면(AWS CLI)

describe-policies 명령을 사용하여 조정 정책의 활성화 상태를 확인합니다.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg \
   --policy-names my-scaling-policy
```

출력의 예제는 다음과 같습니다.

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "my-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:1d52783a-b03b-4710-
bb0e-549fd64378cc:autoScalingGroupName/my-asg:policyName/my-scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-9ca53fdd-7cf5-4223-938a-ae1199204502",
```

```
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-9ca53fdd-7cf5-4223-938a-
ae1199204502"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-7010c83d-d55a-4a7a-
abe0-1cf8b9de6d6c"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 70.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        }
    ]
}
```

Auto Scaling 그룹의 조정 정책 삭제

더 이상 필요 없는 조정 정책은 삭제할 수 있습니다. 조정 정책 유형에 따라 CloudWatch 경보를 삭제해야 할 수도 있습니다. 대상 추적 조정 정책을 삭제하면 관련 CloudWatch 경보도 모두 삭제됩니다. 단계별 조정 정책이나 단순 조정 정책을 삭제하면 기본 경보 작업이 삭제되지만 더 이상 관련 조치가 없더라도 CloudWatch 경보는 삭제되지 않습니다.

조정 정책을 삭제하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 자동 크기 조정(Automatic scaling) 탭의 동적 조정 정책(Dynamic scaling policies)에서 원하는 조정 정책의 오른쪽 상단에 있는 확인란을 선택합니다.

4. 동적 조정 정책(Dynamic scaling policies) 섹션의 맨 위로 스크롤하고 작업(Actions), 삭제(Delete)를 선택합니다.
5. 확인 메시지가 나타나면 예, 삭제합니다(Yes, Delete)를 선택합니다.
6. (선택 사항) 단계별 조정 정책 또는 단순 조정 정책을 삭제한 경우 다음과 같이 정책과 연결된 CloudWatch 경보를 삭제하십시오. 나중에 사용할 수 있도록 경보를 유지하려면 다음 하위 단계를 건너뛸 수 있습니다.
   a. 에서 CloudWatch 콘솔을 엽니다 https://console.aws.amazon.com/cloudwatch/.
   b. 탐색 창에서 Alarms(경보)를 선택합니다.
   c. 경보(예: `Step-Scaling-AlarmHigh-AddCapacity`)를 선택하고 작업, 삭제를 선택합니다.
   d. 확인 메시지가 나타나면 삭제를 선택합니다.

Auto Scaling 그룹에 대한 조정 정책 가져오기(AWS CLI)

조정 정책을 삭제하기 전에 describe-policies 명령을 사용하여 Auto Scaling 그룹에 대해 생성된 조정 정책을 확인합니다. 정책 및 CloudWatch 경보를 삭제할 때 출력을 사용할 수 있습니다.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
```

`--query` 파라미터를 사용하여 조정 정책 유형별로 결과를 필터링할 수 있습니다. `query`에 대한 이 구문은 Linux 또는 macOS에서만 작동합니다. Windows에서는 작은따옴표를 큰따옴표로 변경합니다.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg
  --query 'ScalingPolicies[?PolicyType==`TargetTrackingScaling`]'
```

출력의 예제는 다음과 같습니다.

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "PolicyName": "cpu50-target-tracking-scaling-policy",
        "PolicyARN": "PolicyARN",
        "PolicyType": "TargetTrackingScaling",
        "StepAdjustments": [],
        "Alarms": [
            {
```

```
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e",
                "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
            },
            {
                "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2",
                "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
            }
        ],
        "TargetTrackingConfiguration": {
            "PredefinedMetricSpecification": {
                "PredefinedMetricType": "ASGAverageCPUUtilization"
            },
            "TargetValue": 50.0,
            "DisableScaleIn": false
        },
        "Enabled": true
    }
]
```

조정 정책을 삭제하려면(AWS CLI)

다음 delete-policy 명령을 사용합니다.

```
aws autoscaling delete-policy --auto-scaling-group-name my-asg \
  --policy-name cpu50-target-tracking-scaling-policy
```

CloudWatch 경보를 삭제하려면 ()AWS CLI

단계별 및 단순 조정 정책의 경우 delete-alarms 명령을 사용하여 정책과 연결된 CloudWatch 경보를 삭제합니다. 나중에 경보를 사용하려면 이 단계를 건너뛸 수 있습니다. 한 번에 하나 이상 경보를 삭제할 수 있습니다. 예컨대, 다음 명령을 사용하여 `Step-Scaling-AlarmHigh-AddCapacity` 및 `Step-Scaling-AlarmLow-RemoveCapacity` 경보를 삭제합니다.

```
aws cloudwatch delete-alarms --alarm-name Step-Scaling-AlarmHigh-AddCapacity Step-
Scaling-AlarmLow-RemoveCapacity
```

에 대한 조정 정책 예시 AWS CLI

AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 를 통해 Amazon EC2 Auto Scaling에 대한 조정 정책을 생성할 수 SDKs 있습니다.

다음 예는 AWS CLI put-scaling-policy명령을 사용하여 Amazon EC2 Auto Scaling에 대한 조정 정책을 생성하는 방법을 보여줍니다. 각 항목을 교체하십시오.*user input placeholder* 자신의 정보로

를 사용하여 규모 조정 정책 작성을 시작하려면 대상 추적 조정 정책 및 단계별 조정 및 단순 조정 정책의 입문 연습을 참조하십시오. AWS CLI

예 1: 사전 정의된 지표 사양을 사용하여 대상 추적 조정 정책 적용

```
aws autoscaling put-scaling-policy --policy-name cpu50-target-tracking-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 50.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ASGAverageCPUUtilization"
  }
}
```

자세한 내용은 Amazon EC2 Auto Scaling API 레퍼런스를 참조하십시오 PredefinedMetricSpecification.

> **Note**
>
> 파일이 현재 디렉터리에 없는 경우 파일의 전체 경로를 입력합니다. 파일에서 AWS CLI 파라미터 값을 읽는 방법에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 파일에서 AWS CLI 파라미터 로드를 참조하십시오.

예 2: 사용자 지정된 지표 사양을 사용하여 대상 추적 조정 정책 적용

```
aws autoscaling put-scaling-policy --policy-name sqs100-target-tracking-scaling-policy
 \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 100.0,
```

```
  "CustomizedMetricSpecification": {
    "MetricName": "MyBacklogPerInstance",
    "Namespace": "MyNamespace",
    "Dimensions": [{
      "Name": "MyOptionalMetricDimensionName",
      "Value": "MyOptionalMetricDimensionValue"
    }],
    "Statistic": "Average",
    "Unit": "None"
  }
}
```

자세한 내용은 Amazon EC2 Auto Scaling API 레퍼런스를 참조하십시오 CustomizedMetricSpecification.

예 3: 스케일 아웃을 위한 대상 추적 조정 정책 적용

```
aws autoscaling put-scaling-policy --policy-name alb1000-target-tracking-scaling-policy
 \
  --auto-scaling-group-name my-asg --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
{
  "TargetValue": 1000.0,
  "PredefinedMetricSpecification": {
    "PredefinedMetricType": "ALBRequestCountPerTarget",
    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-
group/943f017f100becff"
  },
  "DisableScaleIn": true
}
```

예 4: 스케일 아웃을 위한 단계별 조정 정책 적용

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-out-policy \
  --policy-type StepScaling \
  --adjustment-type PercentChangeInCapacity \
  --metric-aggregation-type Average \
  --step-adjustments
 MetricIntervalLowerBound=10.0,MetricIntervalUpperBound=20.0,ScalingAdjustment=10 \

 MetricIntervalLowerBound=20.0,MetricIntervalUpperBound=30.0,ScalingAdjustment=20 \
```

```
                        MetricIntervalLowerBound=30.0,ScalingAdjustment=30 \
  --min-adjustment-magnitude 1
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. CloudWatch 경보를 생성할 ARN 때 필요합니다.

예 5: 축소를 위한 단계별 조정 정책 적용

```
aws autoscaling put-scaling-policy \
  --auto-scaling-group-name my-asg  \
  --policy-name my-step-scale-in-policy \
  --policy-type StepScaling \
  --adjustment-type ChangeInCapacity \
  --step-adjustments MetricIntervalUpperBound=0.0,ScalingAdjustment=-2
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. CloudWatch 경보를 생성할 ARN 때 필요합니다.

예 6: 스케일 아웃을 위한 단순 조정 정책 적용

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-out-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment 30 \
  --adjustment-type PercentChangeInCapacity --min-adjustment-magnitude 2
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. CloudWatch 경보를 생성할 ARN 때 필요합니다.

예 7: 축소를 위한 단순 조정 정책 적용

```
aws autoscaling put-scaling-policy --policy-name my-simple-scale-in-policy \
  --auto-scaling-group-name my-asg --scaling-adjustment -1 \
  --adjustment-type ChangeInCapacity --cooldown 180
```

정책의 Amazon 리소스 이름 (ARN) 을 기록해 둡니다. CloudWatch 경보를 생성할 ARN 때 필요합니다.

Amazon EC2 Auto Scaling을 위한 예측 스케일링

예측 스케일링은 과거 부하 데이터를 분석하여 트래픽 흐름의 일별 또는 주별 패턴을 감지하는 방식으로 작동합니다. Amazon Auto Scaling은 이 정보를 사용하여 향후 용량 수요를 예측하므로 Amazon EC2 Auto Scaling은 예상 부하에 맞춰 Auto Scaling 그룹의 용량을 사전에 늘릴 수 있습니다.

예측 조정은 다음과 같은 상황에 매우 적합합니다.

- 주기적 트래픽(예: 정규 업무 시간 동안 리소스 사용량이 많고 저녁 및 주말에는 리소스가 적게 사용되는 경우)
- 반복되는 on-and-off 워크로드 패턴 (예: 일괄 처리, 테스트 또는 주기적 데이터 분석)
- 초기화하는 데 시간이 오래 걸려 스케일 아웃 이벤트 중에 애플리케이션 성능에 눈에 띄는 지연 영향을 받는 애플리케이션

일반적으로 규칙적인 트래픽 증가 패턴과 초기화하는 데 시간이 오래 걸리는 애플리케이션이 있는 경우, 예측 조정 사용을 고려해야 합니다. 예측 조정을 사용하면 본질적으로 대응적인 동적 크기 조정만 사용하는 것에 비해 예상된 로드 전에 용량을 시작하여 더 빠르게 크기를 조정할 수 있습니다. 또한 예측 확장은 용량을 과도하게 프로비저닝할 필요가 없도록 함으로써 잠재적으로 비용을 절감할 수 있습니다. EC2

예컨대, 업무 시간 중에는 사용량이 많고 밤새 사용량이 줄어드는 애플리케이션을 생각해 보겠습니다. 각 영업일이 시작될 때 예측 조정은 트래픽이 처음 유입되기 전에 용량을 추가할 수 있습니다. 이를 통해 사용률이 낮은 기간에서 높은 기간으로 전환될 때 애플리케이션이 고가용성과 성능을 유지할 수 있습니다. 동적 조정이 변화하는 트래픽에 대응할 때까지 기다릴 필요가 없습니다. 또한 애플리케이션의 로드 패턴을 검토하고 예약된 스케일 아웃을 사용하여 적절한 용량을 예약하는 데 시간을 할애할 필요가 없습니다.

주제

- 예측 조정의 작동 방식
- Auto Scaling 그룹에 대한 예측 조정 정책 생성
- 예측적 조정 정책 평가
- 예약된 작업을 사용하여 예측 값 재정의
- 사용자 지정 지표를 사용하는 고급 예측 규모 조정 정책

예측 조정의 작동 방식

이 항목에서는 예측 규모 조정 작동 방식을 설명하고 예측 규모 조정 정책을 만들 때 고려해야 할 사항을 설명합니다.

주제

- 작동 방식

- 최대 용량 제한
- 고려 사항
- 지원되는 지역

작동 방식

예측 규모 조정을 사용하려면 모니터링 및 분석할 CloudWatch 지표를 지정하는 예측 규모 조정 정책을 생성하십시오. 예측 척도를 통해 미래 값을 예측하려면 이 지표에 최소 24시간 분량의 데이터가 있어야 합니다.

정책을 생성한 후 예측 스케일링은 패턴을 식별하기 위해 최대 지난 14일까지의 지표 데이터를 분석하기 시작합니다. 이 분석을 사용하여 향후 48시간 동안의 용량 요구 사항에 대한 시간별 예측을 생성합니다. 예측은 최신 CloudWatch 데이터를 사용하여 6시간마다 업데이트됩니다. 새로운 데이터가 들어오면 예측 스케일링을 통해 미래 예측의 정확도를 지속적으로 개선할 수 있습니다.

예측 스케일링을 처음 활성화하면 예측 전용 모드로 실행됩니다. 이 모드에서는 용량 예측을 생성하지만 이러한 예측을 기반으로 Auto Scaling 그룹을 실제로 확장하지는 않습니다. 이를 통해 예측의 정확성과 적합성을 평가할 수 있습니다. `GetPredictiveScalingForecast`API작업 또는 를 사용하여 예측 데이터를 볼 수 있습니다. AWS Management Console

예측 데이터를 검토하고 해당 데이터를 기반으로 조정을 시작하기로 결정한 후에는 조정 정책을 예측 및 규모 조정 모드로 전환하십시오. 이 모드에서는:

- 예측에 따르면 부하가 증가할 것으로 예상되면 Amazon EC2 Auto Scaling은 규모를 축소하여 용량을 늘립니다.
- 예측에서 부하 감소가 예상되더라도 용량을 줄일 수 있을 정도로 규모를 축소하지는 않을 것입니다. 더 이상 필요하지 않은 용량을 제거하려면 동적 조정 정책을 생성해야 합니다.

기본적으로 Amazon EC2 Auto Scaling은 각 시간이 시작될 때 해당 시간의 예측을 기반으로 Auto Scaling 그룹을 조정합니다. 선택적으로 `PutScalingPolicy` API 작업의 `SchedulingBufferTime` 속성을 사용하거나 의 사전 시작 인스턴스 설정을 사용하여 더 빠른 시작 시간을 지정할 수 있습니다. AWS Management Console이로 인해 Amazon EC2 Auto Scaling은 예측된 수요보다 먼저 새 인스턴스를 시작하여 부팅할 시간을 주고 트래픽을 처리할 준비를 갖추게 됩니다.

예상 수요보다 먼저 새 인스턴스를 시작하도록 지원하려면 Auto Scaling 그룹의 기본 인스턴스 워밍업을 활성화하는 것이 좋습니다. 이는 동적 규모 조정 정책에 따라 용량을 줄여야 한다고 명시되어 있더라도 확장 활동 이후 Amazon EC2 Auto Scaling에서 규모를 축소하지 않는 기간을 지정합니다. 이렇게

하면 새로 시작된 인스턴스가 확장 작업을 고려하기 전에 증가된 트래픽을 처리하기 시작할 충분한 시간을 확보할 수 있습니다. 자세한 내용은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 단원을 참조하십시오.

최대 용량 제한

Auto Scaling 그룹에는 그룹에 대해 시작할 수 있는 최대 EC2 인스턴스 수를 제한하는 최대 용량 설정이 있습니다. 기본적으로 조정 정책이 설정된 경우 최대 용량보다 용량을 늘릴 수 없습니다.

또는 예측 용량이 Auto Scaling 그룹의 최대 용량에 근접하거나 초과할 경우 그룹의 최대 용량이 자동으로 증가하도록 허용할 수 있습니다. 이 동작을 활성화하려면 작업의 `MaxCapacityBreachBehavior` 및 `MaxCapacityBuffer` 속성을 사용하거나 의 최대 용량 동작 설정을 사용하십시오 AWS Management Console. `PutScalingPolicy` API

> **Warning**
>
> 최대 용량을 자동으로 늘리도록 허용할 때는 주의해야 합니다. 증가된 최대 용량을 모니터링 및 관리하지 않으면 의도한 것보다 많은 인스턴스가 시작될 수 있습니다. 증가된 최대 용량은 수동으로 업데이트하기 전까지 Auto Scaling 그룹의 새로운 정상 최대 용량이 됩니다. 최대 용량은 원래 최대 용량으로 자동으로 다시 감소하지 않습니다.

고려 사항

- 예측 스케일 아웃이 워크로드에 적합한지 확인합니다. 워크로드가 특정 요일이나 시간에 반복되는 로드 패턴을 나타내는 경우, 예측 조정에 적합합니다. 이를 확인하려면 예상 전용 모드에서 예측적 조정 정책을 구성한 다음 콘솔의 권장 사항을 참조하세요. Amazon EC2 Auto Scaling은 잠재적 정책 성능에 대한 관찰을 기반으로 권장 사항을 제공합니다. 예측적 조정이 애플리케이션 크기를 능동적으로 조정하도록 하기 전에 예측과 권장 사항을 평가합니다.
- 예측 조정이 예상을 시작하려면 최소 24시간 동안의 기록 데이터가 필요합니다. 그러나 기록 데이터가 2주 전체에 걸쳐 있는 경우, 예측이 더 정확합니다. 새 Auto Scaling 그룹을 생성하고 이전 그룹을 삭제하여 애플리케이션을 업데이트하는 경우, 예측 조정에서 예상 생성을 다시 시작하기 전에 새 Auto Scaling 그룹에는 24시간 동안의 기록 로드 데이터가 필요합니다. 사용자 지정 지표를 사용하여 이전 Auto Scaling 그룹과 새 Auto Scaling 그룹의 지표를 집계할 수 있습니다. 그렇지 않으면 더욱 정확한 예측을 위해 며칠을 기다려야 할 수 있습니다.
- 애플리케이션의 전체 부하를 정확히 나타내며 확장해야 하는 애플리케이션 측면에서 가장 중요한 측면을 나타내는 부하 지표를 선택하십시오.

- 동적 스케일링과 예측 스케일링을 사용하면 트래픽이 적은 기간에는 규모를 조정하고 트래픽이 예상보다 많을 때는 규모를 축소하여 애플리케이션의 수요 곡선을 면밀히 추적할 수 있습니다. 여러 조정 정책이 활성화된 경우, 각 정책에서는 원하는 용량을 독립적으로 결정하며 원하는 용량이 최대값으로 설정됩니다. 예컨대, 대상 추적 조정 정책에서 대상 사용률을 유지하기 위해 인스턴스 10개가 필요하고, 예측 조정 정책에서 대상 사용률을 유지하기 위해 인스턴스 8개가 필요한 경우, 그룹의 원하는 용량은 10으로 설정됩니다. 동적 규모 조정을 처음 사용하는 경우 대상 추적 조정 정책을 사용하는 것이 좋습니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 동적 스케일링 단원을 참조하십시오.
- 예측 조정의 핵심적인 가정은 Auto Scaling 그룹이 동질적이며 모든 인스턴스의 용량이 동일하다는 것입니다. 그룹에 속하지 않는 경우, 예측 용량이 정확하지 않을 수 있습니다. 따라서 용량이 동일하지 않은 다양한 유형의 인스턴스가 프로비저닝될 수 있으므로 혼합 인스턴스 그룹에 대한 예측 조정 정책을 만들 때는 주의해야 합니다. 다음은 예측 용량이 부정확할 수 있는 몇 가지 예입니다.
  - 예측 조정 정책은 CPU 사용률을 기반으로 하지만, 각 Auto Scaling 인스턴스의 vCPUs 수는 인스턴스 유형에 따라 다릅니다.
  - 예측 조정 정책은 네트워크 인 또는 네트워크 아웃에 근거하여 하지만, 각 Auto Scaling 인스턴스의 네트워크 대역폭 처리량은 인스턴스 유형에 따라 다릅니다. 예컨대, M5 및 M5n 인스턴스 유형은 비슷하지만 M5n 인스턴스 유형은 네트워크 처리량이 상당히 높습니다.

지원되는 지역

- 미국 동부(버지니아 북부)
- 미국 동부(오하이오)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오레곤)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(자카르타)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)

- 캐나다(중부)
- 중국(베이징)
- 중국(닝샤)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스톡홀름)
- 중동(바레인)
- 중동 () UAE
- 남아메리카(상파울루)
- AWS GovCloud (미국 동부)
- AWS GovCloud (미국 서부)

Auto Scaling 그룹에 대한 예측 조정 정책 생성

다음 절차는 OR를 사용하여 예측 조정 정책을 생성하는 데 도움이 됩니다 AWS Management Console . AWS CLI

Auto Scaling 그룹이 신규 그룹인 경우 Amazon EC2 Auto Scaling에서 해당 그룹에 대한 예측을 생성하기 전에 최소 24시간 분량의 데이터를 제공해야 합니다.

내용

- 예측 조정 정책 생성(콘솔)
- 예측 조정 정책 생성(AWS CLI)

예측 조정 정책 생성(콘솔)

예측 조정 정책을 처음 생성하는 경우, 콘솔을 사용하여 예측 전용 모드에서 여러 예측 조정 정책을 생성하는 것이 좋습니다. 이를 통해 다양한 지표와 목표값의 잠재적 영향을 테스트할 수 있습니다. 각 Auto Scaling 그룹에 대해 여러 예측 조정 정책을 생성할 수 있지만 활성 조정에는 하나의 정책만 사용할 수 있습니다.

콘솔에서 예측 조정 정책 생성(사전 정의된 지표)

다음 절차를 사용하여 사전 정의된 지표 (CPU, 네트워크 I/O 또는 대상당 Application Load Balancer 요청 수) 를 사용하여 예측 조정 정책을 생성할 수 있습니다. 예측 조정 정책을 만드는 가장 쉬운 방법은 미리 정의된 지표를 사용하는 것입니다. 사용자 지정 지표를 사용하려는 경우, 콘솔에서 예측 조정 정책 생성(사용자 지정 지표)을 참조하세요.

예측 조정 정책 생성

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. Automatic scaling(자동 조정) 탭의 Scaling policies(조정 정책)에서 Create predictive scaling policy(예측 조정 정책 생성)를 선택합니다.
4. 정책의 이름을 입력합니다.
5. 예측 기반 확장을 켜면 Amazon EC2 Auto Scaling에서 바로 조정을 시작할 수 있는 권한을 부여할 수 있습니다.

   정책을 예상 전용 모드에서 유지하려면 Scale based on forecast(예상 기반 조정)를 끕니다.
6. Metrics(지표)의 경우, 옵션 목록에서 지표를 선택합니다. 옵션에는 네트워크 CPU입력, 네트워크 출력, Application Load Balancer 요청 수 및 사용자 지정 지표 쌍이 포함됩니다.

   Application Load Balancer request count per target(대상당 Application Load Balancer 요청 수)을 선택한 경우, Target group(대상 그룹)에서 대상 그룹을 선택합니다. Application Load Balancer request count per target(대상별 Application Load Balancer 요청 수)는 Auto Scaling 그룹에 Application Load Balancer 대상 그룹을 연결한 경우에만 지원됩니다.

   Custom metric pair(사용자 지정 지표 쌍)를 선택하면 Load metric(로드 지표) 및 Scaling metric(조정 지표)의 드롭다운 목록에서 개별 지표를 선택합니다.
7. 목표 사용률에는 Amazon EC2 Auto Scaling에서 유지해야 하는 목표 값을 입력합니다. Amazon EC2 Auto Scaling은 평균 사용률이 목표 사용률에 도달하거나 지정된 최대 인스턴스 수에 도달할 때까지 용량을 확장합니다.

조정 지표	목표 사용률의 의미
CPU	각 인스턴스가 이상적으로 사용해야 CPU 하는 비율입니다.
네트워크 입력	각 인스턴스가 이상적으로 수신해야 하는 분당 평균 바이트 수
네트워크 출력	각 인스턴스가 이상적으로 전송해야 하는 분당 평균 바이트 수
대상당 Application Load Balancer 요청 수	각 인스턴스가 이상적으로 수신해야 하는 분당 평균 요청 수

8. (옵션) Pre-launch instances(사전 인스턴스 시작)에서는 로드 증가에 대한 예상을 호출하기 전 인스턴스를 미리 시작하려는 시간을 선택합니다.
9. (선택 사항) 최대 용량 동작의 경우, 예측 용량이 정의된 최대값을 초과할 때 Amazon EC2 Auto Scaling을 그룹의 최대 용량보다 크게 확장할지 여부를 선택합니다. 이 설정을 켜면 트래픽이 가장 높을 것으로 예측되는 기간에 스케일 아웃이 발생할 수 있습니다.
10. (옵션) Buffer maximum capacity above the forecasted capacity(예상 용량을 초과하는 버퍼 최대 용량)에서 예상 용량이 최대 용량에 근접하거나 최대 용량을 초과하는 경우, 사용할 추가 용량을 선택합니다. 값은 예측 용량에 상대적인 백분율로 지정됩니다. 예컨대, 버퍼가 10이면 이는 10% 버퍼를 의미합니다. 따라서 예상 용량이 50이고 최대 용량이 40이라면 최대 유효 용량은 55입니다.

    0으로 설정된 경우 Amazon EC2 Auto Scaling은 최대 용량보다 큰 용량까지 용량을 예상 용량과 같지만 초과하지는 않도록 확장할 수 있습니다.
11. Create predictive scaling policy(예측 조정 정책 생성)를 선택합니다.

콘솔에서 예측 조정 정책 생성(사용자 지정 지표)

사용자 지정 지표를 사용하여 예측 조정 정책을 생성하려면 다음 절차를 따르세요. 사용자 지정 지표에는 에서 제공하는 다른 CloudWatch 지표나 게시하는 대상 지표가 포함될 수 CloudWatch 있습니다. 대상당 네트워크 I/O 또는 Application Load Balancer 요청 수를 사용하려면 CPU 을 참조하십시오. 콘솔에서 예측 조정 정책 생성(사전 정의된 지표)

사용자 지정 지표를 사용하여 예측 조정 정책을 생성하려면 다음을 수행해야 합니다.

- Amazon EC2 Auto Scaling이 내 지표와 상호 작용할 수 있도록 원시 쿼리를 제공해야 CloudWatch 합니다. 자세한 내용은 사용자 지정 지표를 사용하는 고급 예측 규모 조정 정책 단원을 참조하

십시오. Amazon EC2 Auto Scaling에서 CloudWatch 메트릭 데이터를 추출할 수 있는지 확인하려면 각 쿼리가 데이터 포인트를 반환하는지 확인하십시오. CloudWatch 콘솔이나 CloudWatch GetMetricDataAPI작업을 사용하여 이를 확인하십시오.

> **Note**
>
> Amazon EC2 Auto Scaling 콘솔의 JSON 에디터에서 샘플 JSON 페이로드를 제공합니다. 이 예제는 에서 제공한 다른 CloudWatch 지표 AWS 또는 이전에 게시한 지표를 추가하는 데 필요한 키-값 쌍에 대한 참조를 제공합니다. CloudWatch 이를 시작점으로 사용한 다음 필요에 맞게 사용자 지정할 수 있습니다.

- 메트릭 수학을 사용하는 경우 고유한 시나리오에 JSON 맞게 수동으로 구성해야 합니다. 자세한 내용은 지표 수학 표현식 사용 단원을 참조하십시오. 정책에서 지표 수학을 사용하기 전에 지표 수학 표현식에 근거하여 하는 지표 쿼리가 유효한지 확인하고 단일 시계열을 반환해야 합니다. CloudWatch 콘솔이나 CloudWatch GetMetricDataAPI작업을 사용하여 이를 확인하십시오.

잘못된 Auto Scaling 그룹 명칭과 같이 잘못된 데이터를 제공하여 쿼리에 오류가 발생하면 예측에 데이터가 없습니다. 사용자 지정 지표 문제 해결 방법은 예측 규모 조정 정책의 사용자 지정 지표에 대한 고려 사항을 참조하세요.

예측 조정 정책 생성

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. Automatic scaling(자동 조정) 탭의 Scaling policies(조정 정책)에서 Create predictive scaling policy(예측 조정 정책 생성)를 선택합니다.
4. 정책의 이름을 입력합니다.
5. 예측 기반 확장을 켜면 Amazon EC2 Auto Scaling에서 바로 조정을 시작할 수 있는 권한을 부여할 수 있습니다.

   정책을 예상 전용 모드에서 유지하려면 Scale based on forecast(예상 기반 조정)를 끕니다.
6. Metrics(지표)에서 Custom metric pair(사용자 지정 지표 쌍)을 선택합니다.

a. 로드 지표의 경우 사용자 지정 CloudWatch 지표를 선택하여 사용자 지정 지표를 사용하십시오. 정책에 대한 JSON 부하 지표 정의가 포함된 페이로드를 구성하여 JSON 편집기 상자에 붙여넣고 상자에 이미 있는 내용을 대체합니다.

b. 스케일링 메트릭의 경우 커스텀 CloudWatch 메트릭을 선택하여 커스텀 메트릭을 사용하세요. 정책에 대한 규모 조정 지표 정의가 포함된 JSON 페이로드를 구성하여 JSON 편집기 상자에 붙여넣고 상자에 이미 있는 내용을 대체합니다.

c. (옵션) 사용자 지정 용량 지표를 추가하려면 Add custom capacity metric(사용자 지정 용량 지표 추가) 확인란을 선택합니다. 정책에 대한 용량 메트릭 정의가 포함된 JSON 페이로드를 구성하여 JSON 편집기 상자에 붙여넣고 상자에 이미 있는 내용을 대체합니다.

   용량 지표 데이터가 여러 Auto Scaling 그룹에 걸쳐 있는 경우, 용량에 대한 새 시계열을 생성하려면 이 옵션을 활성화하기만 하면 됩니다. 이 경우, 지표 수학을 사용하여 데이터를 단일 시계열로 집계해야 합니다.

7. 목표 사용률에는 Amazon EC2 Auto Scaling에서 유지해야 하는 목표 값을 입력합니다. Amazon EC2 Auto Scaling은 평균 사용률이 목표 사용률에 도달하거나 지정된 최대 인스턴스 수에 도달할 때까지 용량을 확장합니다.

8. (옵션) Pre-launch instances(사전 인스턴스 시작)에서는 로드 증가에 대한 예상을 호출하기 전 인스턴스를 미리 시작하려는 시간을 선택합니다.

9. (선택 사항) 최대 용량 동작의 경우, 예측 용량이 정의된 최대값을 초과할 때 Amazon EC2 Auto Scaling을 그룹의 최대 용량보다 크게 확장할지 여부를 선택합니다. 이 설정을 켜면 트래픽이 가장 높을 것으로 예측되는 기간에 스케일 아웃이 발생할 수 있습니다.

10. (옵션) Buffer maximum capacity above the forecasted capacity(예상 용량을 초과하는 버퍼 최대 용량)에서 예상 용량이 최대 용량에 근접하거나 최대 용량을 초과하는 경우, 사용할 추가 용량을 선택합니다. 값은 예측 용량에 상대적인 백분율로 지정됩니다. 예컨대, 버퍼가 10이면 이는 10% 버퍼를 의미합니다. 따라서 예상 용량이 50이고 최대 용량이 40이라면 최대 유효 용량은 55입니다.

    0으로 설정된 경우 Amazon EC2 Auto Scaling은 최대 용량보다 큰 용량까지 용량을 예상 용량과 같지만 초과하지는 않도록 확장할 수 있습니다.

11. Create predictive scaling policy(예측 조정 정책 생성)를 선택합니다.

예측 조정 정책 생성(AWS CLI)

다음과 AWS CLI 같이 Auto Scaling 그룹에 대한 예측 조정 정책을 구성하십시오. 각각 교체하십시오.*user input placeholder* 자신의 정보로

지정할 수 있는 CloudWatch 지표에 대한 자세한 내용은 Amazon EC2 Auto Scaling API 참조서를 참조하십시오 PredictiveScalingMetricSpecification.

예 1: 예측을 생성하지만 조정하지 않는 예측 조정 정책

다음 예제 정책은 목표 사용률이 1인 예측 규모 조정을 위해 CPU 사용률 지표를 사용하는 전체 정책 구성을 보여줍니다. 40 `ForecastOnly`사용할 모드를 명시적으로 지정하지 않는 한 모드가 기본적으로 사용됩니다. 이 구성을 `config.json` 파일에 저장합니다.

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 40,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ASGCPUUtilization"
            }
        }
    ]
}
```

명령줄에서 정책을 생성하려면 다음 예와 같이 지정된 구성 파일을 put-scaling-policy사용하여 명령을 실행합니다.

```
aws autoscaling put-scaling-policy --policy-name cpu40-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

성공하면 이 명령은 정책의 Amazon 리소스 이름 (ARN) 을 반환합니다.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/cpu40-predictive-scaling-
policy",
  "Alarms": []
}
```

예 2: 예측하여 조정하는 예측 조정 정책

Amazon EC2 Auto Scaling이 예측하고 확장할 수 있도록 허용하는 정책의 경우 값이 인 속성을 `Mode` 추가하십시오`ForecastAndScale`. 다음 예에서는 Application Load Balancer 요청 수 지표를 사용하

는 정책 구성을 보여 줍니다. 목표 사용률은 1000이고 예측 조정은 ForecastAndScale 모드로 설정되어 있습니다.

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 1000,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ALBRequestCount",
                "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-target-group/943f017f100becff"
            }
        }
    ],
    "Mode": "ForecastAndScale"
}
```

이 정책을 생성하려면 다음 예제와 같이 지정된 구성 파일을 put-scaling-policy사용하여 명령을 실행하십시오.

```
aws autoscaling put-scaling-policy --policy-name alb1000-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

성공하면 이 명령은 정책의 Amazon 리소스 이름 (ARN) 을 반환합니다.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:19556d63-7914-4997-8c81-d27ca5241386:autoScalingGroupName/my-asg:policyName/alb1000-predictive-scaling-policy",
  "Alarms": []
}
```

예 3: 최대 용량보다 높게 조정할 수 있는 예측 조정 정책

다음 예에서는 일반보다 높은 부하를 처리해야 할 때 그룹의 최대 크기 제한보다 높게 조정할 수 있는 정책을 생성하는 방법을 보여 줍니다. 기본적으로 Amazon EC2 Auto Scaling은 정의된 최대 EC2 용량보다 더 많이 용량을 확장하지 않습니다. 그러나 성능 또는 가용성 문제를 피하기 위해 용량을 약간 늘려서 스케일 아웃하는 것이 도움이 될 수 있습니다.

용량이 그룹의 최대 규모에 가까울 것으로 예측될 때 Amazon EC2 Auto Scaling에서 추가 용량을 프로비저닝할 수 있는 공간을 제공하려면 다음 예와 같이 `MaxCapacityBreachBehavior` 및 `MaxCapacityBuffer` 속성을 지정하십시오. `MaxCapacityBreachBehavior`를 값 `IncreaseMaxCapacity`로 지정해야 합니다. 그룹에서 보유할 수 있는 최대 인스턴스 수는 `MaxCapacityBuffer`의 값에 따라 달라집니다.

```
{
    "MetricSpecifications": [
        {
            "TargetValue": 70,
            "PredefinedMetricPairSpecification": {
                "PredefinedMetricType": "ASGCPUUtilization"
            }
        }
    ],
    "MaxCapacityBreachBehavior": "IncreaseMaxCapacity",
    "MaxCapacityBuffer": 10
}
```

이 예에서는 10% 버퍼(`"MaxCapacityBuffer": 10`)를 사용하도록 구성되어 있습니다. 따라서 예상 용량이 50이고 최대 용량이 40이라면 최대 유효 용량은 55입니다. 최대 용량보다 높은 용량을 예측 용량을 초과하지 않고 예측 용량과 동일하게 조정할 수 있는 정책의 버퍼는 0입니다(`"MaxCapacityBuffer": 0`).

이 정책을 생성하려면 다음 예제와 같이 지정된 구성 파일을 put-scaling-policy사용하여 명령을 실행하십시오.

```
aws autoscaling put-scaling-policy --policy-name cpu70-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

성공하면 이 명령은 정책의 Amazon 리소스 이름 (ARN) 을 반환합니다.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:d02ef525-8651-4314-
bf14-888331ebd04f:autoScalingGroupName/my-asg:policyName/cpu70-predictive-scaling-
policy",
  "Alarms": []
}
```

예측적 조정 정책 평가

예측 조정 정책을 사용하여 Auto Scaling 그룹을 확장하기 전에 Amazon EC2 Auto Scaling 콘솔에서 정책에 대한 권장 사항 및 기타 데이터를 검토하십시오. 이는 예측의 정확성이 확인될 때까지 예측적 조정 정책이 실제 용량을 조정하는 것을 원하지 않기 때문에 중요합니다.

Auto Scaling 그룹이 새로 가입한 경우, Amazon EC2 Auto Scaling에서 24시간 동안 첫 번째 예측을 생성하도록 하십시오.

Amazon EC2 Auto Scaling은 예측을 생성할 때 과거 데이터를 사용합니다. Auto Scaling 그룹에 최근 과거 데이터가 아직 많지 않은 경우, Amazon EC2 Auto Scaling은 현재 사용 가능한 과거 집계에서 생성된 집계로 예측을 일시적으로 채울 수 있습니다. 정책 생성 날짜 전 최대 2주 동안 예측이 채워집니다.

내용

- 예측적 조정 권장 사항 보기
- 예측적 조정 모니터링 그래프 검토
- 다음을 사용하여 예측 규모 조정 지표를 모니터링할 수 있습니다. CloudWatch

예측적 조정 권장 사항 보기

효과적인 분석을 위해서는 Amazon EC2 Auto Scaling에서 비교할 예측 규모 조정 정책이 두 개 이상 있어야 합니다. (그러나 여전히 단일 정책에 대한 결과를 검토할 수 있습니다.) 여러 정책을 생성할 때 한 지표를 사용하는 정책을 다른 지표를 사용하는 정책과 비교하여 평가할 수 있습니다. 또한 다양한 대상 값과 지표 조합의 영향을 평가할 수 있습니다. 예측 조정 정책이 생성되면 Amazon EC2 Auto Scaling은 즉시 어떤 정책이 그룹 규모 조정에 더 효과적인지 평가하기 시작합니다.

Amazon EC2 Auto Scaling 콘솔에서 권장 사항을 보려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. Auto scaling 탭의 예측적 조정 정책에서 권장 사항과 함께 정책에 대한 세부 정보를 볼 수 있습니다. 권장 사항은 예측적 조정 정책이 사용하지 않는 것보다 더 나은 작업을 수행하는지 여부를 알려줍니다.

예측적 조정 정책이 그룹에 적합한지 확실하지 않은 경우, 가용성에 미치는 영향 열과 비용에 미치는 영향 열을 검토하여 적합한 정책을 선택합니다. 각 열에 대한 정보는 정책의 영향을 알려줍니다.

- 가용성에 미치는 영향: 정책을 사용하지 않을 때와 비교하여 정책을 통해 워크로드를 처리하기에 충분한 인스턴스를 프로비저닝함으로써 가용성에 미치는 부정적인 영향을 방지할 수 있는지 여부를 설명합니다.
- 비용에 미치는 영향: 정책을 사용하지 않을 때와 비교하여 정책을 통해 인스턴스를 과다 프로비저닝하지 않음으로써 비용에 미치는 부정적인 영향을 방지할 수 있는지 여부를 설명합니다. 과다 프로비저닝하면 인스턴스가 충분히 사용되지 않거나 유휴 상태가 되어 비용에 미치는 영향만 가중됩니다.

정책이 여러 개인 경우, 더 낮은 비용으로 가장 많은 가용성 이점을 제공하는 정책 이름 옆에 최상의 예측 태그가 표시됩니다. 가용성 영향에 더 많은 가중치가 부여됩니다.

4. (옵션) 권장 사항 결과에 대해 원하는 기간을 선택하려면 평가 기간 드롭다운에서 원하는 값(2일, 1주, 2주, 4주, 6주 또는 8주)을 선택합니다. 기본적으로 평가 기간은 지난 2주입니다. 평가 기간이 길수록 권장 사항 결과에 더 많은 데이터 포인트가 제공됩니다. 그러나 로드 패턴이 변경된 경우(예: 예외적인 수요 기간 이후) 데이터 포인트를 더 추가해도 결과가 개선되지 않을 수 있습니다. 이 경우, 보다 최근 데이터를 살펴봄으로써 보다 집중적인 권장 사항을 얻을 수 있습니다.

> **Note**
>
> 예상 전용 모드에 있는 정책에 대한 권장 사항만 생성됩니다. 권장 사항 기능은 평가 기간 내내 정책이 예상 전용 모드일 때 더 잘 작동합니다. 예상 및 조정 모드에서 정책을 시작하고 나중에 예상 전용 모드로 전환하면 해당 정책에 대한 결과가 편향될 가능성이 높습니다. 이는 정책이 이미 실제 용량에 기여했기 때문입니다.

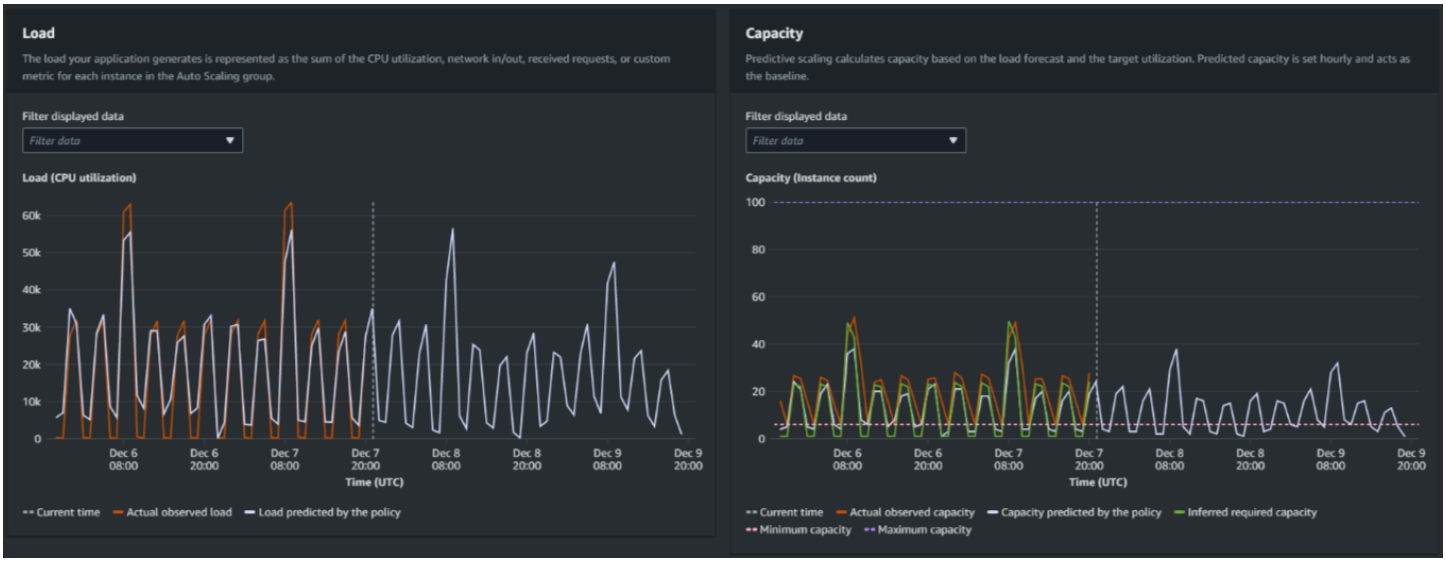
예측적 조정 모니터링 그래프 검토

Amazon EC2 Auto Scaling 콘솔에서 이전 일, 주 또는 월의 예측을 검토하여 시간 경과에 따른 정책 성과를 시각화할 수 있습니다. 정책을 통해 실제 용량을 조정할지 여부를 결정할 때 이 정보를 사용하여 예측의 정확성을 평가할 수도 있습니다.

Amazon EC2 Auto Scaling 콘솔에서 예측 규모 조정 모니터링 그래프를 검토하려면

1. 예측 크기 조정 정책 목록에서 정책을 선택합니다.
2. 모니터링 섹션에서 로드 및 용량 정책에 대한 과거 및 미래 예측을 실제 값과 비교하여 볼 수 있습니다. 로드 그래프는 선택한 로드 지표에 대한 로드 예측 및 실제 값을 보여줍니다. 용량 그래프에는 정책에서 예측한 인스턴스 수가 표시됩니다. 또한 시작된 실제 인스턴스 수도 포함됩니다. 수직선은 과거 값과 미래 예측을 구분합니다. 정책이 생성되면 바로 이 그래프를 사용할 수 있습니다.
3. (옵션) 차트에 표시된 기록 데이터 양을 변경하려면 페이지 상단의 평가 기간 드롭다운에서 원하는 값을 선택합니다. 평가 기간은 어떤 식으로든 이 페이지의 데이터를 변환하지 않으며, 표시되는 기록 데이터의 양만 변경합니다.

다음 이미지는 예측이 여러 번 적용된 경우의 로드 및 용량 그래프를 보여줍니다. 예측적 조정은 기록 로드 데이터에 근거하여 로드를 예측합니다. 애플리케이션이 생성하는 로드는 Auto Scaling 그룹의 각 인스턴스에 대한 CPU 사용률, 네트워크 입/출력, 수신 요청 또는 사용자 지정 지표의 합계로 표시됩니다. 예측적 조정은 로드 예측과 조정 지표에 대해 달성하려는 목표 사용률에 근거하여 향후 용량 요건을 계산합니다.



로드 그래프의 데이터 비교

각 수평선은 1시간 간격으로 보고되는 서로 다른 데이터 포인트 세트를 나타냅니다.

1. 실제 관찰된 부하는 선택한 부하 지표의 SUM 통계를 사용하여 과거의 시간당 총 부하를 보여줍니다.
2. 정책에 의해 예측되는 로드는 시간별 로드 예측을 보여줍니다. 이 예측은 지난 2주간의 실제 로드 관찰에 근거하여 합니다.

용량 그래프의 데이터 비교

각 수평선은 1시간 간격으로 보고되는 서로 다른 데이터 포인트 세트를 나타냅니다.

1. 실제 관찰된 용량은 선택한 기간 동안 적용된 다른 조정 정책과 최소 그룹 크기에 따라 달라지는 과거 Auto Scaling 그룹의 실제 용량을 보여줍니다.
2. 정책에서 예측한 용량은 정책이 예상 및 조정 모드일 때 매 시간 시작 시 예상할 수 있는 기본 용량을 보여줍니다.
3. 추론된 필수 용량은 선택한 대상 값에서 조정 지표를 유지하는 데 이상적인 용량을 보여줍니다.
4. 최소 용량은 Auto Scaling 그룹의 최소 용량을 보여줍니다.
5. 최대 용량은 Auto Scaling 그룹의 최대 용량을 보여줍니다.

추론된 필수 용량을 계산하기 위해 먼저 각 인스턴스가 지정된 대상 값에서 균등하게 사용된다고 가정합니다. 실제로 인스턴스는 균등하게 사용되지 않습니다. 그러나 사용률이 인스턴스 간에 균일하게 분산되어 있다고 가정하면 필요한 용량의 양을 예측할 수 있습니다. 그런 다음 용량 요건은 예측적 조정 정책에 사용한 조정 지표에 반비례하도록 계산됩니다. 즉, 용량이 증가함에 따라 동일한 백분율로 조정 지표가 감소합니다. 예컨대, 용량이 두 배로 증가하면 조정 지표는 절반으로 감소해야 합니다.

추론된 필수 용량에 대한 공식:

```
sum of (actualCapacityUnits*scalingMetricValue)/(targetUtilization)
```

예컨대, 지정된 시간에 대한 `actualCapacityUnits`(`10`) 및 `scalingMetricValue`(`30`)를 가져옵니다. 그런 다음 예측적 조정 정책(`60`)에 지정한 `targetUtilization`을 가져오고 같은 시간에 대해 추론된 필수 용량을 계산합니다. 이 예는 값 5를 반환합니다. 즉, 조정 지표의 대상 값에 반비례하여 용량을 유지하는 데 필요한 추론 용량은 5입니다.

> **Note**
>
> 애플리케이션의 비용 절감 및 가용성을 조정하고 개선하기 위해 다양한 레버를 사용할 수 있습니다.
>
> - 기준 용량과 동적 조정에 예측적 조정을 사용하여 추가 용량을 처리합니다. 동적 조정은 예측적 조정과 독립적으로 작동하며 현재 사용률에 따라 스케일 아웃되고 축소됩니다. 먼저 Amazon EC2 Auto Scaling은 각 동적 조정 정책에 대한 권장 인스턴스 수를 계산합니다. 그런 다음 가장 많은 수의 인스턴스를 제공하는 정책에 따라 조정됩니다.
> - 로드가 감소할 때 스케일인이 수행되도록 하려면 Auto Scaling 그룹에 스케일 인 부분이 활성화된 동적 조정 정책이 항상 하나 이상 있어야 합니다.

- 최소 및 최대 용량이 너무 제한적이지 않은지 확인하여 조정 성능을 개선할 수 있습니다. 권장 인스턴스 수가 최소 및 최대 용량 범위에 속하지 않는 정책은 스케일 인 및 스케일 아웃되지 않습니다.

다음을 사용하여 예측 규모 조정 지표를 모니터링할 수 있습니다. CloudWatch

필요에 따라 Amazon EC2 Auto Scaling 콘솔 CloudWatch 대신 Amazon에서 예측 조정을 위한 모니터링 데이터에 액세스하는 것을 선호할 수 있습니다. 예측 조정 정책을 생성하고 나면, 해당 정책이 미래의 로드와 용량을 예측하는 데 사용되는 데이터를 수집합니다. 이 데이터가 수집되고 나면 정기적으로 데이터가 자동으로 저장됩니다 CloudWatch . 그런 다음 시간 경과에 따른 정책 성과를 시각화하는 CloudWatch 데 사용할 수 있습니다. 성과 지표가 에서 정의한 한도를 초과하여 변경될 경우 알려주는 CloudWatch 경보를 생성할 수도 있습니다. CloudWatch

주제

- 기록 예측 데이터 시각화
- 지표 수식을 사용하여 정확도 지표 생성

기록 예측 데이터 시각화

에서 예측 규모 조정 정책에 대한 부하 및 용량 예측 데이터를 볼 수 있습니다. CloudWatch 이는 다른 CloudWatch 지표에 대한 예측을 단일 그래프로 시각화할 때 유용할 수 있습니다. 또한 시간 경과에 따른 추세를 볼 수 있도록 넓은 시간 범위를 표시할 때 도움이 될 수 있습니다. 최대 15개월의 기록 지표에 액세스하여, 정책의 성능을 전반적으로 더 잘 파악할 수 있습니다.

자세한 내용은 예측 조정 지표 및 차원 단원을 참조하십시오.

콘솔을 사용하여 과거 예측 데이터를 보려면 CloudWatch

1. 에서 CloudWatch 콘솔을 https://console.aws.amazon.com/cloudwatch/여십시오.
2. 탐색 창에서 지표(Metrics)를 선택한 다음 모든 지표(All metrics)를 선택합니다.
3. Auto Scaling 그룹(Auto Scaling) 지표 네임스페이스를 선택합니다.
4. 다음 옵션 중 하나를 선택하여 로드 예측 또는 용량 예측 지표를 봅니다.
   - 예측 조정 로드 예측
   - 예측 조정 용량 예측

5. 검색 필드에 예측 스케일링 정책의 이름 또는 Auto Scaling 그룹의 이름을 입력한 다음 Enter 키를 눌러 결과를 필터링합니다.
6. 측정치를 그래프로 표시하려면 측정치 옆에 있는 확인란을 선택합니다. 그래프 이름을 변경하려면 연필 아이콘을 선택합니다. 시간 범위를 변경하려면 제공되는 값 중 하나를 선택하거나 사용자 지정을 선택합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 지표 그래프 작성을 참조하십시오.
7. 기간을 변경하려면 그래프로 표시된 지표 탭을 선택합니다. 열 머리글이나 개별 값을 선택한 후 다른 통계를 선택합니다. 각 지표에 대해 원하는 통계를 선택할 수 있지만 모든 통계가 지표에 PredictiveScalingLoadForecast유용하지는 않습니다. PredictiveScalingCapacityForecast 예컨대, Average, Minimum 및 Maximum 통계는 유용하지만 Sum 통계는 유용하지 않습니다.
8. 그래프에 다른 지표를 추가하려면 찾아보기(Browse)에서 모두(All)를 선택하고 특정 지표를 찾은 다음 해당 지표 옆에 있는 확인란을 선택합니다. 최대 10개의 지표를 추가할 수 있습니다.

   예를 들어 그래프에 실제 CPU 사용률 값을 추가하려면 EC2네임스페이스를 선택한 다음 By Auto Scaling Group을 선택합니다. 그런 다음 CPUUtilization메트릭과 특정 Auto Scaling 그룹의 확인란을 선택합니다.
9. (선택 사항) 그래프를 CloudWatch 대시보드에 추가하려면 작업, 대시보드에 추가를 선택합니다.

지표 수식을 사용하여 정확도 지표 생성

지표 수학을 사용하면 여러 CloudWatch 지표를 쿼리하고 수학 식을 사용하여 이러한 지표를 기반으로 새 시계열을 만들 수 있습니다. 결과 시계열을 CloudWatch 콘솔에서 시각화하고 대시보드에 추가할 수 있습니다. 지표 수학에 대한 자세한 내용은 Amazon 사용 CloudWatch 설명서의 지표 수학 사용을 참조하십시오.

지표 수학을 사용하면 Amazon EC2 Auto Scaling에서 예측 조정을 위해 생성하는 데이터를 다양한 방식으로 그래프로 표시할 수 있습니다. 이는 시간 경과에 따른 정책 성능을 모니터링하고 지표 조합을 개선할 수 있는지 여부를 파악하는 데 도움이 됩니다.

예를 들어, 메트릭 수학 식을 사용하여 평균 절대 백분율 오차 (MAPE) 를 모니터링할 수 있습니다. 이 MAPE 지표는 예측 기간동안 예측된 값과 관측된 실제 값 간의 차이를 모니터링하는 데 도움이 됩니다. 값의 변화는 응용 프로그램의 특성이 변화함에 따라 시간이 지남에 따라 정책 성능이 저하되고 있는지 여부를 나타낼 MAPE 수 있습니다. 값이 MAPE 증가하면 예측값과 실제 값 사이의 격차가 더 커진다는 의미입니다.

예: 지표 수학 표현식

이 유형의 그래프로 시작하려면, 다음 예에 표시된 것과 같은 지표 수학 표현식을 생성하면 됩니다.

```
{
  "MetricDataQueries": [
    {
      "Expression": "TIME_SERIES(AVG(ABS(m1-m2)/m1))",
      "Id": "e1",
      "Period": 3600,
      "Label": "MeanAbsolutePercentageError",
      "ReturnData": true
    },
    {
      "Id": "m1",
      "Label": "ActualLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/EC2",
          "MetricName": "CPUUtilization",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Sum"
      },
      "ReturnData": false
    },
    {
      "Id": "m2",
      "Label": "ForecastedLoadValues",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/AutoScaling",
          "MetricName": "PredictiveScalingLoadForecast",
          "Dimensions": [
            {
              "Name": "AutoScalingGroupName",
              "Value": "my-asg"
            },
            {
              "Name": "PolicyName",
              "Value": "my-predictive-scaling-policy"
```

```
            },
            {
              "Name": "PairIndex",
              "Value": "0"
            }
          ]
        },
        "Period": 3600,
        "Stat": "Average"
      },
      "ReturnData": false
    }
  ]
}
```

단일 지표 대신, `MetricDataQueries`의 지표 데이터 쿼리 구조의 배열을 사용할 수 있습니다. `MetricDataQueries`의 각 항목은 지표를 가져오거나 수학 표현식을 수행합니다. 첫 번째 항목 `e1`은 수학 표현식입니다. 이 지정된 표현식은 `ReturnData` 파라미터를 `true`로 설정합니다. 이는 궁극적으로 단일 시계열을 생성합니다. 다른 모든 지표의 경우, `ReturnData` 값은 `false`입니다.

이 예제에서 지정된 표현식은 실제 값과 예측 값을 입력으로 사용하고 새 지표 () 를 반환합니다. MAPE `m1`실제 로드 값을 포함하는 CloudWatch 지표입니다 (CPU사용률이 이름이 지정된 정책에 대해 원래 지정된 부하 지표라고 가정). `my-predictive-scaling-policy` `m2`예측된 CloudWatch 부하 값을 포함하는 지표입니다. MAPE지표의 수학 구문은 다음과 같습니다.

(abs ((실제 - 예측)/(실제)))의 평균

정확도 지표 시각화 및 경보 설정

정확도 지표 데이터를 시각화하려면 CloudWatch 콘솔에서 Metrics 탭을 선택합니다. 여기에서 데이터를 그래프로 표시할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 그래프에 수학 식 추가를 참조하십시오.

지표(Metrics) 섹션에서 모니터링하는 지표에 대해 경보를 설정할 수도 있습니다. 그래프로 표시된 지표(Graphed metrics) 탭에서 작업(Actions) 열 아래에 있는 경보 생성(Create alarm) 아이콘을 선택합니다. 이 경보 생성(Create alarm) 아이콘은 작은 종 모양으로 표시됩니다. 자세한 내용 및 알림 옵션은 Amazon CloudWatch User Guide의 지표 수학 식을 기반으로 CloudWatch 경보 생성 및 경보 변경 시 사용자에게 알리기를 참조하십시오.

또는 GetMetricData및 PutMetricAlarm를 사용하여 지표 수학을 사용하여 계산을 수행하고 출력을 기반으로 경보를 생성할 수 있습니다.

예약된 작업을 사용하여 예측 값 재정의

경우에 따라 예상 계산에서 고려할 수 없는 향후 애플리케이션 필요량에 대한 추가 정보가 있을 수 있습니다. 예컨대, 예상 계산은 향후 마케팅 이벤트에 필요한 용량을 과소 평가할 수 있습니다. 예약된 작업을 사용하여 미래 기간에 대한 예상을 임시로 재정의할 수 있습니다. 예약된 작업은 반복적으로 실행되거나 일회성 수요 변동이 있는 특정 날짜 및 시간에 실행될 수 있습니다.

예컨대, 예상한 것보다 높은 최소 용량으로 예약된 작업을 생성할 수 있습니다. Amazon EC2 Auto Scaling은 런타임 시 Auto Scaling 그룹의 최소 용량을 업데이트합니다. 예측 조정은 용량에 맞게 최적화하므로 최소 용량이 예상 값보다 높은 예약된 작업이 적용됩니다. 이렇게 하면 용량이 예상보다 작아지는 것을 방지할 수 있습니다. 예상 재정의를 중지하려면 두 번째 예약된 작업을 사용하여 최소 용량을 원래 설정으로 되돌립니다.

다음 절차에서는 미래 기간의 예상을 재정의하는 단계를 간략하게 설명합니다.

주제

- 1단계: (옵션) 시계열 데이터 분석
- 2단계: 2개의 예약된 작업 생성

> **Important**
>
> 이 주제에서는 예측보다 더 큰 용량으로 확장하기 위해 예측을 재정의하려고 한다고 가정합니다. 예측 조정 정책의 방해 없이 용량을 일시적으로 줄여야 하는 경우에는 예측 전용 모드를 대신 사용하십시오. 예측 전용 모드에서는 예측 규모 조정을 통해 계속해서 예측이 생성되지만 용량이 자동으로 증가하지는 않습니다. 그런 다음 리소스 사용률을 모니터링하고 필요에 따라 그룹 크기를 수동으로 줄일 수 있습니다. 수동 조정에 대한 자세한 내용은 을 참조하십시오Amazon EC2 Auto Scaling을 위한 수동 스케일링.

1단계: (옵션) 시계열 데이터 분석

예상 시계열 데이터 분석으로 시작합니다. 이 단계는 선택 사항이지만 예상의 세부 정보를 파악하려는 경우, 유용합니다.

1. 예상 검색

   예상이 생성되면 예상에서 특정 기간을 쿼리할 수 있습니다. 쿼리의 목표는 특정 기간에 대한 시계열 데이터의 전체 보기를 얻는 것입니다.

쿼리에는 최대 2일간의 미래 예상 데이터가 포함될 수 있습니다. 예측 조정을 잠시 사용한 경우에도 과거 예상 데이터에 액세스할 수 있습니다. 그러나 시작 시간과 해지 시간 사이의 최대 기간은 30일입니다.

get-predictive-scaling-forecast AWS CLI 명령을 사용하여 예측을 가져오려면 명령에 다음 파라미터를 제공하십시오.

- `--auto-scaling-group-name` 파라미터에 Auto Scaling 그룹의 이름을 입력합니다.
- `--policy-name` 파라미터에 정책 이름을 입력합니다.
- 지정한 시간 또는 그 이후의 예상 데이터만 반환하려면 `--start-time` 파라미터에 시작 시간을 입력합니다.
- 지정한 시간 이전의 예상 데이터만 반환하려면 `--end-time` 파라미터에 해지 시간을 입력합니다.

```
aws autoscaling get-predictive-scaling-forecast --auto-scaling-group-name my-asg \
    --policy-name cpu40-predictive-scaling-policy \
    --start-time "2021-05-19T17:00:00Z" \
    --end-time "2021-05-19T23:00:00Z"
```

이 작업이 성공하면 명령에서 다음 예와 비슷한 데이터가 반환됩니다.

```
{
    "LoadForecast": [
        {
            "Timestamps": [
                "2021-05-19T17:00:00+00:00",
                "2021-05-19T18:00:00+00:00",
                "2021-05-19T19:00:00+00:00",
                "2021-05-19T20:00:00+00:00",
                "2021-05-19T21:00:00+00:00",
                "2021-05-19T22:00:00+00:00",
                "2021-05-19T23:00:00+00:00"
            ],
            "Values": [
                153.0655799339254,
                128.8288551285919,
                107.1179447150675,
                197.3601844551528,
                626.4039934516954,
```

```
                596.9441277518481,
                677.9675713779869
            ],
            "MetricSpecification": {
                "TargetValue": 40.0,
                "PredefinedMetricPairSpecification": {
                    "PredefinedMetricType": "ASGCPUUtilization"
                }
            }
        }
    ],
    "CapacityForecast": {
        "Timestamps": [
            "2021-05-19T17:00:00+00:00",
            "2021-05-19T18:00:00+00:00",
            "2021-05-19T19:00:00+00:00",
            "2021-05-19T20:00:00+00:00",
            "2021-05-19T21:00:00+00:00",
            "2021-05-19T22:00:00+00:00",
            "2021-05-19T23:00:00+00:00"
        ],
        "Values": [
            2.0,
            2.0,
            2.0,
            2.0,
            4.0,
            4.0,
            4.0
        ]
    },
    "UpdateTime": "2021-05-19T01:52:50.118000+00:00"
}
```

응답에는 두 개의 예측 (`LoadForecast`및`CapacityForecast`) 이 포함됩니다. `LoadForecast`시간별 부하 예측을 보여줍니다. `CapacityForecast`40.0 (평균 사용률 40%) 을 유지하면서 예측된 로드를 처리하는 데 시간당 필요한 용량에 `TargetValue` 대한 예측 값을 표시합니다. CPU

2. 대상 기간 식별

   일회성 수요 변동이 발생해야 하는 시간을 식별합니다. 예측에 표시된 날짜와 시간은 현재 기준이라는 점을 기억하십시오. UTC

2단계: 2개의 예약된 작업 생성

다음으로, 애플리케이션의 예상 로드보다 높은 특정 기간에 대해 2개의 예약된 작업을 생성합니다. 예컨대, 제한된 기간에 사이트에 대한 트래픽을 높이는 마케팅 이벤트가 있는 경우, 이벤트 시작 시 최소 용량을 업데이트하는 일회성 작업을 예약할 수 있습니다. 그런 다음 이벤트가 해지 시 최소 용량을 원래 설정으로 되돌리도록 다른 작업을 예약합니다.

일회성 이벤트에 대해 2개의 예약된 작업을 생성하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 자동 조정 탭의 Scheduled actions(예약된 작업)에서 Create scheduled action(예약된 작업 생성)을 선택합니다.
4. 다음 예약된 작업 설정을 채웁니다.

   a. 예약된 작업의 이름을 입력합니다.

   b. Min(최소)에 Auto Scaling 그룹의 새 최소 용량을 입력합니다. Min(최소)은 그룹의 최대 크기보다 작거나 같아야 합니다. Min(최소)의 값이 그룹의 최대 크기보다 크면 Max(최대)를 업데이트해야 합니다.

   c. 반복에서 1회(Once)를 선택합니다.

   d. Time zone(표준 시간대)에서 시간대를 선택합니다. 시간대를 선택하지 않으면 `ETC/UTC`가 기본적으로 사용됩니다.

   e. Specific start time(특정 시작 시간)을 정의합니다.
5. Create(생성)를 선택합니다.

   콘솔에 Auto Scaling 그룹에 대해 예약된 작업이 표시됩니다.
6. 이벤트 해지 시 원래 설정으로 되돌아 가도록 두 번째 예약된 작업을 구성합니다. 예측 조정은 Min(최소)에 대해 사용자가 설정한 값이 예상 값보다 작은 경우에만 용량을 조정합니다.

일회성 이벤트에 대해 2개의 예약된 작업을 생성하려면(AWS CLI)

를 사용하여 예약된 작업을 AWS CLI 생성하려면 put-scheduled-update-group-action 명령을 사용하십시오.

예컨대, 5월 19일 오후 5시에 8시간 동안 최소 용량으로 인스턴스 3개를 유지하는 일정을 정의해 보겠습니다. 다음 명령은 이 시나리오를 구현하는 방법을 보여 줍니다.

첫 번째 put-scheduled-update-group-action 명령은 2021년 5월 UTC 19일 오후 5시에 지정된 EC2 Auto Scaling 그룹의 최소 용량을 업데이트하도록 Amazon Auto Scaling에 지시합니다.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-start \
  --auto-scaling-group-name my-asg --start-time "2021-05-19T17:00:00Z" --minimum-capacity 3
```

두 번째 명령은 2021년 5월 UTC 20일 오전 1시에 그룹의 최소 용량을 1명으로 설정하도록 Amazon EC2 Auto Scaling에 지시합니다.

```
aws autoscaling put-scheduled-update-group-action --scheduled-action-name my-event-end \
  --auto-scaling-group-name my-asg --start-time "2021-05-20T01:00:00Z" --minimum-capacity 1
```

이러한 예약된 작업을 Auto Scaling 그룹에 추가하면 Amazon EC2 Auto Scaling은 다음을 수행합니다.

- 2021년 5월 UTC 19일 오후 5시에 예약된 첫 번째 작업이 실행됩니다. 현재 해당 그룹에 인스턴스가 3개 미만 있는 경우, 그룹은 인스턴스 3개로 스케일 아웃됩니다. 예측 용량이 실제 용량보다 많거나 동적 조정 정책이 적용되는 경우 Amazon EC2 Auto Scaling은 이 기간 동안 이후 8시간 동안 계속해서 규모를 확장할 수 있습니다.
- 2021년 5월 UTC 20일 오전 1시에 예정된 두 번째 작업이 실행됩니다. 이렇게 하면 이벤트 해지 시 최소 용량이 원래 설정으로 돌아갑니다.

반복 일정에 따라 크기 조정

매주 동일한 기간에 대한 예상을 재정의하려면 두 개의 예약된 작업을 생성하고 cron 표현식을 사용하여 시간 및 날짜 로직을 제공합니다.

cron 표현식 형식은 다음과 같이 공백으로 구분된 다섯 개의 필드로 구성됩니다. [Minute] [Hour] [Day_of_Month] [Month_of_Year] [Day_of_Week]. 필드에는 특수 문자를 포함하여 허용되는 모든 값을 포함할 수 있습니다.

예컨대, 다음 cron 표현식은 매주 화요일 오전 6시 30분에 작업을 실행합니다. 별표는 필드의 모든 값을 일치시키기 위한 와일드카드로 사용됩니다.

```
30 6 * * 2
```

다음 사항도 참조하세요.

예약된 작업을 생성, 열거, 편집 및 삭제하는 방법에 대한 자세한 설명은 Amazon EC2 Auto Scaling을 위한 스케일링 예약 섹션을 참조하세요.

사용자 지정 지표를 사용하는 고급 예측 규모 조정 정책

예측적 조정 정책에서는 미리 정의된 지표나 사용자 정의 지표를 사용할 수 있습니다. 사용자 지정 지표는 사전 정의된 지표 (CPU, 네트워크 I/O, Application Load Balancer 요청 수) 가 애플리케이션 부하를 충분히 설명하지 못할 때 유용합니다.

사용자 지정 지표로 예측 조정 정책을 만들 때는 에서 제공하는 다른 CloudWatch 지표를 지정하거나 직접 정의하여 AWS게시할 지표를 지정할 수 있습니다. 또한 지표 수학을 사용하여 기존 지표를 집계하고 자동으로 AWS 추적되지 않는 새로운 시계열로 변환할 수 있습니다. 예컨대, 새 합계 또는 평균을 계산하여 데이터의 값을 결합하는 것을 집계라고 합니다. 결과 데이터를 집계라고 합니다.

다음 섹션에는 정책의 JSON 구조를 구성하는 방법에 대한 모범 사례와 예제가 포함되어 있습니다.

주제

- 모범 사례
- 사전 조건
- JSON사용자 지정 메트릭을 위한 구성
- 예측 규모 조정 정책의 사용자 지정 지표에 대한 고려 사항
- 제한 사항

모범 사례

다음 모범 사례는 사용자 정의 지표를 보다 효과적으로 사용하는 데 도움이 될 수 있습니다.

- 로드 지표 사양에서 가장 유용한 지표는 그룹의 용량에 관계없이 Auto Scaling 그룹 전체의 로드를 나타내는 지표입니다.
- 조정 지표 사양의 경우, 가장 유용한 지표는 인스턴스당 평균 처리량 또는 사용률 지표입니다.
- 조정 지표는 용량에 반비례해야 합니다. 즉, Auto Scaling 그룹의 인스턴스 수가 증가하면 조정 지표도 거의 같은 백분율로 감소해야 합니다. 예측적 조정이 예상대로 작동하게 하려면 로드 지표와 조정 지표도 서로 밀접한 상관 관계가 있어야 합니다.

- 목표 사용률은 조정 지표 유형과 일치해야 합니다. CPU사용률을 사용하는 정책 구성의 경우 이는 목표 백분율입니다. 요청 또는 메시지 수와 같이 처리량을 사용하는 정책 구성의 경우, 이는 1분 간격 동안 인스턴스당 요청 또는 메시지의 대상 수입니다.
- 이러한 권장 사항을 따르지 않으면 시계열의 예측된 미래 값이 정확하지 않을 수 있습니다. 데이터가 정확한지 검증하기 위해 Amazon EC2 Auto Scaling 콘솔에서 예측값을 확인할 수 있습니다. 또는 예측 규모 조정 정책을 생성한 후 를 호출하여 반환된 `LoadForecast` 및 `CapacityForecast` 객체를 검사할 수도 있습니다. GetPredictiveScalingForecastAPI
- 예측적 조정이 능동적으로 용량 조정을 시작하기 전에 예상을 평가할 수 있게 예상 전용 모드에서 예측적 조정을 구성하는 것이 좋습니다.

사전 조건

정책에서 사용자 지정 지표를 지정하려면 `cloudwatch:GetMetricData` 권한이 있어야 합니다.

AWS 제공하는 지표 대신 자체 지표를 지정하려면 먼저 지표를 에 게시해야 합니다. CloudWatch 자세한 내용은 Amazon CloudWatch 사용 설명서의 사용자 지정 지표 게시를 참조하십시오.

자체 지표를 게시하는 경우, 최소 5분 간격으로 데이터 요소를 게시해야 합니다. Amazon EC2 Auto Scaling은 필요한 기간을 CloudWatch 기준으로 데이터 포인트를 검색합니다. 예를 들어 부하 지표 사양에서는 시간별 지표를 사용하여 애플리케이션의 부하를 측정합니다. CloudWatch 게시된 지표 데이터를 사용하여 각 1시간 기간에 해당하는 타임스탬프를 사용하여 모든 데이터 포인트를 집계하여 1시간 동안 단일 데이터 값을 제공합니다.

JSON사용자 지정 메트릭을 위한 구성

다음 섹션에는 데이터를 쿼리하도록 예측 스케일링을 구성하는 방법에 대한 예제가 포함되어 있습니다. CloudWatch 이 옵션을 구성하는 방법은 두 가지가 있으며, 선택한 방법에 따라 예측 규모 조정 정책을 구성하는 데 사용하는 형식이 달라집니다. JSON 메트릭 수학을 사용하는 경우 수행 중인 메트릭 수학에 따라 형식이 더 JSON 달라집니다.

1. 에서 제공하는 다른 CloudWatch AWS 지표나 게시하려는 지표에서 직접 데이터를 가져오는 정책을 만들려면 을 CloudWatch 참조하십시오사용자 지정 로드 및 조정 지표가 있는 예측 조정 정책의 예 (AWS CLI).
2. 여러 CloudWatch 지표를 쿼리하고 수학 식을 사용하여 이러한 지표를 기반으로 새 시계열을 만들 수 있는 정책을 만들려면 을 참조하십시오지표 수학 표현식 사용.

사용자 지정 로드 및 조정 지표가 있는 예측 조정 정책의 예(AWS CLI)

를 사용하여 사용자 지정 부하 및 규모 조정 지표가 포함된 예측 규모 조정 정책을 만들려면 `--predictive-scaling-configuration` 에 대한 인수를 라는 JSON `config.json` 파일에 저장하십시오. AWS CLI

다음 예에서 교체 가능한 값을 지표 및 목표 사용률의 값으로 교체하여 사용자 지정 지표를 추가하기 시작합니다.

```
{
  "MetricSpecifications": [
    {
      "TargetValue": 50,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "scaling_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyUtilizationMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Average"
            }
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "MyLoadMetric",
                "Namespace": "MyNameSpace",
                "Dimensions": [
                  {
```

```
                    "Name": "MyOptionalMetricDimensionName",
                    "Value": "MyOptionalMetricDimensionValue"
                  }
                ]
              },
              "Stat": "Sum"
            }
          }
        ]
      }
    }
  ]
}
```

자세한 내용은 Amazon EC2 Auto Scaling API 레퍼런스를 참조하십시오 MetricDataQuery.

> **Note**
>
> 다음은 지표에 대한 CloudWatch 지표 이름, 네임스페이스, 차원 및 통계를 찾는 데 도움이 되는 몇 가지 추가 리소스입니다.
>
> - AWS 서비스에 사용할 수 있는 지표에 대한 자세한 내용은 Amazon CloudWatch User Guide 의 CloudWatch 지표를 게시하는AWS 서비스를 참조하십시오.
> - 가 포함된 지표의 정확한 지표 이름, 네임스페이스 및 차원 (해당하는 경우) 을 가져오려면 목록 CloudWatch 지표를 참조하십시오. AWS CLI

이 정책을 생성하려면 다음 예와 같이 JSON 파일을 입력으로 사용하여 put-scaling-policy명령을 실행합니다.

```
aws autoscaling put-scaling-policy --policy-name my-predictive-scaling-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
```

성공하면 이 명령은 정책의 Amazon 리소스 이름 (ARN) 을 반환합니다.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-predictive-scaling-policy",
  "Alarms": []
```

```
}
```

지표 수학 표현식 사용

다음 섹션에서는 정책에서 지표 수학을 사용하는 방법을 보여 주는 예측 조정 정책의 정보와 예를 제공합니다.

주제

- 지표 수학 이해
- 지표 수학을 사용하여 지표를 결합하는 예측 조정 정책의 예(AWS CLI)
- 블루/그린 배치 시나리오에서 사용할 예측 조정 정책의 예(AWS CLI)

지표 수학 이해

기존 지표 데이터를 집계하기만 하려는 경우 CloudWatch 메트릭 수학을 사용하면 다른 메트릭을 게시하는 데 CloudWatch 드는 노력과 비용을 절약할 수 있습니다. AWS 제공하는 모든 지표를 사용할 수 있으며 응용 프로그램의 일부로 정의한 지표도 사용할 수 있습니다. 예를 들어 인스턴스당 Amazon SQS 대기열 백로그를 계산하고 싶을 수 있습니다. 대기열에서 검색 가능한 대략적인 메시지 수를 가져와 Auto Scaling 그룹의 실행 용량으로 나누어 이를 수행할 수 있습니다.

자세한 내용은 Amazon 사용 CloudWatch 설명서의 지표 수학 사용을 참조하십시오.

예측적 조정 정책에서 지표 수학 표현식을 사용하기로 선택한 경우, 다음 사항을 고려하세요.

- 지표 수학 연산은 고유한 조합의 지표 이름, 네임스페이스 및 지표의 차원 키/값 페어의 데이터 요소를 사용합니다.
- 모든 산술 연산자 (+ - */^), 통계 함수 (예: AVG orSUM) 또는 지원하는 기타 함수를 사용할 수 있습니다. CloudWatch
- 수학 표현식의 공식에서 지표 및 다른 수학 표현식의 결과를 모두 사용할 수 있습니다.
- 지표 수학 표현식은 다양한 집계로 구성될 수 있습니다. 그러나 최종 집계 결과에서 조정 지표에는 `Average`를 사용하고 로드 지표에는 `Sum`을 사용하는 것이 가장 좋습니다.”
- 지표 사양에 사용된 표현식은 결국 단일 시계열을 반환해야 합니다.

지표 수학을 사용하려면 다음을 수행하십시오:

- 지표를 하나 이상 선택합니다. CloudWatch 그런 다음 표현식을 생성합니다. 자세한 내용은 Amazon 사용 CloudWatch 설명서의 지표 수학 사용을 참조하십시오.

- CloudWatch콘솔 또는 를 사용하여 메트릭 수학 표현식이 유효한지 확인하십시오 CloudWatch GetMetricDataAPI.

지표 수학을 사용하여 지표를 결합하는 예측 조정 정책의 예(AWS CLI)

지표를 직접 지정하는 대신 어떤 방식으로든 해당 데이터를 먼저 처리해야 하는 경우가 있습니다. 예를 들어 Amazon SQS 대기열에서 작업을 가져오는 애플리케이션이 있을 때 대기열에 있는 항목 수를 예측 규모 조정 기준으로 사용할 수 있습니다. 대기열의 메시지 수가 필요한 인스턴스 수를 단독으로 정의하지 않습니다. 따라서 인스턴스당 백로그를 계산하는 데 사용할 수 있는 지표를 생성하려면 더 많은 작업이 필요합니다. 자세한 설명은 아마존 기반 규모 조정 정책 SQS 섹션을 참조하세요.

다음은 이 시나리오에 대한 예측적 조정 정책의 예입니다. Amazon SQS `ApproximateNumberOfMessagesVisible` 지표 (대기열에서 검색할 수 있는 메시지 수) 를 기반으로 조정 및 로드 지표를 지정합니다. 또한 Amazon EC2 Auto Scaling `GroupInServiceInstances` 지표와 수학 식을 사용하여 조정 지표의 인스턴스당 백로그를 계산합니다.

```
aws autoscaling put-scaling-policy --policy-name my-sqs-custom-metrics-policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 100,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Label": "Get the queue size (the number of messages waiting to be
 processed)",
            "Id": "queue_size",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ]
              },
              "Stat": "Sum"
```

```
            },
            "ReturnData": false
          },
          {
            "Label": "Get the group size (the number of running instances)",
            "Id": "running_capacity",
            "MetricStat": {
              "Metric": {
                "MetricName": "GroupInServiceInstances",
                "Namespace": "AWS/AutoScaling",
                "Dimensions": [
                  {
                    "Name": "AutoScalingGroupName",
                    "Value": "my-asg"
                  }
                ]
              },
              "Stat": "Sum"
            },
            "ReturnData": false
          },
          {
            "Label": "Calculate the backlog per instance",
            "Id": "scaling_metric",
            "Expression": "queue_size / running_capacity",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_metric",
            "MetricStat": {
              "Metric": {
                "MetricName": "ApproximateNumberOfMessagesVisible",
                "Namespace": "AWS/SQS",
                "Dimensions": [
                  {
                    "Name": "QueueName",
                    "Value": "my-queue"
                  }
                ],
              },
```

```
                "Stat": "Sum"
              },
              "ReturnData": true
            }
          ]
        }
      }
    ]
}
```

이 예제는 정책을 반환합니다. ARN

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-sqs-custom-metrics-policy",
  "Alarms": []
}
```

블루/그린 배치 시나리오에서 사용할 예측 조정 정책의 예(AWS CLI)

검색 표현식은 여러 Auto Scaling 그룹에서 지표를 쿼리하고 이에 대해 수학 표현식을 수행할 수 있는 고급 옵션을 제공합니다. 이는 특히 블루/그린 배치에 유용합니다.

> **Note**
> 블루/그린 배치는 두 개의 별개이지만 동일한 Auto Scaling 그룹을 생성하는 배치 방법입니다. 그룹 중 하나만 프로덕션 트래픽을 수신합니다. 사용자 트래픽은 초기에 이전("블루") Auto Scaling 그룹으로 전달되는 반면 새 그룹("그린")은 새 버전의 애플리케이션 또는 서비스를 테스트하고 평가하는 데 사용됩니다. 새 배치가 테스트되고 수락되면 사용자 트래픽이 그린 Auto Scaling 그룹으로 이동됩니다. 그런 다음 배치가 성공한 후 블루 그룹을 삭제할 수 있습니다.

새로운 Auto Scaling 그룹이 블루/그린 배치의 일부로 생성되면 해당 지표 사양을 변경하지 않고도 각 그룹의 지표 기록이 예측적 조정 정책에 자동으로 포함될 수 있습니다. 자세한 내용은 Compute 블로그에서 블루/그린 배포에 Auto Scaling 예측 조정 정책 사용을 EC2 참조하십시오. AWS

다음 예 정책은 이를 수행하는 방법을 보여줍니다. 이 예제에서 정책은 EC2 Amazon 에서 `CPUUtilization` 생성한 지표를 사용합니다. Amazon EC2 Auto Scaling `GroupInServiceInstances` 지표와 수학 식을 사용하여 인스턴스당 조정 지표의 값을 계산합니다. 또한 `GroupInServiceInstances` 지표를 가져오기 위한 용량 지표 사양을 지정합니다.

검색 표현식은 지정된 검색 기준에 따라 여러 Auto Scaling 그룹에서 인스턴스의 `CPUUtilization`을 찾습니다. 나중에 동일한 검색 기준과 일치하는 새 Auto Scaling 그룹을 생성하면 새 Auto Scaling 그룹에 있는 인스턴스의 `CPUUtilization`이 자동으로 포함됩니다.

```
aws autoscaling put-scaling-policy --policy-name my-blue-green-predictive-scaling-
policy \
  --auto-scaling-group-name my-asg --policy-type PredictiveScaling \
  --predictive-scaling-configuration file://config.json
{
  "MetricSpecifications": [
    {
      "TargetValue": 25,
      "CustomizedScalingMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 300))",
            "ReturnData": false
          },
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))",
            "ReturnData": false
          },
          {
            "Id": "weighted_average",
            "Expression": "load_sum / capacity_sum",
            "ReturnData": true
          }
        ]
      },
      "CustomizedLoadMetricSpecification": {
        "MetricDataQueries": [
          {
            "Id": "load_sum",
            "Expression": "SUM(SEARCH('{AWS/EC2,AutoScalingGroupName} MetricName=
\"CPUUtilization\" ASG-myapp', 'Sum', 3600))"
          }
        ]
      },
      "CustomizedCapacityMetricSpecification": {
```

```
        "MetricDataQueries": [
          {
            "Id": "capacity_sum",
            "Expression": "SUM(SEARCH('{AWS/AutoScaling,AutoScalingGroupName}
 MetricName=\"GroupInServiceInstances\" ASG-myapp', 'Average', 300))"
          }
        ]
      }
    }
  ]
}
```

이 예제는 정책을 반환합니다ARN.

```
{
  "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:2f4f5048-d8a8-4d14-
b13a-d1905620f345:autoScalingGroupName/my-asg:policyName/my-blue-green-predictive-
scaling-policy",
  "Alarms": []
}
```

예측 규모 조정 정책의 사용자 지정 지표에 대한 고려 사항

사용자 정의 지표를 사용하는 동안 문제가 발생하면 다음을 수행하는 것이 좋습니다.

- 오류 메시지가 제공되면 메시지를 읽고 가능한 경우, 보고된 문제를 해결하세요.
- 블루/그린 배치 시나리오에서 검색 표현식을 사용하려고 할 때 문제가 발생하는 경우, 먼저 정확히 일치 대신 부분 일치를 찾는 검색 표현식을 생성하는 방법을 알고 있어야 합니다. 또한 쿼리가 특정 애플리케이션을 실행 중인 Auto Scaling 그룹만 찾는지 확인합니다. 검색 표현식 구문에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 CloudWatch 검색 표현식 구문을 참조하십시오.
- 표현식을 미리 검증하지 않은 경우 조정 정책을 생성할 때 put-scaling-policy명령이 표현식의 유효성을 검사합니다. 그러나 이 명령이 발견된 오류의 정확한 원인을 식별하지 못할 수도 있습니다. 문제를 해결하려면 명령에 대한 요청의 응답에서 발생하는 오류를 해결하세요. get-metric-data 콘솔에서 표현식 문제를 해결할 수도 있습니다. CloudWatch
- 콘솔에서 로드(Load) 및 용량(Capacity) 그래프를 볼 때 용량(Capacity) 그래프에 데이터가 표시되지 않을 수 있습니다. 그래프에 완전한 데이터가 있는지 확인하려면 Auto Scaling 그룹에 대해 그룹 지표를 일관되게 사용 설정해야 합니다. 자세한 설명은 Auto Scaling 그룹 지표 활성화(콘솔) 섹션을 참조하세요.

- 용량 지표 사양은 수명 동안 서로 다른 Auto Scaling 그룹에서 실행되는 애플리케이션이 있는 경우, 블루/그린 배치에만 유용합니다. 이 사용자 정의 지표를 사용하면 여러 Auto Scaling 그룹의 총 용량을 제공할 수 있습니다. 예측적 조정은 이를 사용하여 콘솔의 용량(Capacity) 그래프에 기록 데이터를 표시합니다.
- `falsefor ReturnData MetricDataQueries` if는 SEARCH () 와 같은 수학 함수 없이 SUM () 함수를 자체적으로 지정해야 합니다. 이는 검색 표현식이 여러 시계열을 반환할 수 있고 표현식에 근거하여 하는 지표 사양이 하나의 시계열만 반환할 수 있기 때문입니다.
- 검색 표현식과 관련된 모든 지표는 해상도가 동일해야 합니다.

제한 사항

- 하나의 지표 사양에서 최대 10개 지표의 데이터 요소를 쿼리할 수 있습니다.
- 이 제한을 위해 하나의 표현식이 하나의 지표로 계산됩니다.

축소 시 해지할 Auto Scaling 인스턴스 제어

Amazon EC2 Auto Scaling은 종료 정책을 사용하여 인스턴스 종료 순서를 결정합니다. 사전 정의된 정책을 사용하거나 특정 요구 사항에 맞게 사용자 지정 정책을 생성할 수 있습니다. 사용자 지정 정책 또는 인스턴스 확장 보호를 사용하면 Auto Scaling 그룹이 아직 종료할 준비가 되지 않은 인스턴스를 종료하지 못하도록 방지할 수도 있습니다.

내용

- Amazon EC2 Auto Scaling에서 종료 정책을 사용하는 경우
- Amazon EC2 Auto Scaling에 대한 종료 정책을 구성합니다.
- Lambda를 사용하여 사용자 지정 해지 정책 생성
- 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다.
- 인스턴스 종료를 정상적으로 처리하도록 애플리케이션을 설계하십시오.

Amazon EC2 Auto Scaling에서 종료 정책을 사용하는 경우

다음 섹션에서는 Amazon EC2 Auto Scaling에서 종료 정책을 사용하는 시나리오를 설명합니다.

내용

- 스케일 인 이벤트

- 인스턴스 새로 고침
- 가용 영역 재조정

스케일 인 이벤트

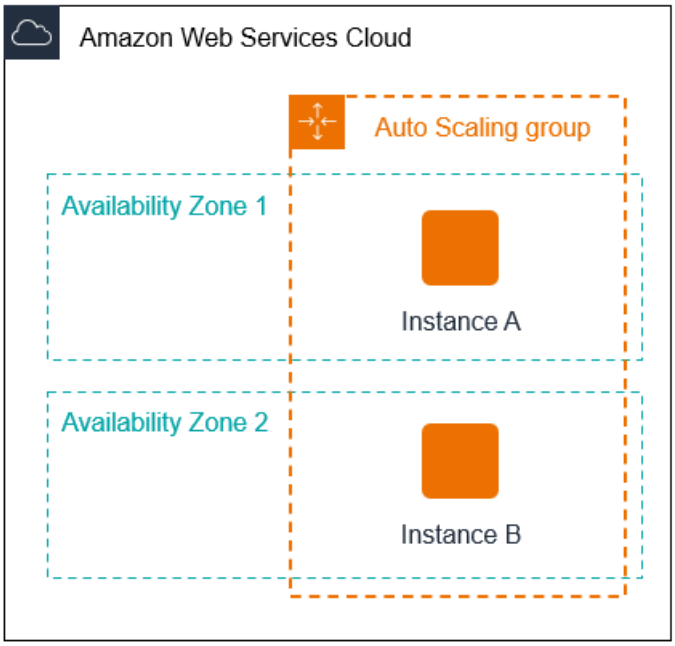
Auto Scaling 그룹의 원하는 용량이 그룹의 현재 용량보다 낮은 새 값이 있을 때 축소 이벤트가 발생합니다.

스케일 인 이벤트는 다음과 같은 시나리오에서 발생합니다.

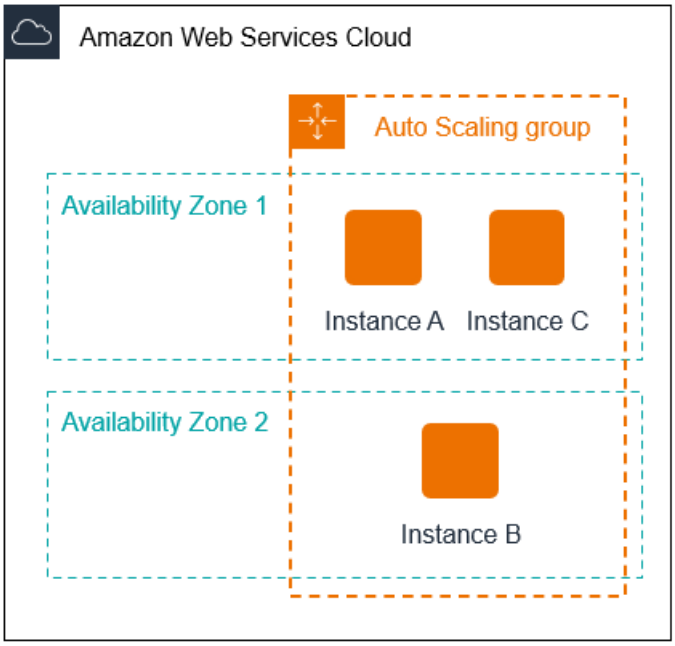
- 동적 조정 정책을 사용할 때 지표 값의 변화로 그룹 크기가 줄어드는 경우
- 예약된 조정을 사용할 때 예약된 작업의 결과로 그룹 크기가 줄어드는 경우
- 그룹의 크기를 수동으로 줄이는 경우

다음 예시는 규모 조정 이벤트가 있을 때 종료 정책이 어떻게 작동하는지 보여줍니다.

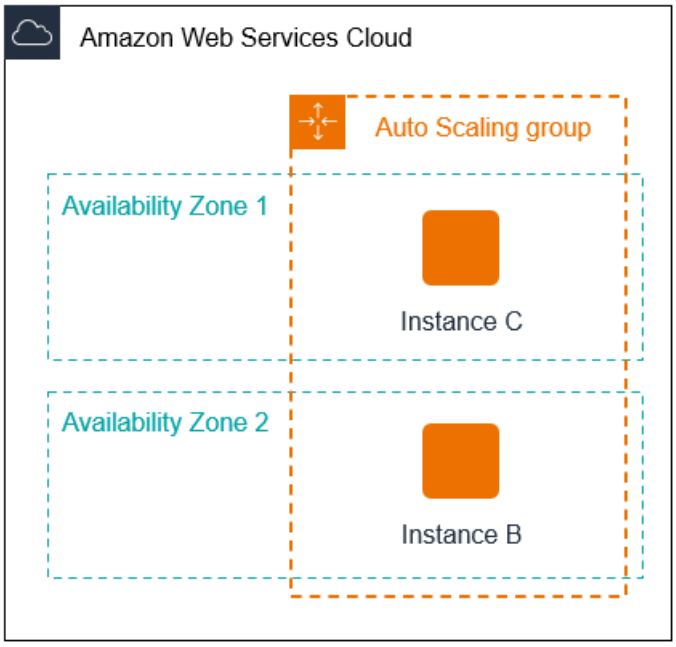
1. 이 예의 Auto Scaling 그룹에는 인스턴스 유형 1개와 가용 영역 2개가 있고 원하는 용량은 인스턴스 2개입니다. 또한 리소스 사용률이 증가하거나 감소할 때 인스턴스를 추가 및 제거하는 동적 조정 정책이 있습니다. 이 그룹에 있는 인스턴스 2개는 다음 다이어그램에 표시된 것처럼 가용 영역 2개에 분산됩니다.



2. Auto Scaling 그룹이 확장되면 Amazon EC2 Auto Scaling에서 새 인스턴스를 시작합니다. 이제 Auto Scaling 그룹에 있는 인스턴스 3개가 다음 다이어그램에 표시된 것처럼 가용 영역 2개에 분산됩니다.



3. Auto Scaling 그룹이 축소되면 Amazon EC2 Auto Scaling은 인스턴스 중 하나를 종료합니다.
4. 그룹에 특정 종료 정책을 할당하지 않은 경우 Amazon EC2 Auto Scaling은 기본 종료 정책을 사용합니다. 두 개의 인스턴스가 있는 가용 영역을 선택하고 시작 구성, 다른 시작 템플릿 또는 현재 시작 템플릿의 가장 오래된 버전에서 시작된 인스턴스를 종료합니다. 인스턴스가 동일한 시작 템플릿 및 버전에서 시작된 경우 Amazon EC2 Auto Scaling은 다음 청구 시간과 가장 가까운 인스턴스를 선택하여 종료합니다.



인스턴스 새로 고침

인스턴스 새로 고침을 시작하여 Auto Scaling 그룹의 인스턴스를 업데이트할 수 있습니다. 인스턴스 새로 고침 중에 Amazon EC2 Auto Scaling은 그룹의 인스턴스를 종료한 다음 종료된 인스턴스에 대한 대체 인스턴스를 시작합니다. Auto Scaling 그룹의 해지 정책은 먼저 교체되는 인스턴스를 제어합니다.

가용 영역 재조정

Amazon EC2 Auto Scaling은 Auto Scaling 그룹에 활성화된 가용 영역 전체에 걸쳐 용량을 균등하게 조정합니다. 이를 통해 가용 영역 중단으로 인한 영향을 줄일 수 있습니다. 가용 영역 간 용량 분배의 균형이 맞지 않을 경우 Amazon EC2 Auto Scaling은 활성화된 가용 영역에서 가장 적은 수의 인스턴스로 인스턴스를 시작하고 다른 곳에서 인스턴스를 종료하여 Auto Scaling 그룹의 균형을 재조정합니다. 해지 정책은 어떤 인스턴스를 가장 먼저 해지할지 우선순위를 제어합니다.

여러 가용 영역에 걸쳐 인스턴스 배치의 균형이 깨지는 데에는 여러 가지 이유가 있습니다.

인스턴스 제거

Auto Scaling 그룹에서 인스턴스를 분리하거나, 인스턴스를 대기 상태로 전환하거나, 인스턴스를 명시적으로 해지하고 원하는 용량을 줄이면 교체 인스턴스가 시작되지 않으므로 그룹의 균형이 깨질 수 있습니다. 이 경우 Amazon EC2 Auto Scaling은 가용 영역을 재조정하여 이를 보완합니다.

원래 지정한 가용 영역과 다른 가용 영역 사용

Auto Scaling 그룹을 확장하여 추가 가용 영역을 포함하거나 사용할 가용 영역을 변경하는 경우, Amazon EC2 Auto Scaling은 새 가용 영역에서 인스턴스를 시작하고 다른 영역의 인스턴스를 종료하여 Auto Scaling 그룹이 가용 영역에 균등하게 확장되도록 합니다.

가용성 중단

가용성 중단은 거의 발생하지 않습니다. 그러나 하나의 가용 영역이 사용할 수 없게 되어 나중에 복구되면 Auto Scaling 그룹이 가용 영역 간에 불균형하게 분배될 수 있습니다. Amazon EC2 Auto Scaling은 그룹을 점진적으로 재조정하려고 시도하므로 재조정을 통해 다른 영역의 인스턴스가 종료될 수 있습니다.

인스턴스 유형이 1개, 가용 영역이 2개가 있고, 원하는 용량이 인스턴스 2개인 Auto Scaling 그룹이 있는 경우를 예로 들어 보겠습니다. 한 가용 영역에 장애가 발생하는 경우 Amazon EC2 Auto Scaling은 정상 가용 영역에서 새 인스턴스를 자동으로 시작하여 비정상 가용 영역에 있는 인스턴스를 대체합니다. 그런 다음 나중에 비정상 가용 영역이 정상 상태로 돌아오면 Amazon EC2 Auto Scaling은 이 영역에서 새 인스턴스를 자동으로 시작하고, 이 인스턴스는 영향을 받지 않는 영역의 인스턴스를 종료합니다.

> **Note**
>
> 재조정 시 Amazon EC2 Auto Scaling은 이전 인스턴스를 종료하기 전에 새 인스턴스를 시작하므로 재조정으로 인해 애플리케이션의 성능이나 가용성이 저하되지 않습니다.
> Amazon EC2 Auto Scaling은 이전 인스턴스를 종료하기 전에 새 인스턴스를 시작하려고 시도하기 때문에 지정된 최대 용량에 도달하거나 근처에 있으면 재조정 작업이 방해되거나 완전히 중단될 수 있습니다. 이 문제를 피하기 위해 시스템에서는 재조정 활동 중에 그룹의 지정된 최대 용량을 10% 여유(또는 인스턴스 1개의 여유 중 큰 쪽)만큼 일시적으로 초과할 수 있습니다. 그룹이 최대 용량에 도달하거나 이에 근접하여 재조정가 필요한 경우, 또는 사용자 요청에 의한 영역 재조정이나 영역 가용성 문제에 대처하기 위한 경우에만 이 여유분만큼 스케일 아웃됩니다. 스케일 아웃은 그룹에 재조정가 필요한 동안에만 유지됩니다.

Amazon EC2 Auto Scaling에 대한 종료 정책을 구성합니다.

종료 정책은 Amazon EC2 Auto Scaling이 특정 순서로 인스턴스를 종료하기 위해 따르는 기준을 제공합니다. 기본적으로 Amazon EC2 Auto Scaling은 오래된 구성을 사용하는 인스턴스를 먼저 종료하도록 설계된 종료 정책을 사용합니다. 종료 정책을 변경하여 가장 먼저 종료하는 것이 가장 중요한 인스턴스를 제어할 수 있습니다.

Amazon EC2 Auto Scaling이 인스턴스를 종료하면 Auto Scaling 그룹에 활성화된 가용 영역 전반의 균형을 유지하려고 합니다. 영역 균형을 유지하는 것이 종료 정책보다 우선합니다. 한 가용 영역에 다른 가용 영역보다 많은 인스턴스가 있는 경우 Amazon EC2 Auto Scaling은 먼저 불균형 영역에 종료 정책을 적용합니다. 가용 영역이 균형을 이루면 종료 정책이 모든 영역에 적용됩니다.

주제

- 기본 종료 정책의 작동 방식
- 기본 해지 정책 및 혼합 인스턴스 그룹
- 사전 정의된 종료 정책
- Auto Scaling 그룹의 종료 정책 변경

기본 종료 정책의 작동 방식

Amazon EC2 Auto Scaling은 인스턴스를 종료해야 하는 경우 먼저 인스턴스가 가장 많고 확장으로부터 보호되지 않는 인스턴스가 하나 이상 있는 가용 영역 (또는 영역) 을 식별합니다. 그런 다음 다음과 같이 식별된 가용 영역 내의 보호되지 않는 인스턴스를 평가합니다.

오래된 구성을 사용하는 인스턴스

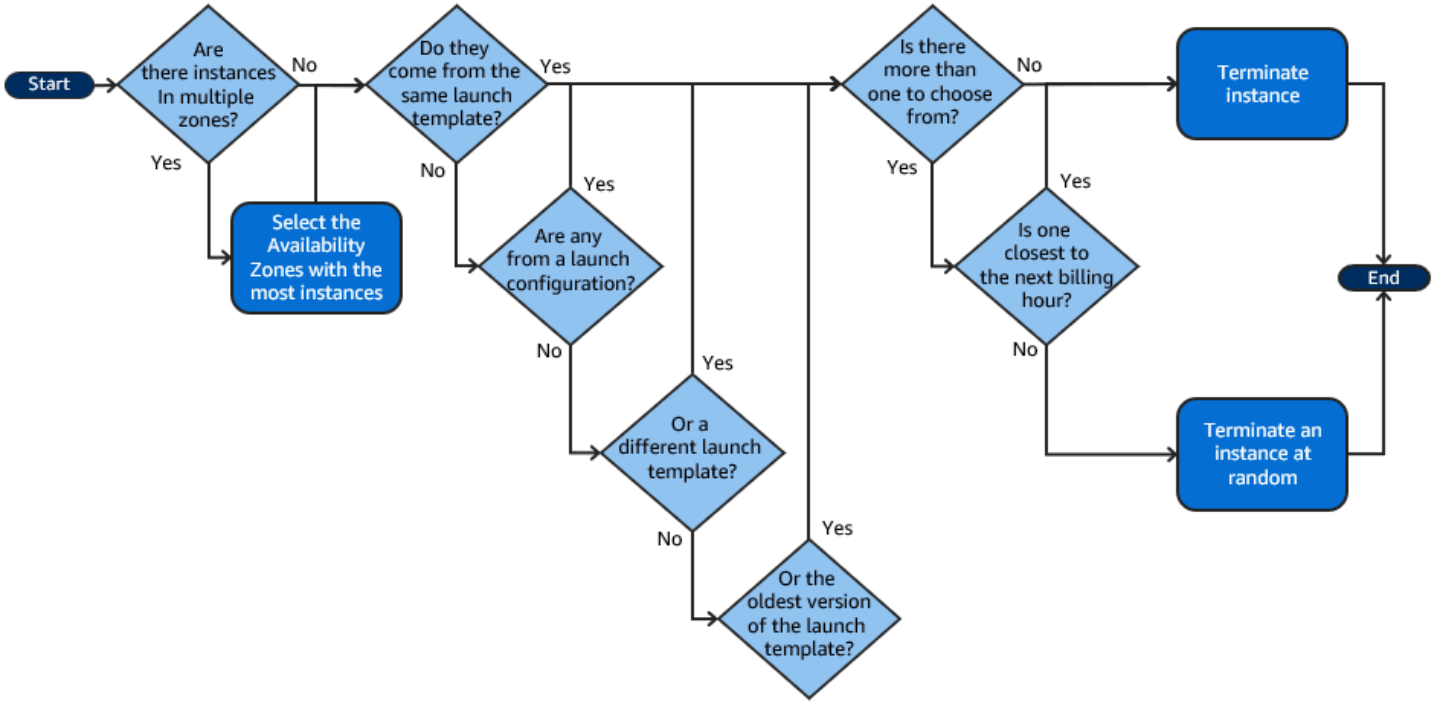
- 시작 템플릿을 사용하는 그룹의 경우 — 다음 순서대로 우선 순위를 지정하여 오래된 구성을 사용하는 인스턴스가 있는지 확인합니다.
  1. 먼저 시작 구성으로 시작된 인스턴스가 있는지 확인합니다.
  2. 그런 다음 현재 시작 템플릿 대신 다른 시작 템플릿을 사용하여 시작된 인스턴스가 있는지 확인합니다.
  3. 마지막으로 현재 시작 템플릿의 가장 오래된 버전을 사용하는 인스턴스가 있는지 확인합니다.
- 시작 구성을 사용하는 그룹의 경우 — 인스턴스 중 가장 오래된 시작 구성을 사용하는 인스턴스가 있는지 확인합니다.

구성이 오래된 인스턴스가 없거나 선택할 수 있는 인스턴스가 여러 개 있는 경우, Amazon EC2 Auto Scaling은 다음 청구 시간이 가까워지면 다음 인스턴스 기준을 고려합니다.

다음 청구 시간이 다가오고 있는 인스턴스

이전 기준을 충족하는 인스턴스 중 다음 청구 시간과 가장 근접한 인스턴스가 있는지 확인하세요. 여러 인스턴스가 비슷하게 비슷한 경우 하나를 무작위로 종료하세요. 이렇게 하면 시간당 요금이 청구되는 인스턴스 사용을 극대화할 수 있습니다. 하지만 이제 대부분의 EC2 사용량은 초당 요금이 청구되므로 이 최적화를 통해 얻을 수 있는 혜택은 줄어듭니다. 자세한 내용은 Amazon EC2 요금을 참조하십시오.

다음 흐름도는 시작 템플릿을 사용하는 그룹에 대한 기본 종료 정책의 작동 방식을 보여줍니다.



기본 해지 정책 및 혼합 인스턴스 그룹

Amazon EC2 Auto Scaling은 혼합 인스턴스 그룹에서 인스턴스를 종료할 때 추가 기준을 적용합니다.

Amazon EC2 Auto Scaling은 인스턴스를 종료해야 하는 경우 먼저 그룹 설정에 따라 종료해야 할 구매 옵션 (스팟 또는 온디맨드) 을 식별합니다. 이렇게 하면 시간이 지남에 따라 그룹이 스팟 및 온디맨드 인스턴스의 지정된 비율을 향해 나아가는 추세를 확인할 수 있습니다.

그런 다음 각 가용 영역 내에 종료 정책을 독립적으로 적용합니다. 가용 영역의 균형을 유지하기 위해 어떤 스팟 또는 온디맨드 인스턴스가 어느 가용 영역에서 종료될지 결정합니다. 인스턴스 유형에 대해 가중치가 정의된 혼합 인스턴스 그룹에도 동일한 논리가 적용됩니다.

각 영역 내에서 기본 종료 정책은 다음과 같이 작동하여 식별된 구매 옵션 내에서 종료할 수 있는 보호되지 않는 인스턴스를 결정합니다.

1. Auto Scaling 그룹에 지정된 할당 전략에 맞게 조정하기 위해 인스턴스를 종료할 수 있는지 여부를 결정합니다. 최적화할 인스턴스가 식별되지 않았거나 선택할 수 있는 인스턴스가 여러 개 있는 경우 평가가 계속됩니다.
2. 오래된 구성을 사용하는 인스턴스가 있는지 확인하고 다음 순서대로 우선 순위를 지정하십시오.
   a. 먼저 시작 구성으로 시작된 인스턴스가 있는지 확인합니다.

b. 그런 다음 현재 시작 템플릿 대신 다른 시작 템플릿을 사용하여 시작된 인스턴스가 있는지 확인합니다.

c. 마지막으로 현재 시작 템플릿의 가장 오래된 버전을 사용하는 인스턴스가 있는지 확인합니다.

구성이 오래된 인스턴스가 없거나 선택할 수 있는 인스턴스가 여러 개 있는 경우 평가가 계속됩니다.

3. 다음 청구 시간과 가장 가까운 인스턴스가 있는지 확인하세요. 여러 인스턴스가 똑같이 비슷한 경우 하나를 무작위로 선택하십시오.

사전 정의된 종료 정책

다음과 같은 사전 정의된 해지 정책 중에서 선택할 수 있습니다.

- **`Default`**— 기본 종료 정책에 따라 인스턴스를 종료합니다.
- **`AllocationStrategy`**— Auto Scaling 그룹의 인스턴스를 종료하여 종료되는 인스턴스 유형 (스팟 인스턴스 또는 온디맨드 인스턴스) 에 대한 할당 전략에 따라 나머지 인스턴스를 정렬합니다. 이 정책은 선호하는 인스턴스 유형이 변경된 경우, 유용합니다. 스팟 할당 전략이 `lowest-price`인 경우, 최저가 스팟 풀 N개 전체에서 스팟 인스턴스 배치를 점차 재조정할 수 있습니다. 스팟 할당 전략이 `capacity-optimized`인 경우, 사용 가능한 스팟 용량이 더 여유 있는 스팟 풀에서 스팟 인스턴스 배치를 점차 재조정할 수 있습니다. 또한 점차 우선순위가 낮은 유형의 온디맨드 인스턴스를 우선순위가 높은 유형의 온디맨드 인스턴스로 교체할 수 있습니다.
- **`OldestLaunchTemplate`**— 가장 오래된 시작 템플릿이 있는 인스턴스를 종료합니다. 이 정책을 사용하는 경우, 최신이 아닌 시작 템플릿을 사용하는 인스턴스가 먼저 해지되고 나서 최신 시작 템플릿의 가장 오래된 버전을 사용하는 인스턴스가 해지됩니다. 이 정책은 그룹을 업데이트하고 이전 구성에서 인스턴스를 단계적으로 해지할 때 유용합니다.
- **`OldestLaunchConfiguration`**— 가장 오래된 시작 구성을 가진 인스턴스를 종료합니다. 이 정책은 그룹을 업데이트하고 이전 구성에서 인스턴스를 단계적으로 해지할 때 유용합니다. 이 정책을 사용하는 경우, 최신이 아닌 시작 구성을 사용하는 인스턴스가 먼저 해지됩니다.
- **`ClosestToNextInstanceHour`**— 다음 청구 시간과 가장 가까운 인스턴스를 종료합니다. 이 정책은 시간제로 요금이 청구되는 인스턴스의 사용을 극대화할 수 있도록 합니다.
- **`NewestInstance`**— 그룹의 최신 인스턴스를 종료합니다. 이 정책은 새로운 시작 구성을 테스트하지만 프로덕션 상태로 유지하고 싶지 않은 경우에 유용합니다.
- **`OldestInstance`**— 그룹에서 가장 오래된 인스턴스를 종료합니다. 이 옵션은 Auto Scaling 그룹의 인스턴스를 새 EC2 인스턴스 유형으로 업그레이드할 때 유용합니다. 따라서 이전 유형의 인스턴스를 새로운 유형의 인스턴스로 점진적으로 교체할 수 있습니다.

> **Note**
>
> Amazon EC2 Auto Scaling은 어떤 종료 정책을 사용하든 항상 가용 영역 전반에서 먼저 인스턴스를 밸런싱합니다. 따라서 일부 최신 인스턴스가 이전 인스턴스보다 먼저 해지되는 상황이 발생할 수 있습니다. 최근에 추가된 가용 영역이 있거나 가용 영역에 그룹에서 사용하는 다른 가용 영역 외에 추가 인스턴스가 있는 경우를 예로 들 수 있습니다.

Auto Scaling 그룹의 종료 정책 변경

Auto Scaling 그룹의 종료 정책을 변경하려면 다음 방법 중 하나를 사용하십시오.

Console

Amazon EC2 Auto Scaling 콘솔에서 Auto Scaling 그룹을 처음 생성할 때는 종료 정책을 변경할 수 없습니다. 기본 해지 정책이 자동으로 사용됩니다. Auto Scaling 그룹이 생성되면 기본 정책을 다른 종료 정책 또는 적용되는 순서대로 나열된 여러 종료 정책으로 바꿀 수 있습니다.

Auto Scaling 그룹의 종료 정책을 변경하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 고급 구성, 편집을 선택합니다.
4. 해지 정책에서 하나 이상의 해지 정책을 선택합니다. 여러 정책을 선택하는 경우, 평가하려는 순서대로 정책을 배치합니다.

   필요에 따라 사용자 정의 해지 정책(Custom termination policy)을 선택한 다음 필요에 맞는 Lambda 함수를 선택할 수 있습니다. Lambda 함수에 대한 버전 및 별칭을 생성한 경우, 버전/별칭(Version/Alias) 드롭다운에서 버전 또는 별칭을 선택할 수 있습니다. 게시되지 않은 버전의 Lambda 함수를 사용하려면 버전/별칭(Version/Alias)을 기본값으로 설정된 상태로 둡니다. 자세한 설명은 Lambda를 사용하여 사용자 지정 해지 정책 생성 섹션을 참조하세요.

   > **Note**
   >
   > 여러 정책을 사용하는 경우, 해당 순서를 올바르게 설정해야 합니다.

- 기본(Default) 정책을 사용할 경우, 목록에서 기본 정책이어야 합니다.
- 사용자 정의 해지 정책(Custom termination policy)을 사용하는 경우, 목록의 첫 번째 정책이어야 합니다.

5. 업데이트를 선택합니다.

AWS CLI

다른 정책을 지정하지 않으면 기본 해지 정책이 자동으로 사용됩니다.

Auto Scaling 그룹의 종료 정책을 변경하려면

다음 명령 중 하나를 사용합니다.

- create-auto-scaling-group
- update-auto-scaling-group

해지 정책은 개별적으로 사용하거나 정책 목록으로 결합할 수 있습니다. 예컨대, 다음 명령으로 Auto Scaling 그룹을 업데이트하여 먼저 `OldestLaunchConfiguration` 정책을 사용한 다음 `ClosestToNextInstanceHour` 정책을 사용합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --
termination-policies "OldestLaunchConfiguration" "ClosestToNextInstanceHour"
```

`Default` 해지 정책을 사용하는 경우, 이 정책이 해지 정책 목록의 맨 마지막에 오도록 합니다. 예: `--termination-policies "OldestLaunchConfiguration" "Default"`.

사용자 지정 종료 정책을 사용하려면 먼저 를 사용하여 종료 정책을 생성해야 AWS Lambda합니다. 해지 정책으로 사용할 Lambda 함수를 지정하려면 해지 정책 목록의 첫 번째 항목을 지정합니다. 예: `--termination-policies "arn:aws:lambda:us-west-2:123456789012:function:HelloFunction:prod" "OldestLaunchConfiguration"`. 자세한 설명은 Lambda를 사용하여 사용자 지정 해지 정책 생성 섹션을 참조하세요.

Lambda를 사용하여 사용자 지정 해지 정책 생성

Amazon EC2 Auto Scaling은 종료 정책을 사용하여 Auto Scaling 그룹의 크기를 줄일 때 (축소라고 함) 먼저 종료할 인스턴스의 우선 순위를 정합니다. Auto Scaling 그룹은 기본 해지 정책을 사용하지만, 사용자는 자체 해지 정책을 선택하거나 생성할 수 있습니다. 미리 정의된 해지 정책을 선택하는 방법에 대한 자세한 설명은 Amazon EC2 Auto Scaling에 대한 종료 정책을 구성합니다. 섹션을 참조하세요.

이 주제에서는 Amazon EC2 Auto Scaling에서 특정 이벤트에 대한 응답으로 호출하는 AWS Lambda 함수를 사용하여 사용자 지정 종료 정책을 생성하는 방법을 알아봅니다. 생성한 Lambda 함수는 Amazon Auto EC2 Scaling에서 전송한 입력 데이터의 정보를 처리하고 종료 준비가 된 인스턴스 목록을 반환합니다.

사용자 지정 해지 정책은 해지되는 인스턴스와 해지 시점을 보다 효과적으로 제어할 수 있습니다. 예를 들어, Auto Scaling 그룹이 축소되는 경우 Amazon EC2 Auto Scaling은 중단되어서는 안 되는 워크로드가 실행 중인지 여부를 판단할 수 없습니다. Lambda 함수를 사용하면 종료 요청을 검증하고 워크로드가 완료될 때까지 기다린 후 종료를 위해 Amazon Auto EC2 Scaling에 인스턴스 ID를 반환할 수 있습니다.

내용

- 입력 데이터
- 응답 데이터
- 고려 사항
- Lambda 함수 생성
- 제한 사항

입력 데이터

Amazon EC2 Auto Scaling은 이벤트 축소에 대한 JSON 페이로드를 생성하며, 최대 인스턴스 수명 또는 인스턴스 새로 고침 기능으로 인해 인스턴스가 종료되려고 할 때도 페이로드를 생성합니다. 또한 가용 영역 전반에서 그룹을 재조정할 때 시작할 수 있는 이벤트 규모에 대한 JSON 페이로드를 생성합니다.

이 페이로드에는 Amazon EC2 Auto Scaling이 종료하는 데 필요한 용량, 종료를 제안하는 인스턴스 목록, 종료를 시작한 이벤트에 대한 정보가 들어 있습니다.

다음은 페이로드 예입니다.

```
{
```

```
  "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:<account-
id>:autoScalingGroup:d4738357-2d40-4038-ae7e-b00ae0227003:autoScalingGroupName/my-asg",
  "AutoScalingGroupName": "my-asg",
  "CapacityToTerminate": [
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 2,
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1b",
      "Capacity": 1,
      "InstanceMarketOption": "spot"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "Capacity": 3,
      "InstanceMarketOption": "on-demand"
    }
  ],
  "Instances": [
    {
      "AvailabilityZone": "us-east-1b",
      "InstanceId": "i-0056faf8da3e1f75d",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-02e1c69383a3ed501",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    {
      "AvailabilityZone": "us-east-1c",
      "InstanceId": "i-036bc44b6092c01c7",
      "InstanceType": "t2.nano",
      "InstanceMarketOption": "on-demand"
    },
    ...
  ],
  "Cause": "SCALE_IN"
}
```

페이로드에는 Auto Scaling 그룹의 이름, Amazon 리소스 이름 (ARN) 및 다음 요소가 포함됩니다.

- `CapacityToTerminate`는 지정된 가용 영역에서 해지되도록 설정된 스팟 또는 온디맨드 용량의 양을 지정합니다.
- `Instances`는 Amazon EC2 Auto Scaling이 의 정보를 기반으로 종료를 제안하는 인스턴스를 나타냅니다`CapacityToTerminate`.
- `Cause`는 해지를 야기한 이벤트를 지정합니다(`SCALE_IN`, `INSTANCE_REFRESH`, `MAX_INSTANCE_LIFETIME` 또는 `REBALANCE`).

다음 정보는 Amazon EC2 Auto Scaling에서 입력 데이터를 생성하는 방식에서 가장 중요한 요소를 간략하게 설명합니다. `Instances`

- 규모 조정 이벤트 및 인스턴스 새로 고침 기반 종료로 인해 인스턴스가 종료되는 경우 가용 영역 간의 균형을 유지하는 것이 우선합니다. 가용 영역 하나의 인스턴스 수가 그룹에서 사용하는 그 외 가용 영역보다 많으면 입력 데이터에는 균형이 맞지 않는 가용 영역의 해지 대상 인스턴스만 포함됩니다. 그룹에서 사용하는 가용 영역이 균형 잡힌 경우, 입력 데이터에는 해당 그룹에 대한 모든 가용 영역의 인스턴스가 포함됩니다.
- 혼합 인스턴스 정책을 사용하는 경우, 각 구매 옵션에 대해 원하는 백분율을 기준으로 스팟 및 온디맨드 용량을 균형 있게 유지하는 것이 우선합니다. 먼저 둘(스팟 또는 온디맨드) 중 어떤 유형이 해지되어야 하는지 확인합니다. 그런 다음 가용 영역의 균형을 가장 잘 유지하기 위해 어떤 가용 영역 내에서 (식별된 구매 옵션 내) 어떤 인스턴스를 해지할지 파악합니다.

응답 데이터

입력 데이터와 응답 데이터는 함께 작동하여 해지할 인스턴스 목록의 범위를 좁힙니다.

주어진 입력을 사용하면 Lambda 함수의 응답은 다음 예와 같아야 합니다.

```
{
  "InstanceIDs": [
    "i-02e1c69383a3ed501",
    "i-036bc44b6092c01c7",
    ...
  ]
}
```

이 응답에서 `InstanceIDs`는 해지할 준비가 된 인스턴스를 나타냅니다.

또는 해지 준비가 된 다른 인스턴스 세트를 반환하여 입력 데이터의 인스턴스를 재정의할 수 있습니다. Lambda 함수가 호출될 때 해지할 준비가 된 인스턴스가 없으면 인스턴스를 반환하지 않도록 선택할 수도 있습니다.

해지할 준비가 된 인스턴스가 없는 경우, Lambda 함수의 응답은 다음 예와 같아야 합니다.

```
{
  "InstanceIDs": [ ]
}
```

고려 사항

사용자 지정 해지 정책을 사용하는 경우, 다음 고려 사항에 유의하세요.

- 응답 데이터에서 인스턴스를 먼저 반환한다고 해서 해당 인스턴스가 해지되는 것은 아닙니다. Lambda 함수를 호출할 때 필요한 수보다 많은 인스턴스가 반환되는 경우 Amazon Auto EC2 Scaling은 사용자가 Auto Scaling 그룹에 지정한 다른 종료 정책과 비교하여 각 인스턴스를 평가합니다. 여러 해지 정책이 있는 경우에는 목록에 있는 그다음 해지 정책을 적용하려고 하고 해지해야 하는 것보다 인스턴스 수가 더 많으면 다음 해지 정책으로 이동합니다. 지정된 다른 해지 정책이 없는 경우, 기본 해지 정책을 사용하여 해지할 인스턴스를 결정합니다.
- 반환된 인스턴스가 없거나 Lambda 함수 제한 시간이 초과되면 Amazon Auto EC2 Scaling은 함수를 다시 호출하기 전에 잠시 기다립니다. 모든 규모 조정 이벤트의 경우 그룹의 원하는 용량이 현재 용량보다 작으면 계속 시도됩니다. 인스턴스 새로 고침 기반 해지의 경우, 한 시간 동안 계속 시도합니다. 이후 계속해서 인스턴스를 해지하지 못하면 인스턴스 새로 고침 작업에 실패합니다. 최대 인스턴스 수명을 사용할 경우 Amazon EC2 Auto Scaling은 최대 수명을 초과한 것으로 식별된 인스턴스를 계속 종료하려고 시도합니다.
- 함수가 반복적으로 다시 시도되므로 Lambda 함수를 사용자 지정 해지 정책으로 사용하기 전에 코드에서 영구적인 오류가 있는지 테스트하고 수정해야 합니다.
- 종료할 자체 인스턴스 목록으로 입력 데이터를 재정의하고 이러한 인스턴스를 종료하여 가용 영역의 균형이 맞지 않는 경우, Amazon EC2 Auto Scaling은 가용 영역 전체의 용량 분배를 점진적으로 재조정합니다. 먼저 Lambda 함수를 호출하여 해지 준비가 된 인스턴스가 있는지 확인해 재조정 여부를 결정할 수 있습니다. 해지할 준비가 된 인스턴스가 있으면 먼저 새 인스턴스를 시작합니다. 인스턴스 시작이 완료되면 그룹의 현재 용량이 원하는 용량보다 많음을 감지하고 이벤트 축소를 시작합니다.
- 사용자 지정 종료 정책은 확장 보호를 사용하여 특정 인스턴스가 종료되지 않도록 보호하는 기능에도 영향을 미치지 않습니다. 자세한 내용은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 단원을 참조하십시오.

Lambda 함수 생성

먼저, Auto Scaling 그룹의 종료 정책에 Amazon 리소스 이름 ARN () 을 지정할 수 있도록 Lambda 함수를 생성합니다.

Lambda 함수를 생성하려면(콘솔)

1. Lambda 콘솔에서 함수 페이지를 엽니다.
2. 화면 상단의 탐색 모음에서 Auto Scaling 그룹을 생성할 때 사용한 것과 동일한 지역을 선택합니다.
3. Create function(함수 생성)과 Author from scratch(새로 작성)를 차례로 선택합니다.
4. Basic information(기본 정보)에서 Function name(함수 이름)에 함수 이름을 입력합니다.
5. Create function(함수 생성)을 선택합니다. 함수의 코드 및 구성으로 돌아갑니다.
6. 함수가 콘솔에 여전히 열려 있으면 Function code(함수 코드)에서 코드를 편집기에 붙여 넣습니다.
7. Deploy(배치)를 선택합니다.
8. 경우에 따라 Versions(버전) 탭을 선택한 다음 Publish new version(새 버전 발행)을 선택하여 Lambda 함수의 게시 버전을 생성합니다. Lambda의 버전 관리에 대한 자세한 설명은 AWS Lambda 개발자 안내서의 Lambda 함수 버전을 참조하세요.
9. 버전을 게시하도록 선택한 경우, Lambda 함수의 이 버전과 별칭을 연결하려면 Aliases(별칭) 탭을 선택합니다. Lambda의 별칭에 대한 자세한 설명은 AWS Lambda 개발자 안내서의 Lambda 함수 별칭을 참조하세요.
10. 그런 다음 Configuration(구성) 탭을 선택한 다음 Permissions(권한)를 선택합니다.
11. Resource-based policy(리소스 기반 정책)까지 스크롤한 다음 Add permissions(권한 추가)를 선택합니다. 리소스 기반 정책은 정책에 지정된 보안 주체에 함수를 호출하는 권한을 부여하는 데 사용됩니다. 이 경우 기본 역할은 EC2Auto Scaling 그룹과 연결된 Amazon Auto Scaling 서비스 연결 역할이 됩니다.
12. Policy statement(정책 설명) 섹션에서 권한을 구성합니다.

    a. AWS 계정를 선택합니다.

    b. [Principal] 에는 통화 서비스 연결 역할 (예:) ARN 의 이름을 입력합니다. **`arn:aws:iam::<aws-account-id>:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling`**

    c. [액션] 에서 lambda:를 선택합니다. InvokeFunction

    d. Statement ID(설명 ID)에 고유한 설명 ID를 입력합니다(예: **`AllowInvokeByAutoScaling`**).

    e. 저장(Save)을 선택합니다.

13. 이 지침을 따랐으면 다음 단계로 Auto Scaling 그룹의 종료 정책에 함수를 계속 지정하십시오. ARN 자세한 내용은 Auto Scaling 그룹의 종료 정책 변경 단원을 참조하십시오.

> **Note**
>
> Lambda 함수 개발을 위한 참조로 사용할 수 있는 예제는 Amazon Auto EC2 Scaling용 GitHub 리포지토리를 참조하십시오.

제한 사항

- Auto Scaling 그룹에 대한 해지 정책에는 Lambda 함수를 하나만 지정할 수 있습니다. 해지 정책이 여러 개 지정된 경우, Lambda 함수를 먼저 지정해야 합니다.
- ARN정규화되지 않은 (접미사 없음) 또는 버전이나 별칭을 접미사로 포함하는 한정된 함수를 사용하여 Lambda 함수를 참조할 수 있습니다. ARN 비정규 함수를 사용하는 경우 (예:`function:my-function`), 게시되지 않은 ARN 버전의 함수에서 리소스 기반 정책을 생성해야 합니다. 검증된 ARN 함수 (예: `function:my-function:1` 또는`function:my-function:prod`) 를 사용하는 경우 해당 게시된 함수 버전에 대해 리소스 기반 정책을 생성해야 합니다.
- 접미사가 ARN 붙은 한정된 버전은 사용할 수 없습니다. `$LATEST` `$LATEST`접미사가 있는 자격을 ARN 갖춘 사람을 참조하는 사용자 지정 종료 정책을 추가하려고 하면 오류가 발생합니다.
- 입력 데이터에 제공되는 인스턴스 수는 30,000개로 제한됩니다. 해지할 수 있는 인스턴스가 30,000개 이상인 경우, 입력 데이터에는 최대 인스턴스 수가 반환됨을 나타내는 `"HasMoreInstances": true`가 포함됩니다.
- Lambda 함수의 최대 실행 시간은 2초(2,000밀리초)입니다. 예상 실행 시간에 근거하여 Lambda 함수의 제한 시간 값을 설정하는 것이 가장 좋습니다. Lambda 함수의 기본 제한 시간은 3초이지만 이 시간은 줄일 수 있습니다.
- 런타임이 2초 제한을 초과하는 경우 런타임이 이 임계값 아래로 떨어질 때까지 모든 스케일 인 액션이 보류됩니다. 런타임이 지속적으로 더 긴 Lambda 함수의 경우, 후속 Lambda 호출 중에 결과를 검색할 수 있는 결과를 캐싱하는 등 런타임을 줄이는 방법을 찾으십시오.

인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다.

인스턴스 확장 보호를 통해 Amazon EC2 Auto Scaling이 종료할 수 있는 인스턴스를 제어할 수 있습니다. 이 기능의 일반적인 사용 사례는 컨테이너 기반 워크로드를 확장하는 것입니다. 자세한 내용은 인스턴스 종료를 정상적으로 처리하도록 애플리케이션을 설계하십시오. 단원을 참조하십시오.

Auto Scaling 그룹을 생성하면 기본적으로 인스턴스 확장 보호가 비활성화됩니다. 즉, Amazon EC2 Auto Scaling은 그룹 내 모든 인스턴스를 종료할 수 있습니다.

Auto Scaling 그룹에서 인스턴스 스케일 인 보호 설정을 활성화하여 인스턴스가 시작되는 즉시 보호할 수 있습니다. 인스턴스 상태가 `InService`이면 인스턴스 스케일 인 방비가 시작됩니다. 그런 다음 해지할 수 있는 인스턴스를 제어하려면 Auto Scaling 그룹 내의 개별 인스턴스에서 스케일 인 보호 설정을 비활성화합니다. 이렇게 하면 특정 인스턴스를 원치 않는 해지로부터 계속 보호할 수 있습니다.

주제

- 고려 사항
- Auto Scaling 그룹의 스케일 인 보호 변경
- 인스턴스의 스케일 인 보호 변경

고려 사항

다음은 인스턴스 스케일 인 보호를 사용할 때의 고려 사항입니다.

- Auto Scaling 그룹의 모든 인스턴스가 스케일 인으로부터 보호되고 스케일 인 이벤트가 발생하면 원하는 용량이 감소합니다. 하지만 Auto Scaling 그룹은 인스턴스 확장 보호 설정이 비활성화될 때까지 필요한 수의 인스턴스를 종료할 수 없습니다. 에서 Auto Scaling 그룹의 활동 기록에는 스케일 인 이벤트 발생 시 Auto Scaling 그룹의 모든 인스턴스가 스케일 인으로부터 보호되는 경우 다음 메시지가 포함됩니다. AWS Management Console`Could not scale to desired capacity because all remaining instances are protected from scale in.`
- 축소로부터 보호되는 인스턴스를 분리하면 해당 인스턴스 확장 보호 설정이 손실됩니다. 인스턴스를 그룹에 다시 연결하면 그룹의 현재 인스턴스 확장 보호 설정이 상속됩니다. Amazon EC2 Auto Scaling에서 새 인스턴스를 시작하거나 웜 풀의 인스턴스를 Auto Scaling 그룹으로 이동하면 해당 인스턴스는 Auto Scaling 그룹의 보호 범위 조정 설정을 상속합니다.
- 인스턴스 스케일 인 방비는 다음 경우에 Auto Scaling 인스턴스를 보호하지 않습니다.
  - 인스턴스가 건전성 체크를 통과하지 못한 경우, 건전성 체크 교체. 자세한 설명은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 섹션을 참조하세요.
  - 스팟 인스턴스 중단 스팟 인스턴스는 용량을 더 이상 사용할 수 없거나 스팟 가격이 최고가를 초과하는 경우, 해지됩니다.
  - 용량 블록 예약이 종료됩니다. Amazon은 용량 블록 인스턴스가 확장되지 않도록 보호되는 경우에도 해당 인스턴스를 EC2 회수합니다.
  - 명령을 통한 수동 종료. `terminate-instance-in-auto-scaling-group` 자세한 내용은 Auto Scaling 그룹에서 인스턴스 해지 (AWS CLI) 단원을 참조하십시오.

- Amazon EC2 콘솔, CLI 명령 및 API 작업을 통한 수동 종료 Auto Scaling 인스턴스를 수동 종료로부터 보호하려면 Amazon EC2 종료 보호를 활성화하십시오. (그렇다고 해서 Amazon EC2 Auto Scaling에서 인스턴스를 종료하거나 `terminate-instance-in-auto-scaling-group` 명령을 통한 수동 종료가 중단되는 것은 아닙니다.) 시작 템플릿에서 Amazon EC2 종료 보호를 활성화하는 방법에 대한 자세한 내용은 을 참조하십시오고급 설정을 사용하여 시작 템플릿 생성.

Auto Scaling 그룹의 스케일 인 보호 변경

Auto Scaling 그룹의 인스턴스 스케일 인 보호 설정을 활성화하거나 비활성화할 수 있습니다. 활성화하면 그룹에서 시작하는 모든 새 인스턴스에 인스턴스 확장 보호가 활성화됩니다.

Auto Scaling 그룹에 대해 이 설정을 활성화하거나 비활성화해도 기존 인스턴스에는 영향을 주지 않습니다.

Console

새 Auto Scaling 그룹에 대한 스케일 인 보호를 활성화하려면

Auto Scaling 그룹을 생성할 때 그룹 크기 및 조정 정책 구성 페이지의 인스턴스 확장 보호에서 인스턴스 확장 보호 활성화 확인란을 선택합니다.

기존 그룹의 스케일 인 보호를 활성화 또는 비활성화하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 세부 정보(Details) 탭에서 고급 구성(Advanced configurations), 편집(Edit)을 선택합니다.
4. 인스턴스 확장 보호의 경우 필요에 따라 인스턴스 규모 조정 보호 활성화 확인란을 선택하거나 선택 취소하여 이 옵션을 활성화하거나 비활성화합니다.
5. 업데이트를 선택합니다.

AWS CLI

새 Auto Scaling 그룹에 대한 스케일 인 보호를 활성화하려면

다음 create-auto-scaling-group명령을 사용하여 인스턴스 스케일 인 보호를 활성화합니다.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg --new-
instances-protected-from-scale-in ...
```

기존 그룹에 대한 스케일 인 보호를 활성화하려면

다음 update-auto-scaling-group명령을 사용하여 지정된 Auto Scaling 그룹에 대한 인스턴스 확장 보호를 활성화합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --new-
instances-protected-from-scale-in
```

기존 그룹의 스케일 인 보호를 비활성화하려면

다음 명령을 사용하여 지정된 그룹의 인스턴트 스케일 인 보호를 비활성화합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg --no-new-
instances-protected-from-scale-in
```

인스턴스의 스케일 인 보호 변경

기본적으로 인스턴스는 속해 있는 Auto Scaling 그룹의 인스턴스 스케일 인 보호 설정을 가져옵니다. 하지만 시작 후 개별 인스턴스에 대한 인스턴스 확장 보호를 활성화하거나 비활성화할 수 있습니다.

Console

인스턴스에 대한 스케일 인 보호를 활성화 또는 비활성화하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 인스턴스 관리(Instance management) 탭의 인스턴스(Instances)에서 인스턴스를 선택합니다.
4. 인스턴스 스케일 인 보호를 활성화하려면 작업(Actions), 스케일 인 보호 설정(Set scale-in protection)을 선택합니다. 메시지가 표시되면 스케일 인 보호 설정(Set scale-in protection)을 선택합니다.

5. 인스턴스 스케일 인 보호를 비활성화하려면 작업(Actions), 스케일 인 보호 제거(Remove scale-in protection)를 선택합니다. 메시지가 표시되면 스케일 인 보호 제거(Remove scale-in protection)를 선택합니다.

AWS CLI

인스턴스에 대한 스케일 인 보호를 활성화하려면

다음 set-instance-protection명령을 사용하여 지정된 인스턴스에 대한 인스턴스 확장 보호를 활성화합니다.

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --protected-from-scale-in
```

인스턴스에 대한 스케일 인 보호를 비활성화하려면

다음 명령을 사용하여 지정된 인스턴스의 인스턴스 스케일 인 보호를 비활성화합니다.

```
aws autoscaling set-instance-protection --instance-ids i-5f2e8a0d --auto-scaling-group-name my-asg --no-protected-from-scale-in
```

> **Note**
> 인스턴스 확장 보호는 작업자 오류가 발생한 경우 (예: 누군가 Amazon 콘솔 또는 Amazon 콘솔을 사용하여 인스턴스를 수동으로 종료하는 경우) 발생 시 인스턴스가 종료되지 않을 것을 보장하지 않는다는 점을 기억하십시오. EC2 AWS CLI인스턴스가 실수로 종료되지 않도록 보호하기 위해 Amazon EC2 종료 보호를 사용할 수 있습니다. 그러나 해지 방지 및 인스턴스 스케일 인 방비가 활성화된 경우에도 건전성 체크 시 인스턴스가 비정상이라고 판단되거나 그룹 자체가 실수로 삭제된 경우, 인스턴스 스토리지에 저장된 데이터가 손실될 수 있습니다. 다른 환경과 마찬가지로 가장 좋은 방법은 데이터를 자주 백업하거나 비즈니스 연속성 요건에 따라 적절하게 백업하는 것입니다.

인스턴스 종료를 정상적으로 처리하도록 애플리케이션을 설계하십시오.

이 주제에서는 Auto Scaling 그룹이 아직 종료할 준비가 되지 않은 인스턴스를 종료하거나 할당된 작업을 완료하기에 너무 빨리 인스턴스를 종료하는 것을 방지하는 데 사용할 수 있는 기능을 다룹니다. 이

세 가지 기능을 모두 조합하여 사용하거나 별도로 사용하여 인스턴스 종료를 정상적으로 처리하도록 애플리케이션을 설계할 수 있습니다.

예를 들어 장기 실행 작업을 위해 수신 메시지를 수집하는 Amazon SQS 대기열이 있다고 가정해 보겠습니다. 새 메시지가 도착하면 Auto Scaling 그룹의 인스턴스가 메시지를 검색하여 처리를 시작합니다. 각 메시지를 처리하는 데 3시간이 걸립니다. 메시지 수가 증가하면 Auto Scaling 그룹에 새 인스턴스가 자동으로 추가됩니다. 메시지 수가 줄어들면 기존 인스턴스는 자동으로 해지됩니다. 이 경우 Amazon EC2 Auto Scaling에서 종료할 인스턴스를 결정해야 합니다. 기본적으로 Amazon EC2 Auto Scaling은 현재 유휴 상태인 인스턴스가 아니라 3시간의 긴 작업을 처리하는 데 2.9시간이 걸리는 인스턴스를 종료할 수 있습니다. Amazon EC2 Auto Scaling을 사용할 때 예상치 못한 종료 문제가 발생하지 않도록 하려면 이 시나리오에 대응하도록 애플리케이션을 설계해야 합니다.

내용

- 인스턴스 스케일 인 보호
- 사용자 지정 해지 정책
- 해지 수명 주기 후크

> **Important**
>
> Amazon EC2 Auto Scaling에서 인스턴스 종료를 정상적으로 처리하도록 애플리케이션을 설계할 때는 다음 사항을 염두에 두십시오.
>
> - 인스턴스가 비정상인 경우 Amazon EC2 Auto Scaling은 사용자가 사용하는 기능에 관계없이 인스턴스를 교체합니다 (`ReplaceUnhealthy`프로세스를 일시 중단하지 않는 한). 수명 주기 후크를 사용하여 애플리케이션이 정상적으로 해지되도록 하거나 인스턴스가 해지되기 전에 복구해야 하는 모든 데이터를 복사할 수 있습니다.
> - 해지 수명 주기 후크는 인스턴스가 해지되기 전에 실행되거나 완료되는 것이 보장되지 않습니다. 문제가 발생하더라도 Amazon EC2 Auto Scaling은 여전히 인스턴스를 종료합니다.

인스턴스 스케일 인 보호

인스턴스 해지가 기본적으로 거부되어야 하고 특정 인스턴스에 대해서만 명시적으로 허용되어야 하는 중요한 작업인 많은 상황에서 인스턴스 스케일 인 보호 기능을 사용할 수 있습니다. 예컨대, 컨테이너화된 워크로드를 실행할 때 모든 인스턴스를 보호하고 현재 또는 예약된 작업이 없는 인스턴스에 대해서만 보호를 해제하는 것이 일반적입니다. ECSAmazon과 같은 서비스에서는 인스턴스 스케일 인 보호 기능이 포함된 통합 기능을 제품에 내장했습니다.

Auto Scaling 그룹에서 스케일인 보호를 활성화하여 인스턴스가 생성될 때 인스턴스에 스케일인 보호를 적용하고 기존 인스턴스에 대해 스케일인 보호를 활성화할 수 있습니다. 인스턴스가 더 이상 수행할 작업이 없는 경우, 보호 기능을 해제할 수 있습니다. 인스턴스는 새 작업에 대한 폴링을 계속하고 새 작업이 할당되면 보호를 다시 활성화할 수 있습니다.

애플리케이션은 인스턴스의 해지 가능 여부를 관리하는 중앙 집중식 제어 영역 또는 인스턴스 자체에서 보호 기능을 설정할 수 있습니다. 그러나 많은 수의 인스턴스가 스케일 인 보호 기능을 계속적으로 토글하는 경우, 제한 문제가 발생할 수 있습니다.

자세한 설명은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 섹션을 참조하세요.

사용자 지정 해지 정책

인스턴스 스케일 인 보호와 마찬가지로 사용자 지정 해지 정책은 Auto Scaling 그룹이 특정 인스턴스를 해지하는 것을 방지하는 데 도움이 됩니다.

기본적으로 Auto Scaling 그룹은 기본 해지 정책을 사용하여 어떤 인스턴스를 먼저 해지할지 결정합니다. 어떤 인스턴스가 먼저 해지되는지 더 세밀하게 제어하고 싶은 경우, Lambda 함수를 사용하여 사용자 지정 해지 정책을 구현할 수 있습니다. Amazon EC2 Auto Scaling은 종료할 인스턴스를 결정해야 할 때마다 함수를 호출합니다. 함수가 반환하는 인스턴스만 해지합니다. 함수에서 오류가 발생하거나, 시간 초과가 발생하거나, 빈 목록이 생성되는 경우 Amazon EC2 Auto Scaling은 인스턴스를 종료하지 않습니다.

사용자 지정 해지 정책은 인스턴스가 과도하게 중복되거나 활용도가 낮은 시점을 파악하여 해지할 수 있는 경우에 유용합니다. 이를 지원하려면 그룹 전체의 워크로드를 모니터링하는 컨트롤 플레인을 사용하여 애플리케이션을 구현해야 합니다. 이렇게 하면 인스턴스가 여전히 작업을 처리 중인 경우, Lambda 함수가 해당 인스턴스를 포함하지 않도록 인식합니다.

자세한 설명은 Lambda를 사용하여 사용자 지정 해지 정책 생성 섹션을 참조하세요.

해지 수명 주기 후크

해지 수명 주기 후크는 이미 해지하도록 선택된 인스턴스의 수명을 연장합니다. 현재 인스턴스에 할당된 모든 메시지 또는 요청을 완료하거나 진행 상황을 저장하고 작업을 다른 인스턴스로 전송할 수 있는 추가 시간을 제공합니다.

많은 워크로드의 경우, 해지를 위해 선택된 인스턴스에서 수명 주기 후크를 통해 애플리케이션을 정상적으로 해지하는 것으로 충분할 수 있습니다. 이는 최선의 방법이며 장애가 발생한 경우, 해지를 방지하는 데 사용할 수 없습니다.

수명 주기 후크를 사용하려면 인스턴스가 해지되도록 선택된 시점을 알아야 합니다. 이를 알 수 있는 방법은 다음 두 가지입니다:

옵션	설명	가장 적합한 용도	설명서 링크
인스턴스 내부	인스턴스 메타데이터 서비스 (IMDS) 는 인스턴스에서 직접 인스턴스 상태를 폴링할 수 있는 안전한 엔드포인트입니다. 메타데이터가 `Terminated` (으)로 돌아오면 인스턴스가 해지되도록 예약된 것입니다.	인스턴스가 해지되기 전에 인스턴스에서 작업을 수행해야 하는 애플리케이션.	대상 수명 주기 상태 검색
인스턴스 외부	인스턴스가 해지되면 이벤트 알림이 생성됩니다. Amazon EventBridgeSQS, Amazon 또는 Amazon을 사용하여 규칙을 SNS 생성하여 이러한 이벤트를 캡처하고 Lambda 함수 등을 사용하여 응답을 호출할 수 있습니다.	인스턴스 외부에서 작업을 수행해야 하는 애플리케이션.	알림 대상 구성

수명 주기 후크를 사용하려면 또한 해당 인스턴스가 언제 완전히 해지될 준비가 되는지를 알아야 합니다. Amazon EC2 Auto Scaling은 CompleteLifecycleAction호출이 오거나 제한 시간이 경과한 시점 중 먼저 발생하는 경우를 제외하고 EC2 Amazon에 인스턴스를 종료하라고 지시하지 않습니다.

기본적으로 인스턴스는 해지 수명 주기 후크로 인해 1시간 동안 계속 실행될 수 있습니다(하트비트 타임아웃). 1시간이 수명 주기 작업을 완료하는 데 충분하지 않은 경우, 기본 시간 제한을 구성할 수 있습니다. 수명 주기 작업이 실제로 진행 중이면 호출을 통해 제한 시간을 연장할 수 있습니다. RecordLifecycleActionHeartbeatAPI

자세한 내용은 Amazon EC2 Auto Scaling 수명 주기 후크 단원을 참조하십시오.

Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개

이 항목에서는 Auto Scaling 그룹의 하나 이상의 프로세스를 일시 중단했다가 재개하여 특정 작업을 일시적으로 비활성화하는 방법에 대해 설명합니다.

프로세스를 일시 중단하는 것은 조정 정책이나 예정된 조치의 방해 없이 문제를 조사하거나 해결해야 할 때 유용할 수 있습니다. 또한 EC2 Auto Scaling 그룹을 변경하는 동안 Amazon Auto Scaling에서 인스턴스를 비정상으로 표시하고 교체하는 것을 방지할 수 있습니다.

주제

- 프로세스 유형
- 프로세스 일시 중단에 대한 고려 사항
- 프로세스 일시 중지
- 프로세스 재개
- 일시 중단된 프로세스가 다른 프로세스에 미치는 영향

> **Note**
>
> Amazon EC2 Auto Scaling은 사용자가 시작한 일시 중단 외에도 인스턴스 시작에 반복적으로 실패하는 Auto Scaling 그룹의 프로세스를 일시 중단할 수 있습니다. 이를 관리적 일시 중지라고도 합니다. 관리적 일시 중지는 24시간 이상 인스턴스를 시작하려고 했지만 시작하지 못한 Auto Scaling 그룹에 가장 흔히 적용됩니다. 관리상의 이유로 Amazon EC2 Auto Scaling에서 일시 중단한 프로세스를 재개할 수 있습니다.

프로세스 유형

일시 중지 후 재개 기능은 다음 프로세스를 지원합니다.

- `Launch`— 그룹이 확장되거나 Amazon EC2 Auto Scaling에서 다른 이유 (예: 웜 풀에 인스턴스를 추가하는 경우) 로 인스턴스를 시작하기로 선택한 경우 Auto Scaling 그룹에 인스턴스를 추가합니다.
- `Terminate`— 그룹이 축소되거나 Amazon EC2 Auto Scaling에서 다른 이유 (예: 인스턴스가 최대 수명 기간을 초과하여 종료되거나 상태 확인에 실패한 경우) 로 인스턴스를 종료하기로 선택한 경우 Auto Scaling 그룹에서 인스턴스를 제거합니다.
- `AddToLoadBalancer`— 연결된 로드 밸런서 대상 그룹 또는 Classic Load Balancer가 시작되면 인스턴스를 추가합니다. 자세한 내용은 Elastic Load Balancing을 사용하여 Auto Scaling 그룹에서 들어오는 애플리케이션 트래픽을 분산하십시오. 단원을 참조하십시오.
- `AlarmNotification`— 동적 조정 정책과 관련된 CloudWatch 경보의 알림을 수락합니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 동적 스케일링 단원을 참조하십시오.

- `AZRebalance`— 그룹이 불균형해질 때 (예: 이전에 사용할 수 없었던 가용 영역이 정상 상태로 돌아오는 경우) 그룹의 EC2 인스턴스 수를 지정된 모든 가용 영역에서 균등하게 조정합니다. 자세한 내용은 재조정 활동 단원을 참조하십시오.
- `HealthCheck`— Amazon EC2 또는 Elastic Load Balancing에서 Amazon EC2 Auto Scaling에 해당 인스턴스가 비정상이라고 통보하면 인스턴스의 상태를 확인하고 인스턴스를 비정상으로 표시합니다. 이 프로세스로 사용자가 수동으로 설정한 인스턴스의 상태를 재정의할 수 있습니다. 자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 단원을 참조하십시오.
- `InstanceRefresh`— 인스턴스 새로 고침 기능을 사용하여 인스턴스를 종료하고 교체합니다. 자세한 내용은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다. 단원을 참조하십시오.
- `ReplaceUnhealthy`— 비정상으로 표시된 인스턴스를 종료한 다음 이를 대체할 새 인스턴스를 생성합니다. 자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 단원을 참조하십시오.
- `ScheduledActions`— 조정 계획을 생성하고 예측 조정을 활성화할 때 사용자가 생성하거나 사용자를 위해 생성된 AWS Auto Scaling 스케줄링된 조정 작업을 수행합니다. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 스케일링 예약 단원을 참조하십시오.

프로세스 일시 중단에 대한 고려 사항

프로세스를 일시 중지하기 전에 다음 사항을 고려하세요.

- 일시 `AlarmNotification` 중단하면 조정 정책 또는 관련 CloudWatch 경보를 삭제하지 않고도 그룹의 대상 추적, 단계 및 단순 조정 정책을 일시적으로 중지할 수 있습니다. 대신 개별 조정 정책을 일시적으로 중지하려면 Auto Scaling 그룹에 대한 조정 정책 비활성화 섹션을 참조하세요.
- Amazon EC2 Auto Scaling에서 상태 확인에 따라 인스턴스를 종료하지 않고 `HealthCheck` 및 `ReplaceUnhealthy` 프로세스를 일시 중단하여 인스턴스를 재부팅하도록 선택할 수 있습니다. 하지만 Amazon EC2 Auto Scaling에서 나머지 인스턴스의 상태 확인을 계속 수행해야 하는 경우에는 대신 대기 기능을 사용하십시오. 자세한 내용은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 단원을 참조하십시오.
- `Launch` 및 `Terminate` 프로세스나 `AZRebalance`를 일시 중지한 다음 인스턴스 분리, 지정된 가용 영역 변경 등의 Auto Scaling 그룹 변경 작업을 수행하면 가용 영역 간에 그룹의 균형이 깨질 수 있습니다. 이 경우 일시 중단된 프로세스를 재개한 후 Amazon EC2 Auto Scaling에서 점진적으로 인스턴스를 가용 영역 간에 균등하게 재분배합니다.
- `Terminate`프로세스를 일시 중단한 경우에도 강제 삭제 옵션과 함께 delete-auto-scaling-group명령을 사용하여 인스턴스를 강제로 종료할 수 있습니다.

- `Terminate`프로세스 일시 중지는 현재 상태에 있는 인스턴스에만 적용됩니다. `InService` 이렇게 해도 다른 상태의 인스턴스나 대기 상태에서 제대로 재개되지 못한 인스턴스가 종료되는 것을 막지는 못합니다. `Pending`
- AWS CLI 또는 를 사용하여 Auto Scaling 그룹을 설명하는 호출에 `RemoveFromLoadBalancerLowPriority` 프로세스가 있는 경우 프로세스를 무시할 수 SDKs 있습니다. 이 프로세스는 더 이상 사용되지 않으며 이전 버전과의 호환성을 위해서만 유지됩니다.

프로세스 일시 중지

Auto Scaling 그룹의 프로세스를 일시 중단하려면 다음 방법 중 하나를 사용하십시오.

Console

프로세스를 일시 중지하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 세부 정보(Details) 탭에서 고급 구성(Advanced configurations), 편집(Edit)을 선택합니다.
4. Suspended processes(일시 중지된 프로세스)에서 일시 중지할 프로세스를 선택합니다.
5. 업데이트를 선택합니다.

AWS CLI

다음 suspend-processes 명령을 사용하여 개별 프로세스를 일시 중지합니다.

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg --scaling-
processes HealthCheck ReplaceUnhealthy
```

프로세스를 모두 일시 중지하려면 다음과 같이 `--scaling-processes` 옵션을 생략합니다.

```
aws autoscaling suspend-processes --auto-scaling-group-name my-asg
```

프로세스 재개

Auto Scaling 그룹에 대해 일시 중단된 프로세스를 재개하려면 다음 방법 중 하나를 사용하십시오.

Console

일시 중지된 프로세스를 재개하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 세부 정보(Details) 탭에서 고급 구성(Advanced configurations), 편집(Edit)을 선택합니다.
4. 일시 중지된 프로세스(Suspended processes)에서 일시 중지된 프로세스를 제거합니다.
5. 업데이트를 선택합니다.

AWS CLI

일시 중단된 프로세스를 재개하려면 다음 resume-processes 명령을 사용하십시오.

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg --scaling-
processes HealthCheck
```

일시 중지된 프로세스를 모두 재개하려면 다음과 같이 `--scaling-processes` 옵션을 생략합니다.

```
aws autoscaling resume-processes --auto-scaling-group-name my-asg
```

일시 중단된 프로세스가 다른 프로세스에 미치는 영향

다음 섹션에서는 여러 프로세스가 개별적으로 일시 중단될 때 발생하는 상황에 대해 설명합니다.

주제

- Launch일시 중단되었습니다.
- Terminate일시 중단되었습니다.

- AddToLoadBalancer일시 중지되었습니다.
- AlarmNotification일시 중단되었습니다.
- AZRebalance일시 중지되었습니다.
- HealthCheck일시 중지됩니다.
- InstanceRefresh일시 중지되었습니다.
- ReplaceUnhealthy일시 중단되었습니다.
- ScheduledActions일시 중단되었습니다.
- 추가 고려 사항

Launch일시 중단되었습니다.

- AlarmNotification이 여전히 활성 상태이지만 Auto Scaling 그룹이 위반 상태의 경보에 대해 스케일 아웃 활동을 시작할 수 없습니다.
- ScheduledActions가 활성 상태이지만 Auto Scaling 그룹이 수행되는 예약된 작업에 대해 스케일 아웃 활동을 시작할 수 없습니다.
- AZRebalance가 그룹 재조정을 중지합니다.
- ReplaceUnhealthy가 비건전 인스턴스를 계속 해지하지만 교체를 시작하지는 않습니다. Launch프로세스를 재개하면 Amazon EC2 Auto Scaling은 일시 중단된 기간 동안 종료된 모든 인스턴스를 즉시 교체합니다. Launch
- InstanceRefresh가 인스턴스를 교체하지 않습니다.

Terminate일시 중단되었습니다.

- AlarmNotification은 (는) 여전히 활성화되어 있지만 Auto Scaling 그룹은 위반이 발생한 경보에 대한 활동 확장을 시작할 수 없습니다.
- ScheduledActions활성 상태이지만 Auto Scaling 그룹은 발생하는 예정된 작업에 대해 활동 확장을 시작할 수 없습니다.
- AZRebalance는 여전히 활성이지만 정상적으로 작동하지 않습니다. 오래된 인스턴스를 해지하지 않고도 새 인스턴스를 시작할 수 있습니다. 이로 인해 Auto Scaling 그룹은 최대 크기보다 최대 10% 더 크게 스케일 아웃될 수 있습니다. 재조정 활동 중에 일시적으로 허용되기 때문입니다. 그리고 Auto Scaling 그룹은 Terminate 프로세스를 재개할 때까지 최대 크기 이상으로 유지될 수 있습니다.

- `ReplaceUnhealthy`는 비활성 상태이지만 `HealthCheck`는 아닙니다. `Terminate`가 재개되면 `ReplaceUnhealthy` 프로세스는 즉시 실행되기 시작합니다. `Terminate`가 일시 중지된 동안 건전하지 않은 것으로 표시되었던 인스턴스는 즉시 교체됩니다.
- `InstanceRefresh`가 인스턴스를 교체하지 않습니다.

`AddToLoadBalancer`일시 중지되었습니다.

- Amazon EC2 Auto Scaling은 인스턴스를 시작하지만 로드 밸런서 대상 그룹이나 Classic Load Balancer에 추가하지는 않습니다. `AddToLoadBalancer` 프로세스를 재개하면 이 프로세스는 인스턴스 시작 시 로드 밸런서에 해당 인스턴스를 다시 추가하기 시작합니다. 하지만 이 프로세스가 일시 중지되면 인스턴스가 시작되어도 인스턴스가 추가되지 않습니다. 따라서 이러한 인스턴스를 수동으로 등록해야 합니다.

`AlarmNotification`일시 중단되었습니다.

- Amazon EC2 Auto Scaling은 CloudWatch 경보 임계값이 위반되는 경우 조정 정책을 호출하지 않습니다. 작업을 `AlarmNotification` 재개하면 Amazon EC2 Auto Scaling은 현재 위반 상태인 경보 임계값이 있는 정책을 고려합니다.

`AZRebalance`일시 중지되었습니다.

- Amazon EC2 Auto Scaling은 특정 이벤트 이후에 인스턴스를 재배포하려고 시도하지 않습니다. 하지만 스케일 아웃 또는 스케일 인 이벤트가 발생하는 경우에도 규모 조정 프로세스는 여전히 가용 영역의 균형을 맞추려고 합니다. 예컨대, 스케일 아웃 시에는 최소한의 인스턴스로 가용 영역의 인스턴스를 시작합니다. 일시 중단된 상태에서 `AZRebalance` 그룹이 불균형 상태가 되었다가 그룹을 재개하면 Amazon EC2 Auto Scaling에서 그룹 재조정을 시도합니다. 이를 위해 먼저 `Launch`를 호출한 다음 `Terminate`를 호출합니다.

`HealthCheck`일시 중지됩니다.

- Amazon EC2 Auto Scaling은 Elastic Load Balancing 상태 확인의 EC2 결과로 인스턴스를 비정상으로 표시하는 것을 중단합니다. 사용자 지정 건전성 체크은 계속 정상적으로 이루어집니다. `HealthCheck`를 일시 중지한 뒤 필요하면 그룹 내 인스턴스의 상태 검사를 수동으로 설정하고 `ReplaceUnhealthy`에서 이를 교체하도록 할 수 있습니다.

`InstanceRefresh`일시 중지되었습니다.

- Amazon EC2 Auto Scaling은 인스턴스 새로 고침의 결과로 인스턴스 교체를 중단합니다. 진행 중인 인스턴스 새로 고침이 있는 경우, 작업을 취소하지 않고 일시 중지합니다.

`ReplaceUnhealthy`일시 중단되었습니다.

- Amazon EC2 Auto Scaling은 비정상으로 표시된 인스턴스의 교체를 중단합니다. 실패한 인스턴스 EC2 또는 Elastic Load Balancing 상태 확인은 여전히 비정상으로 표시됩니다. 프로세스를 재개하는 즉시 Amazon EC2 Auto Scaling은 이 `ReplaceUnhealthy` 프로세스가 일시 중단된 동안 비정상으로 표시된 인스턴스를 대체합니다. `ReplaceUnhealthy` 프로세스는 먼저 `Terminate`를 호출한 다음 `Launch`를 호출합니다.

`ScheduledActions`일시 중단되었습니다.

- Amazon EC2 Auto Scaling은 일시 중지 기간 중에 실행되도록 예약된 작업을 실행하지 않습니다. 작업을 `ScheduledActions` 재개하면 Amazon EC2 Auto Scaling은 예약된 시간이 아직 지나지 않은 예약된 작업만 고려합니다.

추가 고려 사항

또한 `Launch` 또는 `Terminate`가 일시 중지되면 다음 기능이 제대로 작동하지 않을 수 있습니다.

- 최대 인스턴스 수명 — `Launch` 또는 `Terminate` 일시 중단된 경우 최대 인스턴스 수명 기능으로 인스턴스를 대체할 수 없습니다.
- 스팟 인스턴스 중단 - 일시 `Terminate` 중단되고 Auto Scaling 그룹에 스팟 인스턴스가 있는 경우에도 스팟 용량을 더 이상 사용할 수 없는 경우 여전히 종료될 수 있습니다. 일시 중단된 `Launch` 동안에는 Amazon EC2 Auto Scaling이 다른 스팟 인스턴스 풀이나 동일한 스팟 인스턴스 풀에서 대체 인스턴스를 다시 사용할 수 있게 되면 이를 시작할 수 없습니다.
- 용량 재조정 — 일시 중단된 상태에서 용량 재조정을 사용하여 스팟 인스턴스 중단을 처리하는 경우에도 `Terminate` Amazon EC2 Spot 서비스는 스팟 용량을 더 이상 사용할 수 없는 경우에도 인스턴스를 종료할 수 있습니다. 일시 중단된 경우 Amazon EC2 Auto `Launch` Scaling은 대체 인스턴스를 다시 사용할 수 있게 되었을 때 다른 스팟 인스턴스 풀이나 동일한 스팟 인스턴스 풀에서 대체 인스턴스를 시작할 수 없습니다.
- 인스턴스 연결 및 분리 - `Launch` 및 `Terminate` 일시 중단된 경우 Auto Scaling 그룹에 연결된 인스턴스를 분리할 수 있지만 일시 중단된 `Launch` 동안에는 그룹에 새 인스턴스를 연결할 수 없습니다.

- 대기 인스턴스 — 일시 중단된 경우 인스턴스를 Standby 상태로 전환할 수 있지만 일시 중지된 Launch 동안에는 해당 상태의 인스턴스를 서비스 Standby 상태로 되돌릴 수 없습니다. Launch Terminate

Amazon EC2 Auto Scaling 그룹을 모니터링하십시오.

모니터링은 Amazon EC2 Auto Scaling 및 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어 중요한 부분입니다. AWS 클라우드 AWS 는 Amazon EC2 Auto Scaling을 관찰하고, 문제 발생 시 보고하고, 적절한 경우 자동 조치를 취할 수 있는 다음과 같은 모니터링 도구를 제공합니다.

상태 확인

Amazon EC2 Auto Scaling에서는 Auto Scaling 그룹의 인스턴스에 대한 건전성 체크를 주기적으로 수행합니다. 인스턴스가 건전성 체크를 통과하지 못하면 건전하지 않은 것으로 표식되고 그것을 교체하기 위해 Amazon EC2 Auto Scaling이 새 인스턴스를 출범하는 동안 해지됩니다. 자세한 정보는 Auto Scaling 그룹의 인스턴스에 대한 상태 확인을 참조하세요.

AWS Health Dashboard

정보를 AWS Health Dashboard 표시하고 리소스 상태 변경에 따라 호출되는 알림도 제공합니다. AWS 이 정보는 최근 이벤트와 예정된 이벤트를 카테고리별로 보여주는 대시보드와 지난 90일간의 모든 이벤트를 보여주는 전체 이벤트 로그의 두 가지 방법으로 표시됩니다. 자세한 내용은 AWS Health Dashboard Amazon EC2 Auto Scaling에 대한 알림 섹션을 참조하세요.

CloudTrail

를 사용하면 사용자를 대신하여 Amazon EC2 Auto Scaling API에 걸려온 호출을 추적할 수 있습니다. AWS CloudTrail AWS 계정 CloudTrail 지정한 Amazon S3 버킷의 로그 파일에 정보를 저장합니다. 이러한 로그 파일을 사용하여 Auto Scaling 그룹의 활동을 모니터링할 수 있습니다. 로그에는 수행된 요청, 요청의 발신 IP 주소, 요청한 사람, 요청한 시간 등이 기록됩니다. 자세한 설명은 를 사용하여 Amazon EC2 Auto Scaling API 통화 로깅 AWS CloudTrail 섹션을 참조하세요.

> **Amazon EC2 인스턴스에 대한 로그 수집**
>
> 를 CloudWatch 사용하여 운영 체제에서 EC2 인스턴스의 로그를 수집할 수 있습니다. 자세한 내용은 CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스 및 온프레미스 서버에서 지표 및 로그 수집 및 Amazon 사용 설명서의 CloudWatch Logs로 전송된 로그 데이터 보기를 참조하십시오. CloudWatch
> 워크로드에 대한 데이터를 기록하고 수집하는 데 도움이 되는 기타 AWS 서비스에 대한 자세한 내용은 규정 지침의 애플리케이션 소유자를 위한 로깅 및 모니터링 안내서를 참조하십시오.AWS

아마존 CloudWatch

CloudWatch Amazon은 로그를 분석하고 AWS 리소스 및 호스팅된 애플리케이션의 지표를 실시간으로 모니터링할 수 있도록 지원합니다. 지표를 수집 및 추적하고, 맞춤 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임계값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예컨대, 네트워크 활동이 지표의 예상값보다 갑자기 높아지거나 낮아지면 알림을 받을 수 있습니다. 이 서비스를 사용하여 Auto Scaling 그룹 및 인스턴스의 지표를 모니터링하는 방법에 대한 자세한 설명은 Auto Scaling 그룹 및 인스턴스의 CloudWatch 메트릭을 모니터링합니다.을(를) 참조하세요.

CloudWatch 또한 Amazon EC2 Auto Scaling의 AWS API 사용 지표도 추적합니다. 이러한 지표를 사용하여 API 호출량이 정의한 임계값을 위반할 때 경고하는 경보를 구성할 수 있습니다. 자세한 내용은 Amazon AWS 사용 CloudWatch 설명서의 사용량 지표를 참조하십시오.

AWS Compute Optimizer

Compute Optimizer는 새 인스턴스 타입으로 이동할지 여부를 결정하는 데 도움이 될 수 있는 Amazon EC2 인스턴스 권장 사항을 제공합니다. Auto Scaling 그룹의 인스턴스 타입이 최적인지 분석하고 비용을 절감하고 워크로드의 성능을 개선하기 위한 권장 사항을 생성합니다. 자세한 정보는 다음을 통해 인스턴스 유형 권장 사항을 확인하세요. AWS Compute Optimizer을 참조하세요.

아마존 EventBridge

EventBridge Amazon은 다양한 소스의 데이터에 애플리케이션을 쉽게 연결할 수 있게 해주는 서버리스 이벤트 버스 서비스입니다. EventBridge 자체 애플리케이션, SaaS (Software-as-a-Service) 애플리케이션 및 서비스의 실시간 데이터 스트림을 제공하고 해당 데이터를 AWS Lambda와 같은 대상으로 라우팅합니다. 이 방법을 통해 서비스에서 발생하는 이벤트를 모니터링하고 이벤트 기반 아키텍처를 구축할 수 있습니다. 자세한 설명은 Auto Scaling 이벤트를 처리하는 EventBridge 데 사용합니다. 섹션을 참조하세요.

AWS Security Hub

AWS Security Hub을(를) 보안 모범 사례와 관련하여 Amazon EC2 Auto Scaling의 사용량을 모니터링합니다. Security Hub는 탐지 보안 컨트롤을 사용하여 리소스 구성 및 보안 표준을 평가하여 다양한 규정 준수 프레임워크를 준수할 수 있도록 지원합니다. Security Hub를 사용하여 Amazon EC2 Auto Scaling 리소스를 평가하는 방법에 대한 자세한 설명은 AWS Security Hub 사용자 가이드의 Amazon EC2 Auto Scaling 컨트롤을 참조하세요.

Amazon Simple Notification Service

Amazon EC2 Auto Scaling에서 인스턴스를 출범하거나 해지할 때 Amazon SNS 알림을 보내도록 Auto Scaling 그룹을 구성할 수 있습니다. 자세한 설명은 아마존 EC2 오토 스케일링에 대한 아마존 SNS 알림 옵션 섹션을 참조하세요.

Auto Scaling 그룹의 인스턴스에 대한 상태 확인

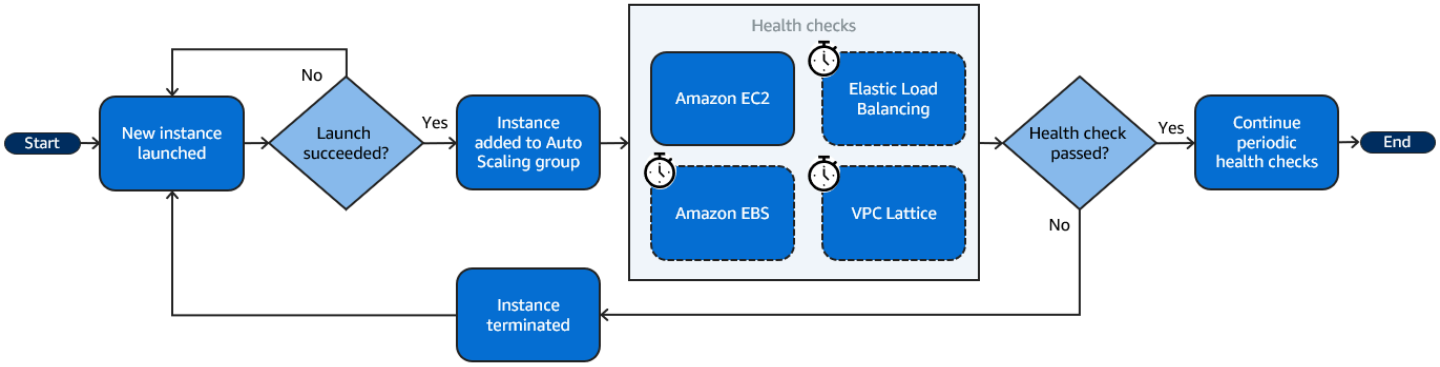
Amazon EC2 Auto Scaling은 Auto Scaling 그룹의 인스턴스 상태를 지속적으로 모니터링하여 원하는 용량을 유지합니다.

Auto Scaling 그룹의 모든 인스턴스는 `Healthy` 상태에서 시작합니다. Amazon EC2 Auto Scaling에서 비정상 상태라는 알림을 받지 않는 한 인스턴스는 정상인 것으로 간주됩니다. 인스턴스가 비정상으로 바뀌어 교체해야 하는 경우 다양한 소스로부터 알림을 받을 수 있습니다. 이 소스에는 다음 사항이 포함됩니다.

- 아마존 EC2
- Elastic Load Balancing
- VPC래티스
- 아마존 EBS
- 사용자가 정의한 사용자 지정 상태 확인

Amazon EC2 Auto Scaling은 `InService` 인스턴스가 비정상이라고 판단하면 이를 새 인스턴스로 교체하여 그룹의 원하는 용량을 유지합니다. 새 인스턴스는 Auto Scaling 그룹의 현재 설정과 연결된 시작 템플릿 또는 시작 구성을 사용하여 시작됩니다.

다음 흐름도는 Auto Scaling 그룹에서 새 인스턴스를 시작하는 프로세스를 보여줍니다. 인스턴스를 시작하는 것으로 시작합니다. 시작에 성공하면 인스턴스가 Auto Scaling 그룹에 추가됩니다. 그러면 Amazon EC2 Auto Scaling이 내장된 Amazon EC2 상태 확인을 사용하여 인스턴스에 대한 상태 확인을 수행하고, 유예 기간이 지나면 그룹에 대해 활성화한 모든 선택적 상태 확인을 수행합니다. 이러한 상태 점검은 주기적으로 계속됩니다. 상태 점검 중 하나라도 실패하면 인스턴스가 교체됩니다.



스팟 인스턴스 중단이나 사용자에 의한 수동 종료와 같이 인스턴스가 예기치 않게 종료되는 경우에도 비정상 인스턴스가 발생할 수 있습니다. 다시 말하지만, Amazon EC2 Auto Scaling은 이러한 경우 자동으로 대체 인스턴스를 시작하여 원하는 용량을 유지합니다.

내용

- Auto Scaling 그룹의 상태 확인 정보
- Auto Scaling 그룹의 상태 확인 유예 기간 설정
- 상태 확인을 사용하여 Amazon EBS 볼륨이 손상된 Auto Scaling 인스턴스를 모니터링합니다.
- Auto Scaling 그룹에 대한 사용자 지정 상태 점검 설정
- 상태 확인 불합격 이유 확인
- Amazon EC2 Auto Scaling에서 비정상 인스턴스 문제 해결

Auto Scaling 그룹의 상태 확인 정보

이 주제에서는 사용 가능한 상태 점검 유형에 대한 개요를 제공하고 Amazon EC2 Auto Scaling 상태 확인을 애플리케이션에 통합하기 위한 주요 고려 사항을 설명합니다.

내용

- 상태 확인 유형
- 아마존 EC2 헬스 체크
- Elastic Load Balancing 상태 확인
- VPC래티스 상태 점검
- Amazon EC2 Auto Scaling을 통해 다운타임을 최소화하는 방법
- 웜 풀에 있는 인스턴스의 건강 점검
- 상태 확인 고려 사항

상태 확인 유형

Amazon EC2 Auto Scaling은 다음 상태 확인 중 하나 이상을 사용하여 `InService` 인스턴스의 상태를 확인할 수 있습니다.

상태 확인 유형	확인 내용
Amazon EC2 상태 확인 및 예정된 이벤트	• 인스턴스가 실행 중인지 확인합니다. • 인스턴스를 손상시킬 수 있는 기본 하드웨어 또는 소프트웨어 문제가 있는지 확인합니다. Auto Scaling 그룹의 기본 상태 확인 유형입니다.
Elastic Load Balancing 상태 확인	• 로드 밸런서가 인스턴스를 정상으로 보고하는지 확인하여 해당 인스턴스가 요청을 처리할 수 있는지 확인합니다. 이 상태 점검 유형을 실행하려면 Auto Scaling 그룹에서 이 상태 점검 유형을 켜야 합니다.
VPC래티스 상태 점검	• VPCLattice가 인스턴스를 정상으로 보고하는지 확인하여 인스턴스가 요청을 처리할 수 있는지 확인합니다. 이 상태 점검 유형을 실행하려면 Auto Scaling 그룹에서 이 상태 점검 유형을 켜야 합니다.
아마존 EBS 헬스 체크	• EBS볼륨에 도달할 수 있는지 확인하고 I/O 상태 검사를 통과합니다. 이 상태 점검 유형을 실행하려면 Auto Scaling 그룹에서 이 상태 점검 유형을 켜야 합니다.
사용자 지정 상태 확인	• 사용자 지정 상태 확인에 따라 인스턴스 상태 문제를 나타낼 수 있는 다른 문제가 있는지 확인합니다.

아마존 EC2 헬스 체크

인스턴스는 시작된 후에 Auto Scaling 그룹에 연결되고 `InService` 상태로 들어갑니다. Auto Scaling 그룹에서 인스턴스의 여러 수명 주기에 대한 자세한 설명은 아마존 EC2 Auto Scaling 인스턴스 라이프 사이클 섹션을 참조하세요.

Amazon EC2 Auto Scaling은 Auto Scaling 그룹 내 모든 인스턴스의 상태를 정기적으로 확인하여 인스턴스가 실행 중이고 양호한 상태인지 확인합니다.

상태 확인

Amazon EC2 Auto Scaling은 Amazon EC2 인스턴스 상태 확인 및 시스템 상태 확인 결과를 사용하여 인스턴스의 상태를 확인합니다. 인스턴스가 Amazon 상태가 아닌 `running` 다른 Amazon EC2 상태이거나 상태 확인 상태가 되면 Amazon EC2 Auto Scaling은 해당 인스턴스를 비정상으로 간주하여 교체합니다. `impaired` 인스턴스가 다음 상태일 때도 위와 같이 실행됩니다.

- `stopping`
- `stopped`
- `shutting-down`
- `terminated`

Amazon EC2 상태 확인에는 특별한 구성이 필요하지 않으며 항상 활성화됩니다. 자세한 내용은 Amazon EC2 사용 설명서의 상태 확인 유형을 참조하십시오.

> **Important**
>
> Amazon EC2 Auto Scaling을 사용하면 어떤 조치도 취하지 않고도 상태 확인이 가끔 실패할 수 있습니다. 상태 확인이 실패하면 Amazon EC2 Auto Scaling은 문제가 해결될 때까지 몇 분 정도 AWS 기다립니다. 상태 확인를 위한 그 상태가 `impaired`가 될 때 그것은 인스턴스를 `Unhealthy`로 즉각 표식하지 않습니다.
> 하지만 Amazon EC2 Auto Scaling에서 인스턴스가 더 이상 `running` 상태가 아님을 감지하면 이 상황은 즉각적인 실패로 간주됩니다. 이 경우 인스턴스를 즉시 로 `Unhealthy` 표시하고 교체합니다.

예약된 이벤트

Amazon은 때때로 특정 타임스탬프 이후에 인스턴스의 이벤트가 실행되도록 예약할 EC2 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 대한 예약 이벤트를 참조하십시오.

인스턴스 중 하나가 예정된 이벤트의 영향을 받는 경우 Amazon EC2 Auto Scaling은 해당 인스턴스를 비정상으로 간주하여 교체합니다. 타임스탬프에 지정된 날짜 및 시간에 도달할 때까지 인스턴스는 종료되지 않습니다.

Elastic Load Balancing 상태 확인

Auto Scaling 그룹에 대해 Elastic Load Balancing 상태 확인을 활성화하면 Amazon EC2 Auto Scaling에서 해당 상태 확인의 결과를 사용하여 인스턴스의 상태를 확인할 수 있습니다.

Auto Scaling 그룹에 대해 Elastic Load Balancing 상태 확인을 활성화하려면 먼저 Elastic Load Balancing 로드 밸런서를 구성하고 상태 점검을 구성하여 인스턴스가 정상인지 확인해야 합니다. 자세한 내용은 Elastic Load Balancing 로드 밸런서 연결 준비 단원을 참조하십시오.

로드 밸런서를 Auto Scaling 그룹에 연결하면 다음과 같은 상황이 발생합니다.

- Amazon EC2 Auto Scaling은 Auto Scaling 그룹의 인스턴스를 로드 밸런서에 등록합니다.
- 인스턴스 등록이 완료되면 `InService` 상태가 되고 로드 밸런서와 함께 사용할 수 있게 됩니다.

기본적으로 Amazon EC2 Auto Scaling은 Elastic Load Balancing 상태 확인의 결과를 무시합니다. Auto Scaling 그룹에 대해 이러한 상태 확인을 활성화한 후 Elastic Load Balancing이 등록된 인스턴스를 로 `Unhealthy` 보고하면 Amazon EC2 Auto Scaling은 해당 인스턴스를 `Unhealthy` 다음 정기 상태 점검 대상으로 표시하고 교체합니다.

로드 밸런서에 연결 드레이닝 (등록 취소 지연) 이 활성화된 경우 Amazon EC2 Auto Scaling은 진행 중인 요청이 완료되거나 최대 제한 시간이 만료될 때까지 기다린 후 비정상 인스턴스가 종료됩니다.

> **Note**
>
> Auto Scaling 그룹에 로드 밸런서를 연결하고 Elastic Load Balancing 상태 확인을 활성화하는 방법에 대한 지침은 을 참조하십시오Auto Scaling 그룹에 Elastic Load Balancing 로드 밸런서를 연결합니다..
> 그룹에 대해 Elastic Load Balancing 상태 확인을 활성화하면 Amazon EC2 Auto Scaling에서 Elastic Load Balancing이 비정상으로 보고한 인스턴스를 교체할 수 있지만, 이는 로드 밸런서가 상태가 된 `InService` 후에만 가능합니다. 자세한 내용은 로드 밸런서의 연결 상태 확인 단원을 참조하십시오.

VPC래티스 상태 점검

기본적으로 Amazon EC2 Auto Scaling은 VPC 래티스 상태 확인 결과를 무시합니다. Auto Scaling 그룹에 대해 이러한 상태 확인을 선택적으로 활성화할 수 있습니다. 이렇게 하면 VPC Lattice에서 등록된 인스턴스를 다음과 같이 `Unhealthy` 보고하면 Amazon EC2 Auto Scaling은 해당 인스턴스를 `Unhealthy` 다음 정기 상태 점검 대상으로 표시하고 교체합니다. 인스턴스를 등록한 다음 상태를 확인하는 프로세스는 Elastic Load Balancing 상태 확인가 작동하는 방식과 동일합니다.

> **Note**
>
> Auto Scaling 그룹에 VPC Lattice 대상 그룹을 연결하고 Auto Scaling 그룹에 대해 VPC Lattice 상태 확인을 켜는 방법에 대한 지침은 을 참조하십시오. Auto Scaling 그룹에 VPC 래티스 대상 그룹을 연결합니다.
> 그룹에 대해 VPC Lattice 상태 확인을 활성화하면 Amazon EC2 Auto Scaling에서 VPC Lattice가 비정상으로 보고한 인스턴스를 대체할 수 있습니다. 단, 대상 그룹이 해당 상태가 된 후에만 가능합니다. `InService` 자세한 내용은 VPCLattice 대상 그룹의 첨부 파일 상태를 확인하십시오. 단원을 참조하십시오.

Amazon EC2 Auto Scaling을 통해 다운타임을 최소화하는 방법

기본적으로 기존 인스턴스가 종료되는 동시에 새 인스턴스가 프로비저닝되므로 새 인스턴스가 완전히 작동할 때까지 새 요청이 수락되지 않을 수 있습니다.

Amazon EC2 Auto Scaling에서 더 이상 실행되지 않는 인스턴스 (또는 `Unhealthy` set-instance-health명령으로 표시된 경우) 가 확인되면 해당 인스턴스를 즉시 교체합니다. 하지만 다른 인스턴스가 비정상인 것으로 확인되면 Amazon EC2 Auto Scaling은 다음 접근 방식을 사용하여 장애를 복구합니다. 이 접근 방식은 일시적인 문제나 잘못 구성된 상태 확인로 인해 발생할 수 있는 가동 중지 시간을 최소화합니다.

- 조정 활동이 진행 중이고 Auto Scaling 그룹이 원하는 용량보다 10% 이상 적은 경우, Amazon EC2 Auto Scaling은 진행 중인 조정 활동을 기다렸다가 비정상 인스턴스를 교체합니다.
- 확장 시 Amazon EC2 Auto Scaling은 인스턴스가 초기 상태 확인을 통과할 때까지 기다립니다. 또한 새 인스턴스가 준비되었는지 확인하기 위해 기본 인스턴스 워밍업이 완료될 때까지 기다립니다.
- 인스턴스 워밍업이 완료되고 그룹이 원하는 용량의 90% 이상으로 증가하면 Amazon EC2 Auto Scaling은 다음과 같이 비정상 인스턴스를 대체합니다.
  - Amazon EC2 Auto Scaling은 한 번에 그룹 희망 용량의 최대 10% 까지만 대체합니다. 비건전 인스턴스가 모두 교체될 때까지 이를 계속합니다.

- 인스턴스 교체 시 새 인스턴스가 초기 상태 확인를 통과할 때까지 기다립니다. 또한 계속하기 전에 기본 인스턴스 워밍업이 완료될 때까지 기다립니다.

> **Note**
>
> Auto Scaling 그룹의 크기가 작아서 결과 값인 10% 가 1보다 작을 경우 Amazon EC2 Auto Scaling은 대신 비정상 인스턴스를 한 번에 하나씩 교체합니다. 이로 인해 그룹에 약간의 가동 중지 시간이 발생할 수 있습니다.
> 또한 Elastic Load Balancing 상태 확인에서 Auto Scaling 그룹의 모든 인스턴스가 비정상으로 보고되고 로드 밸런서가 `InService` 상태에 있는 경우 Amazon EC2 Auto Scaling은 한 번에 더 적은 수의 인스턴스를 비정상으로 표시할 수 있습니다. 이로 인해 한 번에 교체되는 인스턴스 수가 다른 시나리오에 적용된 10%보다 훨씬 적을 수 있습니다. 이렇게 하면 Amazon EC2 Auto Scaling에서 전체 그룹을 자동으로 종료하지 않고도 문제를 해결할 시간을 확보할 수 있습니다.

웜 풀에 있는 인스턴스의 건강 점검

또한 Amazon EC2 Auto Scaling은 웜 풀의 인스턴스에 대한 상태 확인을 수행합니다. 자세한 내용은 건전성 체크의 상태 및 건전성 체크 불합격 이유 보기 단원을 참조하십시오.

상태 확인 고려 사항

Amazon EC2 Auto Scaling 상태 확인을 사용할 때 고려할 사항은 다음과 같습니다.

- 해지 중인 인스턴스나 시작 중인 인스턴스에서 어떤 이벤트가 발생해야 하는 경우, 수명 주기 후크를 사용할 수 있습니다. 이러한 후크를 사용하면 Amazon EC2 Auto Scaling에서 인스턴스를 시작하거나 종료할 때 사용자 지정 작업을 수행할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 수명 주기 후크 단원을 참조하십시오.
- Amazon EC2 Auto Scaling은 상태 확인에서 Amazon EC2 상태 확인 및 예약된 이벤트를 제거하는 방법을 제공하지 않습니다. 인스턴스를 교체하지 않으려면 개별 Auto Scaling 그룹의 `ReplaceUnhealthy` 및 `HealthCheck` 프로세스를 일시 중지하는 것이 좋습니다. 자세한 내용은 Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개 단원을 참조하십시오.
- 비정상 인스턴스의 상태를 수동으로 다시 `Healthy` 설정하려면 set-instance-health명령을 사용해 볼 수 있습니다. 인스턴스가 이미 해지 중이기 때문에 오류가 발생하는 것일 수 있습니다. 일반적으로 set-instance-health명령을 사용하여 인스턴스의 상태를 다시 `Healthy` 로 설정하는 것은 프로세스나 `ReplaceUnhealthy` 프로세스가 일시 중단된 경우에만 유용합니다. `Terminate`

- 상태 확인의 방해 없이 인스턴스 문제를 해결해야 하는 경우 인스턴스를 `Standby` 상태로 전환할 수 있습니다. Amazon EC2 Auto Scaling은 사용자가 인스턴스를 다시 서비스 상태로 전환할 때까지 해당 `Standby` 상태에 있는 인스턴스에 대한 상태 확인을 수행하지 않습니다. 자세한 내용은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 단원을 참조하십시오.
- 인스턴스가 해지되면 연결된 모든 Elastic IP 주소와의 연결이 해제되고 새 인스턴스와 자동으로 연결되지 않습니다. 탄력적 IP 주소를 새 인스턴스와 수동으로 연결하거나 수명 주기 후크 기반 솔루션을 사용하여 자동으로 연결해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 엘라스틱 IP 주소를 참조하십시오.
- 마찬가지로, 인스턴스가 종료되면 연결된 EBS 볼륨이 분리되거나 볼륨의 `DeleteOnTermination` 속성에 따라 삭제됩니다. 이러한 EBS 볼륨을 새 인스턴스에 수동으로 연결하거나 수명 주기 후크 기반 솔루션을 사용하여 자동으로 연결해야 합니다. 자세한 내용은 Amazon EBS 사용 설명서의 인스턴스에 Amazon EBS 볼륨 연결을 참조하십시오.

Auto Scaling 그룹의 상태 확인 유예 기간 설정

Amazon EC2 Auto Scaling 상태 점검에서 `InService` 인스턴스가 비정상인 것으로 확인되면 새 인스턴스로 교체합니다. 상태 확인 유예 기간은 새 인스턴스가 건전하지 않은 것으로 확인된 경우, 해지되기 전에 서비스를 유지하는 최소 시간(초)을 지정합니다.

Elastic Load Balancing 상태 확인이 실패하고 인스턴스가 아직 초기화 중이기 때문인 경우 Amazon EC2 Auto Scaling에서 조치를 취하지 않도록 하는 사용 사례의 예가 될 수 있습니다. Elastic Load Balancing 상태 확인은 인스턴스가 로드 밸런서에 등록될 때부터 병렬로 실행됩니다. 유예 기간을 적용하면 Amazon EC2 Auto Scaling에서 새로 시작된 인스턴스를 `Unhealthy` 표시하고 상태가 된 후 상태 확인을 즉시 통과하지 못하는 경우 불필요하게 인스턴스를 종료하지 않습니다. `InService`

콘솔에서 Auto Scaling 그룹을 생성하는 경우, 기본적으로 상태 확인 유예 기간은 300초입니다. AWS CLI 또는 an을 사용하여 Auto Scaling 그룹을 생성할 때 기본값은 SDK 0초입니다. 값이 0이면 상태 확인 유예 기간이 해제됩니다.

이 값을 너무 높게 설정하면 Amazon EC2 Auto Scaling 상태 확인의 효율성이 떨어집니다. 인스턴스 시작을 위해 수명 주기 후크를 사용하는 경우, 상태 확인 유예 기간을 0으로 설정할 수 있습니다. Amazon EC2 Auto Scaling은 수명 주기 후크를 통해 인스턴스가 `InService` 상태에 들어가기 전에 항상 초기화되도록 하는 방법을 제공합니다. 자세한 내용은 Amazon EC2 Auto Scaling 수명 주기 후크 단원을 참조하십시오.

유예 기간은 다음 인스턴스에 적용됩니다.

- 새로 시작된 인스턴스

- 대기 상태에서 서비스를 다시 시작하는 인스턴스
- 그룹에 수동으로 연결한 인스턴스

> **Important**
>
> 상태 확인 유예 기간 동안 Amazon EC2 Auto Scaling에서 인스턴스가 더 이상 Amazon EC2 `running` 상태가 아님을 감지하면 즉시 해당 인스턴스를 `Unhealthy` 표시하고 교체합니다. 예컨대, Auto Scaling 그룹이 어떤 인스턴스를 중단시키면 그것은 `Unhealthy`로 표식되고 교체됩니다.

그룹의 상태 확인 유예 기간 설정

신규 및 기존 Auto Scaling 그룹의 상태 확인 유예 기간을 설정할 수 있습니다.

Console

새 그룹의 상태 확인 유예 기간을 수정하려면

Auto Scaling 그룹을 생성할 때 고급 옵션 구성 페이지, 상태 점검, 상태 점검 유예 기간에 시간 (초)을 입력합니다. Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다.

AWS CLI

새 그룹의 상태 확인 유예 기간을 수정하려면

create-auto-scaling-group명령에 `--health-check-grace-period` 옵션을 추가합니다. 다음 예에서는 *`my-asg`*라는 새 Auto Scaling 그룹에 대한 상태 확인 유예 기간을 *`60`*초 단위로 구성합니다.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 60 ...
```

Console

기존 그룹의 상태 점검 유예 기간을 수정하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.

2. 화면 상단의 탐색 모음에서 Auto Scaling 그룹을 생성한 AWS 리전 을 선택합니다.
3. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
4. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
5. Health check grace period(상태 확인 유예 기간)에 시간을 초 단위로 입력합니다. Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다.
6. 업데이트를 선택합니다.

AWS CLI

기존 그룹의 상태 점검 유예 기간을 수정하려면

update-auto-scaling-group명령에 `--health-check-grace-period` 옵션을 추가합니다. 다음 예에서는 *my-asg*라는 기존 Auto Scaling 그룹에 대한 상태 확인 유예 기간을 *120*초 단위로 구성합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-grace-period 120
```

> **Note**
>
> Auto Scaling 그룹의 기본 인스턴스 준비 시간도 설정하는 것이 좋습니다. 자세한 내용은 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정 단원을 참조하십시오.

상태 확인을 사용하여 Amazon EBS 볼륨이 손상된 Auto Scaling 인스턴스를 모니터링합니다.

Auto Scaling 그룹의 Amazon EBS 상태 확인을 활성화하여 Amazon EC2 Auto Scaling이 애플리케이션이 실행되는 전체 시스템을 모니터링하도록 할 수 있습니다.

이러한 상태 확인을 활성화하면 Amazon EC2 Auto Scaling은 인스턴스의 연결된 EBS 볼륨에서 수행된 Amazon EC2 상태 확인 결과를 수신합니다. 볼륨에 연결할 수 없거나 I/O 상태 확인을 통과하지 못하면 상태 확인이 실패하고 해당 인스턴스는 비정상으로 간주됩니다. Amazon EC2 Auto Scaling에서 비정상 인스턴스를 감지하면 해당 인스턴스를 대체합니다.

이 주제에서는 첨부된 상태 검사에 익숙하다고 가정합니다. EBS 그렇지 않은 경우 Amazon EC2 사용 설명서의 첨부된 EBS 상태 확인 섹션에서 자세한 내용을 참조하십시오. 다음 주제에서는 첨부된 EBS 상태 확인을 기반으로 Amazon EC2 Auto Scaling 상태 확인을 활성화하는 방법을 설명합니다.

> **Note**
>
> 모든 Auto Scaling 그룹에 대해 Amazon EBS 상태 확인을 활성화할 수 있습니다. 하지만 이러한 상태 점검은 AWS Nitro 시스템에 구축된 인스턴스에만 사용할 수 있습니다.

그룹에 대한 Amazon EBS 상태 점검 켜기

신규 및 기존 Auto Scaling 그룹에 대해 Amazon EBS 상태 확인을 활성화할 수 있습니다.

Console

새 그룹에 대한 Amazon EBS 상태 점검 켜기

Auto Scaling 그룹을 생성할 때 고급 옵션 구성 페이지의 상태 확인, 추가 상태 확인 유형에 대해 Amazon EBS 상태 확인 활성화를 선택합니다. 그런 다음, 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. 이 시간은 Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.

AWS CLI

새 그룹에 대한 Amazon EBS 상태 점검 켜기

create-auto-scaling-group명령에 `--health-check-type` 옵션을 추가합니다. 다음 예제에서는 라는 새 Auto Scaling 그룹의 `--health-check-type` 옵션을 지정합니다 **EBS***my-asg*.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60 ...
```

`--health-check-type`옵션에 여러 값을 지정할 수 있습니다. 예를 들어 Amazon EBS 및 Elastic Load Balancing 상태 확인 유형을 모두 추가하려면 다음 명령을 사용합니다.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS,ELB" --health-check-grace-period 60 ...
```

값 이름은 대/소문자를 구분합니다.

Console

기존 그룹의 Amazon EBS 상태 점검 켜기

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 화면 상단의 탐색 모음에서 Auto Scaling 그룹을 생성한 AWS 리전 을 선택합니다.
3. 기존 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다.
4. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
5. 상태 점검, 추가 상태 점검 유형의 경우 Amazon EBS 상태 확인 활성화를 선택합니다.
6. 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. 이 시간은 Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.
7. 업데이트를 선택합니다.

AWS CLI

기존 그룹의 Amazon EBS 상태 점검 켜기

update-auto-scaling-group명령에 `--health-check-type` 옵션을 추가합니다. 다음 예제에서는 라는 기존 Auto Scaling 그룹의 `--health-check-type` 옵션을 지정합니다 **EBS***my-asg*.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EBS" --health-check-grace-period 60
```

여러 상태 확인 유형을 사용하려면 `--health-check-type` 옵션에 여러 값 (예:**EBS,ELB**) 을 지정할 수 있습니다.

값 이름은 대/소문자를 구분합니다.

Auto Scaling 그룹에 대한 Amazon EBS 상태 확인을 비활성화합니다.

다음 주제에서는 Auto Scaling 그룹에 대한 Amazon EBS 상태 확인을 비활성화하는 방법을 설명합니다. Amazon EBS 상태 확인이 더 이상 필요하지 않은 경우 다음 절차에 따라 상태 점검을 끄십시오.

Console

그룹에 대한 Amazon EBS 상태 점검 끄기

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
4. 상태 점검, 추가 상태 점검 유형의 경우 Amazon EBS 상태 확인 켜기를 선택 취소하십시오.
5. 업데이트를 선택합니다.

AWS CLI

그룹에 대한 Amazon EBS 상태 점검 끄기

Auto Scaling 그룹에서 더 이상 Amazon 상태 확인을 사용하지 않도록 EBS 상태 확인을 update-auto-scaling-group업데이트하려면 명령을 사용하십시오. `--health-check-type` 옵션과 **`EC2`** 값을 포함합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

다른 EBS 상태 확인 유형 (예: Elastic Load Balancing) 을 끄지 않고 Amazon 상태 확인을 끄려면 대신 해당 유형을 지정해야 합니다**`EC2`**. 예를 들어 Elastic Load Balancing 상태 확인의 **`ELB`** 경우 `--health-check-type` 옵션을 지정합니다.

값 이름은 대/소문자를 구분합니다.

Auto Scaling 그룹에 대한 사용자 지정 상태 점검 설정

사용자 지정 상태 확인을 사용하여 Amazon EC2 Auto Scaling에서 제공하는 기존 상태 확인 옵션을 보완할 수 있습니다. 사용자 지정 상태 확인을 다른 상태 확인 유형과 결합하여 애플리케이션 요구 사항에 맞는 포괄적인 상태 모니터링 시스템을 만들 수 있습니다.

시작하려면 사용자 지정 테스트를 생성하여 Auto Scaling 그룹의 인스턴스가 제대로 작동하고 수신 트래픽을 처리할 수 있는지 확인하십시오. 구성한 상태 확인에서 인스턴스가 응답하지 않는 것으로 감지

되면 해당 특정 인스턴스를 로 `Unhealthy` 표시합니다. 그러면 Amazon EC2 Auto Scaling에서 해당 인스턴스를 즉시 대체합니다.

AWS CLI 또는 를 사용하여 Amazon EC2 Auto Scaling에 인스턴스의 상태를 직접 전송할 수 SDK 있습니다. 다음 예는 를 사용하여 인스턴스의 AWS CLI 상태를 구성한 다음 인스턴스의 상태를 확인하는 방법을 보여줍니다.

다음 set-instance-health명령을 사용하여 지정된 인스턴스의 상태를 로 설정합니다**Unhealthy**.

```
aws autoscaling set-instance-health --instance-id i-1234567890abcdef0 --health-
status Unhealthy
```

기본적으로 이 명령은 상태 확인 유예 기간을 준수합니다. 그러나 `--no-should-respect-grace-period` 옵션을 포함하여 이 동작을 무시하고 유예 기간을 준수하지 않을 수 있습니다.

다음 describe-auto-scaling-groups명령을 사용하여 인스턴스의 상태가 상태인지 확인합니다`Unhealthy`.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names my-asg
```

다음은 인스턴스의 상태가 상태이고 인스턴스가 종료 중임을 보여주는 예제 응답입니다. `Unhealthy`

```
{
    "AutoScalingGroups": [
        {
            ....
            "Instances": [
                {
                    "ProtectedFromScaleIn": false,
                    "AvailabilityZone": "us-west-2a",
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcdef0"
                    },
                    "InstanceId": "i-1234567890abcdef0",
                    "InstanceType": "t2.micro",
                    "HealthStatus": "Unhealthy",
                    "LifecycleState": "Terminating"
                },
                ...
```

```
            ]
        }
    ]
}
```

상태 확인 불합격 이유 확인

다음 절차에 따라, 상태 확인로 인해 교체된 모든 인스턴스에 대한 정보를 확인할 수 있습니다.

기본적으로 Amazon EC2 Auto Scaling은 비정상 인스턴스를 종료하기 위한 새로운 조정 활동을 생성한 다음 종료합니다. 인스턴스를 해지하는 동안 다른 크기 조정 활동이 새 인스턴스를 시작합니다. 인스턴스 유지 관리 정책을 사용하여 가능한 한 빨리 새 인스턴스를 시작하도록 이 동작을 변경할 수 있습니다. 자세한 내용은 인스턴스 유지 관리 정책 단원을 참조하십시오.

Console

상태 확인 실패 원인 보기

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹(Auto Scaling groups) 페이지 하단에 분할 창이 열립니다.
3. 활동(Activity) 탭에서 활동 기록(Activity history)의 상태(Status) 열에 Auto Scaling 그룹이 성공적으로 인스턴스를 시작 또는 해지했는지가 표시됩니다.

   비건전 인스턴스를 해지한 경우, 원인(Cause) 열에는 해지 날짜 및 시간과 상태 확인 불합격 이유가 표시됩니다. 예: `At 2022-05-14T20:11:53Z an instance was taken out of service in response to a user health-check`. 이 메시지는 사용자 지정 상태 확인이 인스턴스를 비정상으로 표시했음을 나타냅니다.

   상태 확인 실패에 대한 도움이 필요하면 을 참조하십시오Amazon EC2 Auto Scaling에서 비정상 인스턴스 문제 해결.

AWS CLI

상태 점검 실패 원인 보기

다음 describe-scaling-activities 명령을 사용합니다.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

다음은 응답 예시이며, 여기에는 상태 확인 실패 이유가 `Cause` 포함되어 있습니다.

```
{
  "Activities": [
    {
      "ActivityId": "4c65e23d-a35a-4e7d-b6e4-2eaa8753dc12",
      "AutoScalingGroupName": "my-asg",
      "Description": "Terminating EC2 instance: i-04925c838b6438f14",
      "Cause": "At 2021-04-01T21:48:35Z an instance was taken out of service in
 response to a user health-check.",
      "StartTime": "2021-04-01T21:48:35.859Z",
      "EndTime": "2021-04-01T21:49:18Z",
      "StatusCode": "Successful",
      "Progress": 100,
      "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2a\"...}",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
    },
...
  ]
}
```

출력의 필드에 대한 설명은 Amazon EC2 Auto Scaling API 참조의 활동을 참조하십시오.

Auto Scaling 그룹이 삭제된 후의 조정 활동을 설명하려면 describe-scaling-activities명령에 `--include-deleted-groups` 옵션을 추가하십시오.

Amazon EC2 Auto Scaling에서 비정상 인스턴스 문제 해결

다음은 Amazon EC2 Auto Scaling에서 반환한 오류 메시지, 잠재적 원인, 문제 해결을 위해 취할 수 있는 조치입니다.

오류 메시지를 검색하려면 상태 확인 불합격 이유 확인를 참조하세요.

오류 메시지

- 인스턴스 상태 확인 실패에 대한 응답으로 EC2 인스턴스가 서비스 중단되었습니다.

- 종료 또는 중지된 것으로 표시된 EC2 상태 점검에 따라 인스턴스가 서비스가 중단되었습니다.
- ELB시스템 상태 점검 실패에 대한 대응으로 인스턴스가 서비스가 중단되었습니다.
- 추가 리소스

인스턴스 상태 확인 실패에 대한 응답으로 EC2 인스턴스가 서비스 중단되었습니다.

문제: Auto Scaling 인스턴스가 Amazon EC2 상태 확인에 실패합니다.

원인 1: Amazon이 Auto Scaling 그룹의 인스턴스를 손상된 EC2 것으로 간주하는 문제가 있는 경우, Amazon EC2 Auto Scaling은 상태 확인의 일환으로 인스턴스를 자동으로 교체합니다.

해결 방법 1: 인스턴스 상태 확인이 실패하는 경우 일반적으로 애플리케이션에서 더 이상 문제가 나타나지 않을 때까지 인스턴스 구성을 변경하여 문제를 직접 해결해야 합니다. 이 문제를 해결하려면 다음 단계에 따릅니다.

1. Auto Scaling 그룹에 속하지 않는 Amazon EC2 인스턴스를 수동으로 생성하고 문제를 조사하십시오. 손상된 인스턴스를 조사하는 데 대한 일반적인 도움이 필요하면 Amazon 사용 설명서의 상태 확인에 실패한 인스턴스 문제 해결 및 Amazon EC2 사용 설명서의 Windows 인스턴스 문제 해결을 참조하십시오. EC2
2. 인스턴스가 성공적으로 시작되고 정상임을 확인한 후 오류가 없는 새로운 인스턴스 구성을 Auto Scaling 그룹에 배치합니다.
3. 생성한 인스턴스를 삭제해 자신의 AWS 계정에 요금이 계속해서 부과되는 것을 피할 수 있습니다.

종료 또는 중지된 것으로 표시된 EC2 상태 점검에 따라 인스턴스가 서비스가 중단되었습니다.

문제: 중지, 재부팅 또는 해지된 Auto Scaling 인스턴스가 교체됩니다.

원인 1: 사용자가 수동으로 인스턴스를 중지, 재부팅 또는 해지했습니다.

해결 방법 1: Auto Scaling 그룹의 인스턴스를 중지하거나 재부팅해야 하는 경우 먼저 인스턴스를 대기 상태로 두는 것이 좋습니다. 자세한 내용은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거 단원을 참조하십시오.

원인 2: Amazon EC2 Auto Scaling은 Amazon 스팟 서비스가 인스턴스를 중단한 후 EC2 스팟 인스턴스를 교체하려고 합니다. 스팟 가격이 최고 가격보다 인상되거나 용량을 더 이상 사용할 수 없기 때문입니다.

솔루션 2: 특정 시점에 요청을 이행하기 위해 스팟 인스턴스가 존재한다고 보장할 수 없습니다. 그러나 다음과 같은 방법을 시도할 수 있습니다.

- 더 높은 스팟 최고가(온디맨드 가격일 수 있음)를 사용합니다. 최고 가격을 더 높게 설정하면 Amazon EC2 Spot 서비스를 시작하고 필요한 용량을 유지할 가능성이 높아집니다.
- 여러 가용 영역에서 여러 인스턴스 유형을 실행하여 인스턴스를 시작할 수 있는 다양한 용량 풀의 수를 늘립니다. 자세한 설명은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.
- 여러 인스턴스 유형을 사용하는 경우, 용량 재조정 기능을 사용하도록 설정하는 것이 좋습니다. 이는 Amazon EC2 스팟 서비스가 실행 중인 인스턴스가 종료되기 전에 새 스팟 인스턴스를 시작하려고 시도하려는 경우에 유용합니다. 자세한 내용은 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리 단원을 참조하십시오.

원인 3: 용량 블록을 사용하면 Amazon은 용량 블록 종료 시간 30분 전에 아직 실행 중인 모든 인스턴스를 EC2 종료합니다. 이러한 갑작스러운 종료로 인해 Auto Scaling 그룹은 용량 블록이 종료되더라도 원하는 용량을 유지하기 위해 새 인스턴스를 시작하려고 합니다.

해결 방법 3: 이 문제를 해결하려면 다음을 시도해 보십시오.

- Auto Scaling 그룹에서 새 인스턴스를 시작하려고 시도하지 않도록 하려면 원하는 용량을 줄이십시오. 자세한 내용은 Amazon EC2 Auto Scaling을 위한 수동 스케일링 단원을 참조하십시오.
- 이 오류가 자주 발생하지 않도록 용량 블록 종료 시간 30분 전에 Auto Scaling 그룹을 확장해야 합니다. 모든 라이프사이클 후크가 용량 블록 종료 시간 30분 전에 완료되었는지 확인하십시오. 자세한 내용은 사용 Capacity Blocks 기계 학습 워크로드용 단원을 참조하십시오.

ELB시스템 상태 점검 실패에 대한 대응으로 인스턴스가 서비스가 중단되었습니다.

문제: Auto Scaling 인스턴스가 EC2 상태 검사를 통과할 수 있습니다. 그러나 해당 인스턴스가 Auto Scaling 그룹이 등록된 대상 그룹 또는 Classic Load Balancers에 대한 Elastic Load Balancing 건전성 체크에는 실패할 수 있습니다.

원인 1: Auto Scaling 그룹이 Elastic Load Balancing에서 제공하는 상태 확인에 의존하는 경우 Amazon EC2 Auto Scaling은 상태 확인과 Elastic Load Balancing EC2 상태 확인의 결과를 모두 확인하여 인스턴스의 상태를 확인합니다. 로드 밸런서는 각 인스턴스에 요청을 보내고 올바른 응답을 기다리거나 인스턴스와 연결을 설정하여 건전성 체크를 수행합니다. 인스턴스에서 실행 중인 애플리케이션에는 로드 밸런서가 인스턴스를 서비스에서 제외시키는 것을 고려하게 하는 문제가 있기 때문에 인스턴스가 Elastic Load Balancing 건전성 체크에 실패할 수 있습니다.

솔루션 1: Elastic Load Balancing 건전성 체크를 통과하려면:

- 대상 그룹의 건전성 체크 설정이 올바르게 구성되었는지 확인합니다. 대상 그룹당 로드 밸런서의 건전성 체크 설정을 정의합니다. 자세한 내용은 대상의 상태 확인을 구성하십시오. 단원을 참조하십시오.
- 로드 밸런서에 필요한 성공 코드를 기록해 두고 성공 시 이들 코드를 반환하도록 애플리케이션이 올바르게 구성되어 있는지 확인합니다.
- 로드 밸런서 및 Auto Scaling 그룹의 보안 그룹이 올바르게 구성되었는지 확인합니다.
- 로드 밸런서가 Auto Scaling 그룹과 동일한 가용 영역에서 구성되어 있는지 확인합니다.

솔루션 2: Elastic Load Balancing 건전성 체크를 비활성화하도록 Auto Scaling 그룹을 업데이트합니다. 이러한 상태 확인을 비활성화하는 방법에 대한 지침은 을 참조하십시오Auto Scaling 그룹에 Elastic Load Balancing 로드 밸런서를 연결합니다..

원인 2: 건전성 체크 유예 기간과 인스턴스 시작 시간 간에 불일치가 있습니다.

해결 방법 3: Auto Scaling 그룹의 상태 점검 유예 기간을 편집합니다. Elastic Load Balancing이 새로 시작된 인스턴스를 정상으로 간주하기 전에 필요한 연속적인 성공 상태 확인 횟수를 지원할 수 있도록 충분한 기간으로 유예 기간을 설정하십시오. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.

추가 리소스

다른 문제가 있는 경우 다음 AWS re:Post 문서에서 추가 문제 해결 도움말을 참조하십시오.

- Amazon EC2 Auto Scaling이 인스턴스를 종료한 이유는 무엇입니까?
- Amazon EC2 Auto Scaling에서 비정상 인스턴스를 종료하지 않은 이유는 무엇입니까?

AWS Health Dashboard Amazon EC2 Auto Scaling에 대한 알림

귀하는 AWS Health Dashboard Amazon EC2 Auto Scaling에서 오는 알림을 지원합니다. 이러한 알림은 애플리케이션에 영향을 줄 수 있는 리소스 성능 또는 가용성 문제를 알리고 수정 지침을 제시합니다. 지금은 누락된 보안 그룹 및 시작 템플릿과 관련된 이벤트만 사용할 수 있습니다.

AWS Health 서비스의 AWS Health Dashboard 일부입니다. 설정이 필요하지 않으며, 계정에서 인증된 사용자면 누구나 볼 수 있습니다. 자세한 내용은 AWS Health 대시보드 시작하기를 참조하십시오.

다음 메시지와 유사한 메시지를 받으면 조치를 하기 위한 경보로 처리해야 합니다.

예: 누락된 보안 그룹 때문에 Auto Scaling 그룹이 확장되지 않음

```
 Hello,

 At 2020-01-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
AWS ## 123456789012.

 A security group associated with this Auto Scaling group cannot be found. Each time
a
 scale out operation is performed, it will be prevented until you make a change that
 fixes the issue.

 We recommend that you review and update your Auto Scaling group configuration to
change
 the launch template or launch configuration that depends on the unavailable security
 group.

 Sincerely,
 Amazon Web Services
```

예: 누락된 시작 템플릿 때문에 Auto Scaling 그룹이 확장되지 않음

```
 Hello,

 At 2021-05-11 04:00 UTC, we detected an issue with your Auto Scaling group [ARN] in
AWS ## 123456789012.

 The launch template associated with this Auto Scaling group cannot be found. Each
time
 a scale out operation is performed, it will be prevented until you make a change
that
 fixes the issue.

 We recommend that you review and update your Auto Scaling group configuration and
 specify an existing launch template to use.

 Sincerely,
 Amazon Web Services
```

Auto Scaling 그룹 및 인스턴스의 CloudWatch 메트릭을 모니터링합니다.

지표는 Amazon의 기본 CloudWatch 개념입니다. 지표는 게시되는 데이터 요소를 시간순으로 정렬한 것을 나타냅니다. CloudWatch 지표는 모니터링할 변수로, 데이터 요소는 시간에 따른 변수의 값을 나타내는 것으로 간주합니다. 이러한 지표를 사용하여 시스템이 예상대로 수행되고 있는지 확인할 수 있습니다.

Auto Scaling 그룹에 대한 정보를 수집하는 Amazon EC2 Auto Scaling 지표는 `AWS/AutoScaling` 네임스페이스에 있습니다. Auto Scaling 그룹 인스턴스에서 CPU 및 기타 사용량 데이터를 수집하는 Amazon EC2 인스턴스 지표는 `AWS/EC2` 네임스페이스에 있습니다.

Amazon EC2 Auto Scaling 콘솔에는 그룹 지표와 그룹에 대해 집계된 인스턴스 지표를 보여주는 일련의 그래프가 표시됩니다. 필요에 따라 Amazon EC2 Auto Scaling 콘솔 CloudWatch 대신 Amazon에서 Auto Scaling 그룹 및 인스턴스에 대한 데이터에 액세스하는 것을 선호할 수도 있습니다.

자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

내용

- Amazon EC2 Auto Scaling 콘솔에서 모니터링 그래프 보기
- 아마존 EC2 CloudWatch Auto Scaling용 아마존 메트릭스
- Auto Scaling 인스턴스에 대한 모니터링 구성

Amazon EC2 Auto Scaling 콘솔에서 모니터링 그래프 보기

Amazon EC2 콘솔의 Amazon EC2 Auto Scaling 섹션에서 지표를 사용하여 개별 Auto Scaling 그룹의 진행 상황을 minute-by-minute 모니터링할 수 있습니다. CloudWatch

다음과 같은 타입의 지표를 모니터링할 수 있습니다.

- Auto Scaling 지표 - Auto Scaling 지표는 사용 설정된 경우에만 켜집니다. 자세한 설명은 Auto Scaling 그룹 지표 활성화(콘솔) 섹션을 참조하세요. Auto Scaling 지표가 사용 설정되면 모니터링 그래프는 Auto Scaling 지표에 대해 1분 단위로 게시된 데이터를 표시합니다.
- EC2 지표 — Amazon EC2 인스턴스 지표는 항상 활성화됩니다. 세부 모니터링이 사용 설정된 경우, 모니터링 그래프는 인스턴스 지표에 대해 1분 단위로 게시된 데이터를 표시합니다. 자세한 설명은 Auto Scaling 인스턴스에 대한 모니터링 구성 섹션을 참조하세요.

Amazon EC2 Auto Scaling 콘솔을 사용하여 모니터링 그래프 보기

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. 지표를 보려는 Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling groups(Auto Scaling 그룹) 페이지 아래쪽에 분할 창이 열립니다.
3. 모니터링 탭을 선택합니다.

   Amazon EC2 Auto Scaling은 Auto Scaling 지표에 대한 모니터링 그래프를 표시합니다.
4. 그룹에 대해 집계된 인스턴스 지표의 모니터링 그래프를 보려면 EC2를 선택합니다.

그래프 작업

- 데이터 요소를 가리키면 특정 시간(UTC)에 대한 데이터 팝업이 표시됩니다.
- 그래프를 확대하려면 그래프의 오른쪽 상단에 있는 메뉴 도구(세로 점 3개)에서 확대(Enlarge)를 선택합니다. 또는 그래프 상단에서 최대화 아이콘을 선택합니다.
- 미리 정의된 기간 값 중 하나를 선택하여 그래프에 표시되는 데이터의 기간을 조정합니다. 그래프가 확대되면 사용자 정의(Custom)를 선택하여 기간을 직접 정의할 수 있습니다.
- 메뉴 도구에서 새로 고침(Refresh)을 선택하여 그래프의 데이터를 업데이트합니다.
- 그래프 데이터 위로 커서를 끌어 특정 범위를 선택합니다. 그런 다음 메뉴 도구에서 시간 범위 적용(Apply time range)을 선택할 수 있습니다.
- 콘솔에서 관련 로그 스트림 (있는 경우) 을 보려면 메뉴 도구에서 로그 보기를 선택합니다. CloudWatch
- 그래프를 보려면 메뉴 도구에서 지표로 보기를 선택합니다. CloudWatch 그러면 해당 그래프의 CloudWatch 페이지로 이동합니다. 여기에서 더 많은 정보를 보거나 기록 정보에 액세스하여 Auto Scaling 그룹이 장기간에 걸쳐 어떻게 변경되었는지 더 잘 파악할 수 있습니다.

Auto Scaling 그룹에 대한 그래프 지표

Auto Scaling 그룹을 생성한 후 Amazon EC2 Auto Scaling 콘솔을 열고 모니터링(Monitoring) 탭에서 그룹에 대한 모니터링 그래프를 봅니다.

Auto Scaling 섹션에 있는 그래프 지표에는 다음과 같은 지표가 포함됩니다. 이러한 지표는 해지 인스턴스 수 또는 보류 중인 인스턴스 수와 같은 잠재적 문제의 지표가 될 수 있는 측정값을 제공합니다. 아

마존 EC2 CloudWatch Auto Scaling용 아마존 메트릭스에서 이러한 지표의 정의를 확인할 수 있습니다.

표시 명칭	CloudWatch 지표 이름
최소 그룹 크기	`GroupMinSize`
최대 그룹 크기	`GroupMaxSize`
원하는 용량	`GroupDesiredCapacity`
서비스 상태의 인스턴스	`GroupInServiceInstances`
보류 중인 인스턴스	`GroupPendingInstances`
대기 인스턴스	`GroupStandbyInstances`
해지 중인 인스턴스	`GroupTerminatingInstances`
인스턴스 합계	`GroupTotalInstances`

EC2 섹션에서 Amazon EC2 인스턴스에 대한 주요 성능 지표에 근거하여 하는 다음 그래프 지표를 찾을 수 있습니다. 이러한 EC2 지표는 그룹의 모든 인스턴스에 대한 지표의 집계입니다. Amazon EC2 사용 설명서의 인스턴스에 사용할 수 있는 CloudWatch 지표 나열에서 이러한 지표에 대한 정의를 찾을 수 있습니다.

표시 명칭	CloudWatch 메트릭 이름
CPU 활용도	`CPUUtilization`
디스크 읽기	`DiskReadBytes`
디스크 읽기 작업	`DiskReadOps`
디스크 쓰기	`DiskWriteBytes`
디스크 쓰기 작업	`DiskWriteOps`
네트워크 입력	`NetworkIn`

표시 명칭	CloudWatch 메트릭 이름
네트워크 출력	`NetworkOut`
건전성 체크 불합격(해당되는 경우)	`StatusCheckFailed`
건전성 체크 불합격(인스턴스)	`StatusCheckFailed_Instance`
건전성 체크 불합격(시스템)	`StatusCheckFailed_System`

또한 일부 지표는 Auto Scaling 그래프 지표의 특정 사용 사례에 사용할 수 있습니다.

다음 지표는 각 인스턴스가 그룹의 원하는 용량에 기여하는 단위 수를 정의하는 가중치를 갖는 그룹에 유용합니다. 아마존 EC2 CloudWatch Auto Scaling용 아마존 메트릭스에서 이러한 지표의 정의를 확인할 수 있습니다.

표시 명칭	CloudWatch 지표 이름
서비스 용량 단위	`GroupInServiceCapacity`
보류 용량 단위	`GroupPendingCapacity`
대기 용량 단위	`GroupStandbyCapacity`
해지 중인 용량 단위	`GroupTerminatingCapacity`
총 용량 단위	`GroupTotalCapacity`

그룹에서 웜 풀 기능을 사용하는 경우, 다음 지표이 유용합니다. 아마존 EC2 CloudWatch Auto Scaling용 아마존 메트릭스에서 이러한 지표의 정의를 확인할 수 있습니다.

표시 명칭	CloudWatch 지표 이름
웜 풀 최소 크기	`WarmPoolMinSize`
웜 풀 원하는 용량	`WarmPoolDesiredCapacity`

표시 명칭	CloudWatch 지표 이름
웜 풀 보류 중인 용량 단위	`WarmPoolPendingCapacity`
웜 풀 해지 용량 단위	`WarmPoolTerminatingCapacity`
웜 풀 워밍 용량 단위	`WarmPoolWarmedCapacity`
웜 풀 출범된 총 용량 단위	`WarmPoolTotalCapacity`
그룹 및 웜 풀 원하는 용량	`GroupAndWarmPoolDesiredCapacity`
그룹 및 웜 풀 출범된 총 용량 단위	`GroupAndWarmPoolTotalCapacity`

관련 리소스

- 인스턴스별 지표를 모니터링하려면 Amazon EC2 사용 설명서의 인스턴스에 대한 그래프 지표를 참조하십시오.
- CloudWatch 대시보드는 콘솔에서 사용자 지정할 수 있는 홈 페이지입니다. CloudWatch 이 페이지를 사용하면 여러 지역에 분산된 리소스까지 포함한 리소스를 단일 보기에서 모니터링할 수 있습니다. CloudWatch 대시보드를 사용하여 리소스에 대한 지표 및 경보에 대한 사용자 지정 보기를 만들 수 있습니다. AWS 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

아마존 EC2 CloudWatch Auto Scaling용 아마존 메트릭스

Amazon EC2 Auto Scaling은 `AWS/AutoScaling` 네임스페이스에 다음 지표를 게시합니다. 실제로 사용할 수 있는 Auto Scaling 그룹 지표는 그룹 지표를 사용하도록 설정했는지 여부와 활성화한 그룹 지표에 따라 달라집니다. 그룹 지표는 추가 비용 없이 1분 단위로 사용할 수 있지만 활성화해야 합니다.

Auto Scaling 그룹 지표를 활성화하면 Amazon EC2 Auto Scaling에서 최선을 다해 샘플 데이터를 CloudWatch 1분마다 전송합니다. 드문 CloudWatch 경우이긴 하지만, 서비스 중단이 발생하는 경우, 그룹 지표 기록의 공백을 메우기 위해 데이터가 다시 채워지지 않습니다.

내용

- Auto Scaling 그룹 지표
- Auto Scaling 그룹 지표를 위한 차원

- 예측 조정 지표 및 차원
- Auto Scaling 그룹 지표 활성화(콘솔)
- Auto Scaling 그룹 지표 활성화(AWS CLI)

Auto Scaling 그룹 지표

이러한 지표를 사용하면 시간 경과에 따른 그룹 크기 변화와 같은 Auto Scaling 그룹 기록을 거의 계속적으로 확인할 수 있습니다.

지표	설명
`GroupMinSize`	Auto Scaling 그룹의 최소 크기입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`GroupMaxSize`	Auto Scaling 그룹의 최대 크기입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`GroupDesiredCapacity`	Auto Scaling 그룹에서 정비를 시도하는 인스턴스의 수입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`GroupInServiceInstances`	Auto Scaling 그룹의 일부로 실행되는 인스턴스의 수입니다. 이 지표에는 보류 중이거나 해지되는 인스턴스가 포함되지 않습니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`GroupPendingInstances`	보류 중인 인스턴스의 수입니다. 보류 중인 인터페이스는 아직 서비스되지 않습니다. 이 지표에는 서비스되거나 해지되는 인스턴스가 포함되지 않습니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`GroupStandbyInstances`	`Standby` 상태에 있는 인스턴스의 수입니다. 이 상태의 인스턴스는 계속해서 실행되지만 적극적으로 서비스되지는 않습니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.

지표	설명
GroupTerminatingInstances	해지 과정이 진행 중인 인스턴스의 수입니다. 이 지표에는 서비스되거나 보류 중인 인스턴스가 포함되지 않습니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
GroupTotalInstances	Auto Scaling 그룹에 있는 총 인스턴스 수입니다. 이 지표는 서비스되거나, 보류 중이거나, 해지되는 인스턴스의 수를 식별합니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.

각 인스턴스 타입의 vCPU 수에 따라 가중치를 할당하는 등 원하는 용량을 다른 단위로 측정하도록 혼합 인스턴스 그룹을 구성하는 경우, 다음 지표는 Auto Scaling 그룹에서 사용하는 단위 수를 계산합니다. 원하는 용량을 다른 단위로 측정하도록 혼합 인스턴스 그룹을 구성하지 않은 경우, 다음 지표이 채워지지만 이는 이전 표에 정의된 지표과 동일합니다. 자세한 설명은 혼합 인스턴스 그룹 생성을 위한 설정 개요 섹션을 참조하세요.

측정치	설명
GroupInServiceCapacity	Auto Scaling 그룹의 일부로 실행 중인 용량 단위의 수입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
GroupPendingCapacity	보류 중인 용량 단위의 수입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
GroupStandbyCapacity	Standby 상태에 있는 용량 단위의 수입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
GroupTerminatingCapacity	해지 과정이 진행 중인 용량 단위의 수입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
GroupTotalCapacity	Auto Scaling 그룹의 총 용량 단위 수입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.

Amazon EC2 Auto Scaling은 웜 풀이 있는 Auto Scaling 그룹에 대한 다음 지표도 보고합니다. 자세한 설명은 웜 풀을 사용하여 부팅 시간이 긴 애플리케이션의 지연 시간을 줄입니다. 섹션을 참조하세요.

측정치	설명
`WarmPoolMinSize`	웜 풀의 최소 크기입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`WarmPoolDesiredCapacity`	Amazon EC2 Auto Scaling이 웜 풀에서 유지하려고 하는 용량입니다. 이는 Auto Scaling 그룹의 최대 크기에서 원하는 용량을 뺀 값 또는 Auto Scaling 그룹의 최대 준비 용량에서 원하는 용량을 뺀 값(설정된 경우)과 같습니다. 단, 웜 풀의 최소 크기가 Auto Scaling 그룹의 최대 크기 또는 최대 준비 용량(설정된 경우)과 원하는 용량의 차이 이상인 경우, 웜 풀의 원하는 용량은 `WarmPoolMinSize` 와 동일합니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`WarmPoolPendingCapacity`	보류 중인 웜 풀의 용량입니다. 이 지표에는 실행 중이거나 중지되었거나 해지 중인 인스턴스가 포함되지 않습니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`WarmPoolTerminatingCapacity`	해지 중인 웜 풀의 용량입니다. 이 지표에는 실행 중이거나 중지되었거나 보류 중인 인스턴스가 포함되지 않습니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`WarmPoolWarmedCapacity`	축소 중 Auto Scaling 그룹에 들어갈 수 있는 용량입니다. 이 지표에는 보류 중이거나 해지되는 인스턴스가 포함되지 않습니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`WarmPoolTotalCapacity`	실행 중, 중지됨, 보류 중 또는 해지 중인 인스턴스를 포함한 웜 풀의 총 용량입니다.

측정치	설명
	보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`GroupAndWarmPoolDesiredCapacity`	Auto Scaling 그룹과 웜 풀의 원하는 용량을 합친 용량입니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.
`GroupAndWarmPoolTotalCapacity`	Auto Scaling 그룹과 웜 풀의 원하는 용량을 합친 총 용량입니다. 여기에는 실행 중, 중지됨, 보류 중, 해지 중 또는 서비스 중인 인스턴스가 포함됩니다. 보고 기준: 지표 모음이 활성화된 경우, 보고됩니다.

Auto Scaling 그룹 지표를 위한 차원

다음 차원을 사용하여 이전 표에 열거된 지표를 구체화할 수 있습니다.

차원	설명
`AutoScalingGroupName`	Auto Scaling 그룹의 이름을 필터링합니다.

예측 조정 지표 및 차원

`AWS/AutoScaling` 네임스페이스에는 예측 조정에 대한 다음 지표가 포함됩니다.

지표는 1시간의 분해능으로 제공됩니다.

예측된 값과 실제 값을 비교하여 예측 정확도를 평가할 수 있습니다. 이러한 지표를 사용한 예측 정확도 평가에 대한 자세한 설명은 다음을 사용하여 예측 규모 조정 지표를 모니터링할 수 있습니다. CloudWatch 섹션을 참조하세요.

지표	설명	차원
`PredictiveScalingLoadForecast`	애플리케이션에서 생성될 것으로 예상되는 로드의 양입니다.	`AutoScalingGroupName` , `PolicyName` , `PairIndex`

지표	설명	차원
	Average, Minimum 및 Maximum 통계는 유용하지만 Sum 통계는 유용하지 않습니다. 보고 기준: 초기 예측이 생성된 후에 보고됩니다.	
PredictiveScalingCapacityForecast	애플리케이션 수요를 충족하는 데 필요할 것으로 예상되는 용량입니다. 이는 Auto Scaling 그룹 인스턴스를 유지하려는 로드 예측 및 목표 사용률 수준을 기준으로 합니다. Average, Minimum 및 Maximum 통계는 유용하지만 Sum 통계는 유용하지 않습니다. 보고 기준: 초기 예측이 생성된 후에 보고됩니다.	AutoScalingGroupName , PolicyName
PredictiveScalingMetricPairCorrelation	조정 지표와 로드 지표의 인스턴스당 평균 사이의 상관 관계입니다. 예측적 조정은 높은 상관 관계를 가정합니다. 따라서 이 지표의 값이 낮은 경우, 지표 페어를 사용하지 않는 것이 좋습니다. Average, Minimum 및 Maximum 통계는 유용하지만 Sum 통계는 유용하지 않습니다. 보고 기준: 초기 예측이 생성된 후에 보고됩니다.	AutoScalingGroupName , PolicyName , PairIndex

> **Note**
>
> PairIndex 차원은 Amazon EC2 Auto Scaling에 의해 할당된 로드 조정 지표 페어의 인덱스와 관련한 정보를 반환합니다. 현재 유일한 유효 값은 0입니다.

Auto Scaling 그룹 지표 활성화(콘솔)

그룹 지표를 활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. Monitoring(모니터링) 탭에서 Auto Scaling(자동 크기 조정) 아래 페이지 맨 위에 있는 Auto Scaling group metrics collection(Auto Scaling 그룹 지표 수집), Enable(활성화) 확인란을 선택합니다.

그룹 지표를 비활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. Auto Scaling 그룹을 선택합니다.
3. Monitoring(모니터링) 탭에서 Auto Scaling group metrics collection(Auto Scaling 그룹 지표 수집), Enable(활성화) 확인란을 선택 취소합니다.

Auto Scaling 그룹 지표 활성화(AWS CLI)

Auto Scaling 그룹 지표를 활성화하려면

enable-metrics-collection 명령을 사용하여 한 개 이상의 그룹 지표를 활성화합니다. 예컨대, 다음 명령은 지정된 Auto Scaling 그룹에 대해 단일 지표를 사용하도록 설정합니다.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \
  --metrics GroupDesiredCapacity --granularity "1Minute"
```

`--metrics` 옵션을 생략하면 모든 지표가 활성화됩니다.

```
aws autoscaling enable-metrics-collection --auto-scaling-group-name my-asg \
  --granularity "1Minute"
```

Auto Scaling 그룹 지표를 비활성화하려면

모든 그룹 지표를 비활성화하려면 disable-metrics-collection 명령을 사용합니다.

```
aws autoscaling disable-metrics-collection --auto-scaling-group-name my-asg
```

Auto Scaling 인스턴스에 대한 모니터링 구성

Amazon EC2는 인스턴스에서 원시 데이터를 수집하여 Auto Scaling 그룹의 CPU 및 기타 사용량 데이터를 설명하는 지표로 처리합니다. 1분 또는 5분의 세분화 수준을 선택하여 이러한 지표를 모니터링하는 간격을 구성할 수 있습니다.

인스턴스 모니터링은 인스턴스가 시작될 때마다 활성화되며, 기본 모니터링(5분 단위) 또는 세부 모니터링(1분 단위)을 사용합니다. 세부 모니터링의 경우, 추가 요금이 부과됩니다. 자세한 내용은 Amazon CloudWatch 요금 및 Amazon EC2 사용 CloudWatch 설명서의 인스턴스 모니터링을 참조하십시오.

Auto Scaling 그룹을 생성하기 전에, 애플리케이션에 적합한 모니터링 타입을 허용하는 출범 템플릿 또는 출범 구성을 생성해야 합니다. 그룹에 조정 정책을 추가하는 경우, 세부 모니터링을 사용해 EC2 인스턴스에 대한 지표 데이터를 1분 간격으로 가져오도록 하면 로드의 변화에 빠르게 응답할 수 있습니다.

내용

- 세부 모니터링 활성화(콘솔)
- 세부 모니터링 활성화(AWS CLI)
- 기본 모니터링과 세부 모니터링 간 전환
- 에이전트를 사용하여 추가 지표를 수집하세요. CloudWatch

세부 모니터링 활성화(콘솔)

기본 모니터링은 를 사용하여 시작 템플릿 또는 시작 구성을 생성할 때 기본적으로 활성화됩니다. AWS Management Console

출범 템플릿에서 세부 모니터링을 활성화하려면

를 사용하여 시작 템플릿을 만드는 경우 고급 세부 정보 섹션의 세부 CloudWatch 모니터링에서 활성화를 선택합니다. AWS Management Console그렇지 않으면 기본 모니터링이 활성화됩니다. 자세한 설명은 고급 설정을 사용하여 시작 템플릿 생성 섹션을 참조하세요.

출범 구성에서 세부 모니터링을 활성화하려면

를 사용하여 시작 구성을 생성하는 경우 추가 구성 섹션에서 내 CloudWatch EC2 인스턴스 세부 모니터링 활성화를 선택합니다. AWS Management Console그렇지 않으면 기본 모니터링이 활성화됩니다. 자세한 설명은 시작 구성 생성 섹션을 참조하세요.

세부 모니터링 활성화(AWS CLI)

기본 모니터링은 기본적으로 AWS CLI을 사용하여 출범 템플릿을 생성할 때 활성화됩니다. AWS CLI를 사용하여 출범 구성을 생성하면 기본적으로 세부 모니터링이 활성화됩니다.

출범 템플릿에서 세부 모니터링을 활성화하려면

출범 템플릿의 경우, create-launch-template 명령을 사용하고 출범 템플릿 생성을 위한 정보를 포함하는 JSON 파일을 전달합니다. 모니터링 특성을 `"Monitoring":{"Enabled":true}`로 설정하여 세부 모니터링을 활성화하거나 `"Monitoring":{"Enabled":false}`로 설정하여 기본 모니터링을 활성화합니다.

출범 구성에서 세부 모니터링을 활성화하려면

출범 구성의 경우, `--instance-monitoring` 옵션으로 create-launch-configuration 명령을 사용합니다. 이 옵션을 `true`로 설정하여 세부 모니터링을 활성화하거나 `false`로 설정하여 기본 모니터링을 활성화합니다.

```
--instance-monitoring Enabled=true
```

기본 모니터링과 세부 모니터링 간 전환

새 EC2 인스턴스에서 활성화되는 모니터링 타입을 변경하려면 출범 템플릿을 업데이트하거나, Auto Scaling 그룹을 업데이트하여 새로운 출범 템플릿 도는 출범 구성을 사용하게 합니다. 기존 인스턴스는 이전에 활성화된 모니터링 타입을 계속 사용합니다. 모든 인스턴스를 업데이트하려면 인스턴스를 해지해서 Auto Scaling 그룹에서 교체하게 하거나, monitor-instances 및 unmonitor-instances를 사용하여 인스턴스를 개별적으로 업데이트하세요.

> **Note**
>
> 인스턴스 새로 고침 및 최대 인스턴스 수명 기능을 사용하면 Auto Scaling 그룹의 모든 인스턴스를 교체하여 새 설정을 사용하는 새 인스턴스를 출범할 수도 있습니다. 자세한 설명은 Auto Scaling 그룹의 인스턴스를 교체합니다. 섹션을 참조하세요.

기본 모니터링과 세부 모니터링 간에 전환할 때:

Auto Scaling 그룹의 단계별 조정 정책 또는 단순 조정 정책과 관련된 CloudWatch 경보가 있는 경우 put-metric-alarm 명령을 사용하여 각 경보를 업데이트하십시오. 각 기간이 모니터링 타입과 맞는지 확

인하세요(기본 모니터링의 경우, 300초, 세부 모니터링의 경우, 60초). 세부 모니터링에서 기본 모니터링으로 변경하면서 5분 기간과 일치하도록 경보를 업데이트하지 않으면 경보가 계속해서 1분마다 통계를 검사합니다. 따라서 전체 5개 기간 중 4개에서 사용 가능한 데이터를 찾지 못할 수 있습니다.

에이전트를 사용하여 추가 지표를 수집하세요. CloudWatch

사용 가능한 메모리 및 사용된 메모리와 같은 운영 체제 수준 메트릭을 수집하려면 에이전트를 설치해야 합니다. CloudWatch 추가 요금이 적용될 수 있습니다. CloudWatch 에이전트를 사용하여 Amazon EC2 인스턴스에서 시스템 지표와 로그 파일을 모두 수집할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서에서 CloudWatch 에이전트가 수집한 지표를 참조하십시오.

를 사용하여 Amazon EC2 Auto Scaling API 통화 로깅 AWS CloudTrail

Amazon EC2 Auto Scaling은 사용자AWS CloudTrail, 역할 또는 AWS 서비스. CloudTrail capture가 Auto Scaling을 이벤트로 API 호출하는 작업에 대한 레코드를 제공하는 서비스인 와 통합됩니다. 캡처된 호출에는 의 AWS Management Console 호출과 Auto Scaling API 작업에 대한 코드 호출이 포함됩니다. 에서 수집한 정보를 사용하여 Auto Scaling 수행된 요청 CloudTrail, 요청이 수행된 IP 주소, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부.
- IAM Identity Center 사용자를 대신하여 요청이 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

CloudTrail 는 계정을 생성할 AWS 계정 때 에서 활성화되며 CloudTrail 이벤트 기록 에 자동으로 액세스할 수 있습니다. CloudTrail 이벤트 기록은 에서 기록된 관리 이벤트의 지난 90일에 대한 보기 가능하고, 검색 가능하고, 다운로드 가능하고, 변경할 수 없는 레코드를 제공합니다 AWS 리전. 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 기록 작업을 참조하세요. 이벤트 기록 보기에는 요금이 부과되지 않습니다 CloudTrail.

AWS 계정 지난 90일 동안 진행 중인 이벤트 레코드의 경우 추적을 생성합니다.

CloudTrail 추적

추적을 사용하면 CloudTrail 가 Amazon S3 버킷에 로그 파일을 전달할 수 있습니다. 를 사용하여 생성된 모든 추적 AWS Management Console 은 다중 리전입니다. AWS CLI를 사용하여 단일 리전 또는 다중 리전 추적을 생성할 수 있습니다. 계정 AWS 리전 의 모든 에서 활동을 캡처하기 때문에 다중 리전 추적을 생성하는 것이 좋습니다. 단일 리전 추적을 생성하는 경우 추적의 AWS 리전에 로깅된 이벤트만 볼 수 있습니다. 추적에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 Creating a trail for your AWS 계정 및 Creating a trail for an organization을 참조하세요.

추적을 생성하여 진행 중인 관리 이벤트의 사본 하나를 에서 Amazon S3 버킷 CloudTrail 에 무료로 전달할 수 있지만 Amazon S3 스토리지 요금이 있습니다. CloudTrail 요금에 대한 자세한 내용은 AWS CloudTrail 요금 섹션을 참조하세요. Amazon S3 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하세요.

의 Auto Scaling 관리 이벤트 CloudTrail

관리 이벤트는 의 리소스에서 수행되는 관리 작업에 대한 정보를 제공합니다 AWS 계정. 이를 제어 영역 작업이라고도 합니다. 기본적으로 는 관리 이벤트를 CloudTrail 기록합니다.

Amazon EC2 Auto Scaling은 모든 Auto Scaling 제어 영역 작업을 관리 이벤트로 기록합니다. Auto Scaling이 에 로그하는 Amazon EC2 Auto Scaling 제어 영역 작업 목록은 Amazon EC2 Auto Scaling API 참조를 CloudTrail참조하세요. Auto Scaling

Auto Scaling 이벤트 예제

이벤트는 모든 소스의 단일 요청을 나타내며 요청된 API 작업, 작업 날짜 및 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문된 스택 추적이 아니므로 이벤트는 특정 순서로 표시되지 않습니다.

다음 예제는 `CreateLaunchConfiguration` 작업을 보여주는 CloudTrail 이벤트를 보여줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Root",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
```

```
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-08-21T17:05:42Z"
            }
        }
    },
    "eventTime": "2018-08-21T17:07:49Z",
    "eventSource": "autoscaling.amazonaws.com",
    "eventName": "CreateLaunchConfiguration",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "Coral/Jakarta",
    "requestParameters": {
        "ebsOptimized": false,
        "instanceMonitoring": {
            "enabled": false
        },
        "instanceType": "t2.micro",
        "keyName": "EC2-key-pair-oregon",
        "blockDeviceMappings": [
            {
                "deviceName": "/dev/xvda",
                "ebs": {
                    "deleteOnTermination": true,
                    "volumeSize": 8,
                    "snapshotId": "snap-01676e0a2c3c7de9e",
                    "volumeType": "gp2"
                }
            }
        ],
        "launchConfigurationName": "launch_configuration_1",
        "imageId": "ami-6cd6f714d79675a5",
        "securityGroups": [
            "sg-00c429965fd921483"
        ]
    },
    "responseElements": null,
    "requestID": "0737e2ea-fb2d-11e3-bfd8-99133058e7bb",
    "eventID": "3fcfb182-98f8-4744-bd45-b38835ab61cb",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

CloudTrail 레코드 내용에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 레코드 내용을 참조하세요.

아마존 EC2 오토 스케일링에 대한 아마존 SNS 알림 옵션

애플리케이션에 영향을 미치는 중요한 이벤트를 알리도록 Auto Scaling 그룹을 구성할 수 있습니다. 알림을 사용하면 폴링을 제거할 수도 있으며 폴링으로 인해 가끔 발생하는 `RequestLimitExceeded` 오류가 발생하지 않습니다.

Amazon EC2 Auto Scaling에 대한 알림을 수신하는 방법은 두 가지가 있습니다.

- Amazon 단순 알림 서비스 — Amazon은 Auto Scaling 그룹이 인스턴스를 시작하거나 종료할 때 사용자에게 알릴 SNS 수 있습니다. Amazon SNS 알림만 켜거나 끌 수 있습니다. 자세한 내용은 SNS 아마존과 아마존 EC2 오토 스케일링 단원을 참조하십시오.
- Amazon EventBridge — 지정된 기준에 부합하고 Amazon을 비롯한 다양한 대상으로 전송되는 고급 이벤트 기반 알림을 EventBridge 제공합니다. SNS EventBridge 또한 보다 정확한 모니터링을 위해 더 넓은 범위의 Auto Scaling 이벤트를 모니터링할 수 있습니다. 자세한 내용은 Auto Scaling 이벤트를 처리하는 EventBridge 데 사용합니다. 단원을 참조하십시오.

선택적으로 수명 주기 후크가 있는 알림을 사용하여 시작 또는 종료 중에 인스턴스에 대한 사용자 지정 작업을 수행할 수 있습니다. 라이프사이클 후크와 함께 사용하도록 알림을 구성하는 방법에 대한 자세한 내용은 을 참조하십시오Amazon EC2 Auto Scaling 수명 주기 후크.

SNS아마존과 아마존 EC2 오토 스케일링

이 섹션에서는 SNS Amazon을 사용하여 Auto Scaling 그룹이 인스턴스를 시작하거나 종료하는 시기를 모니터링하는 방법을 보여줍니다.

예컨대, `autoscaling: EC2_INSTANCE_TERMINATE` 알림 유형을 사용하도록 Auto Scaling 그룹을 구성하면 Auto Scaling 그룹에서 인스턴스를 해지하고 이메일 알림을 보냅니다. 이 이메일에는 해지된 인스턴스의 세부 정보(예: 인스턴스 ID 및 인스턴스 해지 사유)가 포함됩니다.

Amazon EC2 Auto Scaling이 그룹에서 인스턴스를 추가하거나 제거하면 이러한 변경에 대한 알림이 인스턴스당 하나의 알림과 함께 사용자에게 전송된다는 점에 유의하십시오. 하지만 이러한 알림은 최선을 다해 전송하므로 초기 알림 이후에도 인스턴스가 실패할 수 있습니다 (예: 나중에 상태 확인이 실패하는 경우). 상태 확인 프로세스에 대한 자세한 내용은 을 참조하십시오Auto Scaling 그룹의 인스턴스에 대한 상태 확인.

SNS일반적으로 Amazon에 대한 자세한 내용은 Amazon 단순 알림 서비스 개발자 안내서를 참조하십시오.

내용

- SNS알림
- Amazon EC2 Auto Scaling을 위한 아마존 SNS 알림 구성
  - 아마존 SNS 주제 만들기
  - Amazon SNS 주제 구독하기
  - 아마존 SNS 구독 확인하기
  - 알림을 보내도록 Auto Scaling 그룹 구성
  - 알림 테스트
  - 알림 구성 삭제
- 암호화된 Amazon SNS 주제에 대한 주요 정책

SNS알림

Amazon EC2 Auto Scaling은 다음과 같은 이벤트 발생 시 아마존 SNS 알림 전송을 지원합니다.

이벤트	설명
`autoscaling:EC2_INSTANCE_LAUNCH`	인스턴스 시작 성공
`autoscaling:EC2_INSTANCE_LAUNCH_ERROR`	인스턴스 시작 실패
`autoscaling:EC2_INSTANCE_TERMINATE`	인스턴스 해지 성공
`autoscaling:EC2_INSTANCE_TERMINATE_ERROR`	인스턴스 해지 실패

메시지에 포함되는 정보는 다음과 같습니다.

- `Event` — 이벤트입니다.
- `AccountId` — Amazon Web Services 계정 ID입니다.

- `AutoScalingGroupName` — Auto Scaling 그룹의 이름입니다.
- `AutoScalingGroupARN`— Auto Scaling 그룹의. ARN
- `EC2InstanceId`— EC2 인스턴스의 ID.

예:

```
Service: AWS Auto Scaling
Time: 2016-09-30T19:00:36.414Z
RequestId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Event: autoscaling:EC2_INSTANCE_LAUNCH
AccountId: 123456789012
AutoScalingGroupName: my-asg
AutoScalingGroupARN: arn:aws:autoscaling:region:123456789012:autoScalingGroup...
ActivityId: 4e6156f4-a9e2-4bda-a7fd-33f2ae528958
Description: Launching a new EC2 instance: i-0598c7d356eba48d7
Cause: At 2016-09-30T18:59:38Z a user request update of AutoScalingGroup constraints
 to ...
StartTime: 2016-09-30T19:00:04.445Z
EndTime: 2016-09-30T19:00:36.414Z
StatusCode: InProgress
StatusMessage:
Progress: 50
EC2InstanceId: i-0598c7d356eba48d7
Details: {"Subnet ID":"subnet-id","Availability Zone":"zone"}
Origin: AutoScalingGroup
Destination: EC2
```

Amazon EC2 Auto Scaling을 위한 아마존 SNS 알림 구성

SNSAmazon을 사용하여 이메일 알림을 보내려면 먼저 주제를 생성한 다음 이메일 주소를 구독해야 합니다.

아마존 SNS 주제 만들기

SNS주제는 Auto Scaling 그룹이 알림을 보내는 데 사용하는 통신 채널인 논리적 액세스 포인트입니다. 주제의 이름을 지정하여 주제를 생성합니다.

주제 이름을 생성할 때 이름은 다음 요건을 충족해야 합니다.

- 1~256자 이내로 생성합니다.
- 대문자 및 ASCII 소문자, 숫자, 밑줄 또는 하이픈을 포함해야 합니다.

자세한 내용은 Amazon 단순 알림 서비스 개발자 안내서의 Amazon SNS 주제 생성을 참조하십시오.

Amazon SNS 주제 구독하기

Auto Scaling 그룹에서 주제로 전송하는 알림을 받으려면 엔드포인트가 해당 주제를 구독해야 합니다. 이 절차에서 엔드포인트에 대해 Amazon EC2 Auto Scaling으로부터 알림을 받을 이메일 주소를 지정합니다.

자세한 내용은 Amazon 단순 알림 서비스 개발자 안내서의 Amazon SNS 주제 구독을 참조하십시오.

아마존 SNS 구독 확인하기

Amazon은 이전 단계에서 지정한 이메일 주소로 확인 이메일을 SNS 보냅니다.

다음 단계로 넘어가기 전에 AWS 알림의 이메일을 열어 구독 확인 링크를 선택합니다.

에서 승인 메시지를 받게 됩니다. AWS SNSAmazon은 이제 알림을 수신하고 지정한 이메일 주소로 알림을 이메일로 보내도록 구성되었습니다.

알림을 보내도록 Auto Scaling 그룹 구성

인스턴스 시작 또는 인스턴스 종료와 같은 조정 이벤트가 발생할 SNS 때 Amazon에 알림을 보내도록 Auto Scaling 그룹을 구성할 수 있습니다. Amazon은 지정한 이메일 주소로 인스턴스에 대한 정보가 포함된 알림을 SNS 보냅니다.

Auto Scaling 그룹에 대한 Amazon SNS 알림을 구성하려면 (콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 아래쪽에 분할 창이 열리고 선택한 그룹에 대한 정보가 표시됩니다.
3. Activity(활동) 탭에서 Activity notifications(활동 알림), Create notification(알림 생성)을 선택합니다.
4. Create notifications(알림 생성) 창에서 다음을 수행하십시오:

   a. SNS주제에서는 SNS 주제를 선택합니다.

   b. Event types(이벤트 유형)에서 알림을 보낼 이벤트를 선택합니다.

   c. 생성(Create)을 선택합니다.

Auto Scaling 그룹에 대한 Amazon SNS 알림을 구성하려면 (AWS CLI)

다음 put-notification-configuration 명령을 사용합니다.

```
aws autoscaling put-notification-configuration --auto-scaling-group-name my-
asg --topic-arn arn --notification-types "autoscaling:EC2_INSTANCE_LAUNCH"
 "autoscaling:EC2_INSTANCE_TERMINATE"
```

알림 테스트

시작 이벤트에 대해 알림이 발생하도록 하려면 Auto Scaling 그룹의 용량을 1만큼 늘려 Auto Scaling 그룹을 업데이트합니다. 인스턴스 시작 후 몇 분 내에 알림을 받게 됩니다.

원하는 용량을 변경하려면(콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 아래쪽에 분할 창이 열리고 선택한 그룹에 대한 정보가 표시됩니다.
3. 세부 정보(Details) 탭에서 그룹 세부 정보(Group details), 편집(Edit)을 선택합니다.
4. 원하는 용량에 대해 현재 값을 1씩 늘립니다. 이 값이 최대 용량을 초과하는 경우, 최대 용량 값도 1씩 늘려야 합니다.
5. Update(업데이트)를 선택합니다.
6. 몇 분 후 이벤트 알림이 전송됩니다. 이 테스트에서 시작한 인스턴스가 추가로 필요하지 않으면 원하는 용량을 1만큼 줄일 수 있습니다. 몇 분 후 이벤트 알림이 전송됩니다.

알림 구성 삭제

Amazon EC2 Auto Scaling 알림 구성을 더 이상 사용하지 않는 경우 삭제할 수 있습니다.

Amazon EC2 Auto Scaling 알림 구성을 삭제하려면 (콘솔)

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹을 선택합니다.
3. Activity(활동) 탭에서 삭제할 알림 옆의 확인란을 선택한 다음 Actions(작업),Delete(삭제)를 차례로 선택합니다.

Amazon EC2 Auto Scaling 알림 구성을 삭제하려면 (AWS CLI)

다음 delete-notification-configuration 명령을 사용합니다.

```
aws autoscaling delete-notification-configuration --auto-scaling-group-name my-asg --
topic-arn arn
```

Amazon SNS 주제 및 Auto Scaling 그룹과 관련된 모든 구독을 삭제하는 방법에 대한 자세한 내용은 Amazon Simple 알림 서비스 개발자 안내서의 Amazon SNS 구독 및 주제 삭제를 참조하십시오.

암호화된 Amazon SNS 주제에 대한 주요 정책

지정한 Amazon SNS 주제는 를 사용하여 생성한 고객 관리 키로 암호화될 수 AWS Key Management Service있습니다. Amazon EC2 Auto Scaling에서 암호화된 주제에 게시할 권한을 부여하려면 먼저 KMS 키를 생성한 다음 키 정책에 다음 설명을 추가해야 합니다. KMS 예제를 키에 ARN 대한 액세스가 ARN 허용된 적절한 서비스 연결 역할의 것으로 대체하십시오. 자세한 내용은 Amazon Simple Storage Service 개발자 안내서의 AWS KMS 권한 구성을 참조하십시오.

이 예제에서 정책 설명은 서비스 연결 역할에 고객 관리 키를 사용할 수 있는 AWSServiceRoleForAutoScaling권한이라는 이름을 부여합니다. Amazon EC2 Auto Scaling 서비스 연결 역할에 대한 자세한 내용은 을 참조하십시오. Amazon EC2 Auto Scaling의 서비스 연결 역할

```
{
  "Sid": "Allow service-linked role use of the customer managed key",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*"
}
```

Amazon EC2 Auto Scaling에서 암호화된 주제에 게시할 수 있도록 허용하는 키 정책에서는 `aws:SourceArn` 및 `aws:SourceAccount` 조건 키가 지원되지 않습니다.

AWS Amazon EC2 Auto Scaling과 통합된 서비스

Amazon EC2 Auto Scaling은 다른 AWS 서비스와 통합될 수 있습니다. 각 서비스가 Amazon EC2 Auto Scaling에서 작동하는 방식에 대해 자세히 알아보려면 다음 통합 옵션을 검토하세요.

주제

- 용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리
- 용량 예약을 통해 특정 가용 영역의 용량을 예약할 수 있습니다.
- 명령줄에서 다음을 사용하여 Auto Scaling 그룹을 생성합니다. AWS CloudShell
- AWS CloudFormation을 이용한 Auto Scaling 그룹 생성
- 다음을 통해 인스턴스 유형 권장 사항을 확인하세요. AWS Compute Optimizer
- Elastic Load Balancing을 사용하여 Auto Scaling 그룹에서 들어오는 애플리케이션 트래픽을 분산하십시오.
- VPCLattice 대상 그룹을 통한 트래픽 흐름 관리
- Auto Scaling 이벤트를 처리하는 EventBridge 데 사용합니다.
- Amazon VPC를 사용하여 Auto Scaling 인스턴스에 네트워크 연결 제공

용량 재조정을 사용하여 Amazon EC2 스팟 중단 처리

스팟 인스턴스의 가용성에 영향을 주는 변경 사항을 모니터링하고 이에 자동으로 대응하도록 Amazon EC2 Auto Scaling을 구성할 수 있습니다. 용량 리밸런싱을 사용하면 실행 중인 인스턴스가 Amazon EC2에 의해 중단되기 전에 미리 새 스팟 인스턴스로 플릿을 보강할 수 있으므로 워크로드 가용성을 유지하는 데 도움이 됩니다.

용량 재조정의 목표는 중단 없이 워크로드를 계속 처리하는 것입니다. 스팟 인스턴스가 중단될 위험이 높아지면 Amazon EC2 스팟 서비스는 Amazon EC2 Auto Scaling에 EC2 인스턴스 재조정 권장 사항을 알립니다.

Auto Scaling 그룹에 대해 용량 재조정을 활성화하면 Amazon EC2 Auto Scaling은 재조정 권장 사항을 받은 그룹의 스팟 인스턴스를 사전에 교체하려고 합니다. 이는 중단 위험이 높지 않은 새로운 스팟 인스턴스로 워크로드를 재조정할 수 있는 기회를 제공합니다. 기존 인스턴스가 중단되기 전에 Amazon EC2 Auto Scaling이 새 스팟 인스턴스를 시작하는 동안 워크로드는 계속 작업을 처리할 수 있습니다.

용량 재조정을 사용하지 않는 경우, Amazon EC2 스팟 서비스가 인스턴스를 중단하고 건전성 체크에 실패할 때까지 Amazon EC2 Auto Scaling은 스팟 인스턴스를 교체하지 않습니다. 인스턴스를 중단하

기 전에 Amazon EC2는 항상 EC2 인스턴스 재조정 권장 사항과 스팟 인스턴스 중단 2분 알림을 둘 다 제공합니다.

내용

- 개요
- 용량 재조정 동작
- 고려 사항
- OR를 사용하여 용량 재조정을 활성화합니다. AWS Management ConsoleAWS CLI

개요

Auto Scaling 그룹과 함께 용량 재조정을 사용하려면 다음과 같은 기본 단계를 따르세요.

1. 여러 인스턴스 유형 및 가용 영역을 사용하도록 Auto Scaling 그룹을 구성합니다. 이러한 방식으로 Amazon EC2 Auto Scaling은 각 가용 영역에서 스팟 인스턴스를 위해 사용 가능한 용량을 확인할 수 있습니다. 자세한 설명은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.
2. 필요에 따라 수명 주기 후크를 추가하여 재조정 알림을 수신하는 인스턴스 내에서 애플리케이션을 정상적으로 해지할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 수명 주기 후크 섹션을 참조하세요.

   다음은 수명 주기 후크를 사용할 수 있는 몇 가지 이유입니다:

   - Amazon SQS 작업자를 정상적으로 해지하는 경우
   - 도메인 이름 시스템(DNS)에서 등록 취소를 완료하려면
   - 시스템 또는 애플리케이션 로그를 당겨 Amazon Simple Storage Service(Amazon S3)에 업로드하려면
3. 수명 주기 후크에 대한 사용자 지정 작업을 개발합니다. 사용자 지정 작업을 최대한 빨리 호출하려면 인스턴스가 해지될 준비가 된 시점을 알아야 합니다. 인스턴스의 수명 주기 상태를 감지하여 이를 알아보세요.

   - 인스턴스 외부에서 작업을 호출하려면 규칙을 작성하고 이벤트 패턴이 EventBridge 규칙과 일치할 때 수행할 작업을 자동화하십시오.
   - 인스턴스 내에서 작업을 호출하려면 해지 스크립트를 실행하고 인스턴스 메타데이터를 통해 수명 주기 상태를 검색하도록 인스턴스를 구성하세요.

2분 이내에 완료되도록 사용자 지정 작업을 설계하는 것이 중요합니다. 이렇게 하면 인스턴스가 해지되기 전에 작업을 완료할 수 있는 충분한 시간이 확보됩니다.

이 단계를 완료하면 용량 재조정 사용을 시작할 수 있습니다.

용량 재조정 동작

용량 재조정을 사용하면 인스턴스가 재조정 권장 사항을 수신할 때 Amazon EC2 Auto Scaling은 다음과 같은 방식으로 작동합니다.

- 새 스팟 인스턴스가 시작되면 Amazon EC2 Auto Scaling은 새 인스턴스가 건전성 체크를 통과할 때까지 기다렸다가 이전 인스턴스를 해지합니다. 둘 이상의 인스턴스를 교체하는 경우, 새 인스턴스가 시작되어 건전성 체크를 통과하면 각각의 이전 인스턴스가 해지되기 시작합니다.
- Amazon EC2 Auto Scaling에서는 이전 인스턴스 해지 전에 새 인스턴스를 시작하려고 하므로 지정된 최대 용량에 도달하거나 이에 근접하면 재조정 활동을 지연시키거나 완전히 중지할 수 있습니다. 이 문제를 방지하기 위해 Amazon EC2 Auto Scaling은 일시적으로 그룹의 최대 크기를 원하는 용량의 최대 10%까지 초과할 수 있습니다.
- Auto Scaling 그룹에 수명 주기 후크를 추가하지 않은 경우, Amazon EC2 Auto Scaling은 새 인스턴스가 건전성 체크를 통과하자마자 이전 인스턴스를 해지하기 시작합니다.
- 수명 주기 후크를 추가한 경우, 이전 인스턴스 해지를 시작하기까지 걸리는 시간이 수명 주기 후크에 지정한 시간 초과 값만큼 연장됩니다.
- 스케일링 정책 또는 예약된 스케일링을 사용하는 경우, 스케일링 활동이 병렬로 실행됩니다. 크기 조정 활동이 진행 중이고 Auto Scaling 그룹이 원하는 새 용량 미만일 경우, Amazon EC2 Auto Scaling은 이전 인스턴스를 해지하기 전에 먼저 스케일 아웃합니다.

한 가용 영역에 인스턴스 유형에 대한 용량이 없는 경우, Amazon EC2 Auto Scaling은 성공할 때까지 활성화된 다른 가용 영역에서 스팟 인스턴스를 계속 시작하려고 시도합니다.

더 나쁜 경우의 시나리오에서 또는 새 인스턴스가 시작에 실패했거나 건전성 체크에 불합격하는 경우, Amazon EC2 Auto Scaling은 계속해서 인스턴스를 다시 시작시키려고 합니다. 새 인스턴스를 시작시키려고 시도하는 동안, 이전 인스턴스는 결국 중단되고 강제로 해지될 것이며 장애 2분전 고지가 이루어집니다.

고려 사항

용량 재조정 사용시 다음 사항을 고려하십시오:

스팟 중단에 내성을 갖도록 애플리케이션을 설계하세요.

귀하의 애플리케이션은 인스턴스 수의 동적 변경과 스팟 인스턴스가 조기에 중단될 가능성에 대처할 수 있어야 합니다. 예컨대, Auto Scaling 그룹이 Elastic Load Balancing 로드 밸런서 뒤에 있는 경우, Amazon EC2 Auto Scaling은 인스턴스가 로드 밸런서에서 등록을 취소할 때까지 기다렸다가 수명 주기 후크를 호출합니다. 인스턴스 등록을 취소하고 수명 주기 작업을 완료하는 데 시간이 너무 오래 걸리면 인스턴스를 해지하기 전에 Amazon EC2 Auto Scaling이 수명 주기 작업 완료를 기다리는 동안 인스턴스가 중단될 수도 있습니다.

Amazon EC2에서 항상 스팟 인스턴스 장애 2분전 고지보다 먼저 리밸런싱 권고 신호를 전송할 수 있는 것은 아닙니다. 때때로, 리밸런싱 권고 신호가 장애 2분전 고지와 동시에 도착합니다. 이런 일이 발생하면 Amazon EC2 Auto Scaling은 수명 주기 후크를 호출하고 즉시 새 스팟 인스턴스를 시작하려고 시도합니다.

교체 스팟 인스턴스의 중단 위험 증가 방지

`lowest-price` 할당 전략을 사용하는 경우, 교체 스팟 인스턴스가 중단될 위험이 높아질 수도 있습니다. 이는 시작 직후에 교체 스팟 인스턴스가 중단될 수 있더라도 해당 순간에 사용 가능한 용량이 있는 최저가의 풀에서 인스턴스를 항상 시작시키기 때문입니다. 서비스 중단 위험이 높아지는 것을 방지하려면 `lowest-price` 할당 전략을 사용하지 않는 것이 좋습니다. 대신 `price-capacity-optimized` 할당 전략을 사용하는 것이 좋습니다. 이 전략은 중단될 가능성이 가장 낮고 가격이 가장 낮은 스팟 풀에서 교체 스팟 인스턴스를 시작합니다. 따라서 가까운 장래에 중단될 가능성이 적습니다.

Amazon EC2 Auto Scaling은 가용성이 동일하거나 더 나은 경우에만 새 인스턴스를 시작함

용량 리밸런싱의 목표 중 하나는 스팟 인스턴스의 가용성을 개선하는 것입니다. 기존 스팟 인스턴스에 대한 리밸런싱 권장 사항이 있는 경우, Amazon EC2 Auto Scaling은 새 인스턴스가 기존 인스턴스와 동일하거나 더 나은 가용성을 제공하는 경우에만 새 인스턴스를 시작합니다. 새 인스턴스의 중단 위험이 기존 인스턴스보다 더 높은 경우, Amazon EC2 Auto Scaling은 새 인스턴스를 시작하지 않습니다. 하지만 Amazon EC2 Auto Scaling은 Amazon EC2 스팟 서비스에서 제공하는 정보에 근거하여 스팟 용량 풀을 계속 평가하여 가용성이 향상되면 새 인스턴스를 시작합니다.

Amazon EC2 Auto Scaling이 사전에 새 인스턴스를 시작하지 않은 채로 기존 인스턴스가 중단될 수 있습니다. 이 경우, Amazon EC2 Auto Scaling은 스팟 인스턴스 중단 알림을 받는 즉시 새 인스턴스를 시작하려고 시도합니다. 이는 새 인스턴스의 중단 위험이 높은지 여부와 관계없이 발생합니다.

용량 리밸런싱은 스팟 인스턴스의 간섭 속도를 증가시키지 않음

용량 리밸런싱을 활성화하면 스팟 인스턴스 중단 속도(Amazon EC2가 용량을 회수해야 할 때 회수되는 스팟 인스턴스의 수)가 증가하지 않습니다. 그러나 용량 리밸런싱에서 중단 위험이 있는 인스

턴스를 탐지할 경우, Amazon EC2 Auto Scaling이 즉시 새로운 인스턴스를 시작하려고 시도합니다. 그러므로 위험에 처한 인스턴스가 간섭된 후 Amazon EC2 Auto Scaling이 시작되기를 기다리면 새 인스턴스를 시작시키는 것보다 더 많은 인스턴스가 교체될 수 있습니다.

용량 리밸런싱을 활성화하면 더 많은 인스턴스를 교체하게 될 수도 있지만, 미리 대비할 수 있어 도움이 됩니다. 이렇게 하면 인스턴스가 중단되기 전에 작업을 수행할 수 있는 시간이 더 늘어납니다. 스팟 인스턴스 중단 알림이 오면 일반적으로 최대 2분 이내에 인스턴스를 적절히 해지해야 합니다. 용량 재조정을 통해 새 인스턴스를 미리 시작하면 위험한 인스턴스에서 기존 프로세스가 완료될 확률을 높일 수 있습니다. 또한 인스턴스 해지 절차를 시작하고, 위험 인스턴스에서 새 작업이 예약되지 않도록 하며, 새로 시작된 인스턴스가 애플리케이션을 인수할 수 있도록 준비할 수 있습니다. 용량 리밸런싱에서 전향적으로 인스턴스를 교체하기 때문에 적절한 연속성을 누릴 수 있습니다.

용량 리밸런싱을 사용할 때의 위험과 장점을 보여주는 이론적 예:

- 오후 2:00 – 인스턴스 A에 대한 리밸런싱 권고를 받음. Amazon EC2 Auto Scaling이 즉시 교체 인스턴스 B를 시작시키려고 시도하므로 귀하는 해지 절차를 시작할 시간이 마련됩니다.
- 오후 2:30 – 인스턴스 B에 대한 리밸런싱 권고를 받고 인스턴스 C로 교체하므로 해지 절차를 시작할 시간이 마련됩니다.
- 오후 2:32 – 용량 리밸런싱을 활성화하지 않은 경우와 인스턴스 A에 대한 스팟 인스턴스 장애 알림을 오후 2시 32분에 받지 않은 경우, 조치를 취할 시간이 2분에 불과할 것입니다. 하지만 인스턴스 A는 이때까지 계속 실행되었을 것입니다.

OR를 사용하여 용량 재조정을 활성화합니다. AWS Management ConsoleAWS CLI

AWS Management Console 또는 AWS CLI 를 사용하여 Auto Scaling 그룹의 용량 재조정을 활성화할 수 있습니다. Amazon EC2 Auto Scaling은 재조정 권고를 받은 그룹 내 스팟 인스턴스를 사전에 교체하려고 합니다.

용량 재조정 활성화(콘솔)

Auto Scaling 그룹을 생성하거나 업데이트할 때 용량 재조정을 사용하거나 사용하지 않도록 설정할 수 있습니다.

새 Auto Scaling 그룹에 대해 용량 재조정 활성화

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택합니다.

2. Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.
3. 1단계: 시작 템플릿 또는 구성을 선택을 위해, Auto Scaling 그룹의 명칭을 입력하고 시작 템플릿을 선택한 후 다음을 선택하여 그 다음 단계로 진행합니다.
4. 2단계: 인스턴스 시작 옵션 선택의 인스턴스 유형 요건에서 설정을 선택하여 혼합 인스턴스 그룹을 만듭니다. 여기에는 시작할 수 있는 인스턴스 유형, 인스턴스 구매 옵션, 스팟 및 온디맨드 인스턴스에 대한 할당 전략이 포함됩니다. 기본적으로 이러한 설정은 구성되어 있지 않습니다. 이러한 설정을 구성하려면 Override launch template(시작 템플릿 재정의)를 선택해야 합니다. 혼합 인스턴스 그룹 생성에 대한 자세한 설명은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.
5. 네트워크에서 원하는 옵션을 선택합니다. 사용할 서브넷이 다른 가용 영역에 있는지 확인합니다.
6. 할당 전략 섹션에서 스팟 할당 전략을 선택합니다. 용량 재조정을 선택하거나 확인란을 선택 취소하여 용량 재조정을 활성화하거나 비활성화합니다. 인스턴스 구매 옵션 섹션에서 스팟 인스턴스로 띄울 Auto Scaling 그룹의 백분율을 요청한 경우에만 이 옵션이 표시됩니다.
7. Auto Scaling 그룹을 생성합니다.
8. (옵션) 필요에 따라 수명 주기 후크를 추가합니다. 자세한 설명은 Auto Scaling 그룹에 라이프사이클 후크 추가 섹션을 참조하세요.

기존 Auto Scaling 그룹에 대해 용량 재조정을 활성화 또는 비활성화하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다. 페이지 하단에 분할 창이 열립니다.
3. Details(세부 정보) 탭에서 Allocation strategies(할당 전략), Edit(편집)을 선택합니다.
4. 할당 전략 섹션에서 용량 재조정 아래의 확인란을 선택하거나 선택 취소하여 용량 재조정을 사용하거나 사용하지 않도록 설정합니다.
5. 업데이트(Update)를 선택합니다.

용량 재조정 활성화(AWS CLI)

다음 예는 를 사용하여 용량 재조정을 활성화 및 AWS CLI 비활성화하는 방법을 보여줍니다.

create-auto-scaling-groupor update-auto-scaling-group명령을 다음 파라미터와 함께 사용합니다.

- `--capacity-rebalance`/`--no-capacity-rebalance`— 용량 재조정 활성화 여부를 나타내는 부울 값입니다.

create-auto-scaling-group명령을 호출하기 전에 Auto Scaling 그룹에서 사용하도록 구성된 시작 템플릿의 이름이 필요합니다. 자세한 정보는 Auto Scaling 그룹에 대한 시작 템플릿 생성을 참조하세요.

> **Note**
>
> 다음 절차는 JSON 또는 YAML로 형식의 구성 파일을 사용하는 방법을 보여줍니다. AWS CLI 버전 1을 사용하는 경우 JSON 형식의 구성 파일을 지정해야 합니다. AWS CLI 버전 2를 사용하는 경우 YAML 또는 JSON 형식의 구성 파일을 지정할 수 있습니다.

JSON

새 Auto Scaling 그룹 생성 및 구성

- 다음 create-auto-scaling-group명령을 사용하여 새 Auto Scaling 그룹을 생성하고 용량 재조정을 활성화합니다. 이 명령은 JSON 파일을 Auto Scaling 그룹의 유일한 파라미터로 참조합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-json file://~/config.json
```

혼합 인스턴스 정책을 지정하는 CLI 구성 파일이 아직 없는 경우에서 하나 생성합니다.

구성 파일의 최상위 JSON 객체에 다음 줄을 추가합니다.

```
{
    "CapacityRebalance": true
}
```

다음은 예 `config.json` 파일입니다.

```
{
    "AutoScalingGroupName": "my-asg",
    "DesiredCapacity": 12,
    "MinSize": 12,
    "MaxSize": 15,
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "InstancesDistribution": {
            "OnDemandBaseCapacity": 0,
            "OnDemandPercentageAboveBaseCapacity": 25,
            "SpotAllocationStrategy": "price-capacity-optimized"
```

```
        },
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "$Default"
            },
            "Overrides": [
                {
                    "InstanceType": "c5.large"
                },
                {
                    "InstanceType": "c5a.large"
                },
                {
                    "InstanceType": "m5.large"
                },
                {
                    "InstanceType": "m5a.large"
                },
                {
                    "InstanceType": "c4.large"
                },
                {
                    "InstanceType": "m4.large"
                },
                {
                    "InstanceType": "c3.large"
                },
                {
                    "InstanceType": "m3.large"
                }
            ]
        }
    },
    "TargetGroupARNs": "arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-alb-target-group/943f017f100becff",
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

YAML

새 Auto Scaling 그룹 생성 및 구성

- 다음 create-auto-scaling-group명령을 사용하여 새 Auto Scaling 그룹을 생성하고 용량 재조정을 활성화합니다. 이 명령은 YAML 파일을 Auto Scaling 그룹의 유일한 파라미터로 참조합니다.

```
aws autoscaling create-auto-scaling-group --cli-input-yaml file://~/config.yaml
```

YAML로 형식이 지정된 구성 파일에 다음 줄을 추가합니다.

```
CapacityRebalance: true
```

다음은 예 `config.yaml` 파일입니다.

```
---
AutoScalingGroupName: my-asg
DesiredCapacity: 12
MinSize: 12
MaxSize: 15
CapacityRebalance: true
MixedInstancesPolicy:
  InstancesDistribution:
    OnDemandBaseCapacity: 0
    OnDemandPercentageAboveBaseCapacity: 25
    SpotAllocationStrategy: price-capacity-optimized
  LaunchTemplate:
    LaunchTemplateSpecification:
      LaunchTemplateName: my-launch-template
      Version: $Default
    Overrides:
    - InstanceType: c5.large
    - InstanceType: c5a.large
    - InstanceType: m5.large
    - InstanceType: m5a.large
    - InstanceType: c4.large
    - InstanceType: m4.large
    - InstanceType: c3.large
    - InstanceType: m3.large
TargetGroupARNs:
- arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-alb-target-
group/943f017f100becff
```

```
VPCZoneIdentifier: subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782
```

기존 Auto Scaling 그룹에 대해 용량 재조정 활성화

- 다음 update-auto-scaling-group명령을 사용하여 용량 재조정을 활성화합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --capacity-rebalance
```

기존 Auto Scaling 그룹에 대해 용량 재조정가 활성화되었는지 확인

- 다음 describe-auto-scaling-groups명령을 사용하여 용량 재조정이 활성화되었는지 확인하고 세부 정보를 볼 수 있습니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

다음은 응답의 예입니다.

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn",
            ...
            "CapacityRebalance": true
        }
    ]
}
```

용량 재조정을 비활성화하려면

update-auto-scaling-group명령을 `--no-capacity-rebalance` 옵션과 함께 사용하여 용량 재조정을 비활성화합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --no-capacity-rebalance
```

관련 리소스

용량 재조정에 대한 자세한 내용은 Compute Blog의 EC2 Auto Scaling을 위한 새로운 용량 재조정 기능을 사용하여 스팟 인스턴스 수명 주기를 사전에 관리하기를 참조하십시오. AWS

EC2 인스턴스 재조정 권장 사항에 대한 자세한 내용은 Amazon EC2 사용 설명서의 EC2 인스턴스 재조정 권장 사항을 참조하십시오.

수명 주기 후크에 대한 자세한 설명은 다음 리소스를 참조하세요.

- 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성 EventBridge(사용)
- 자습서: 데이터 스크립트와 인스턴스 메타데이터를 사용하여 수명 주기 상태 검색

제한 사항

- Amazon EC2 Auto Scaling은 인스턴스가 스케일 인으로부터 보호되지 않는 경우에만 재조정 알림을 받는 인스턴스를 교체할 수 있습니다. 그러나 스케일 인 보호는 스팟 중단으로 인한 해지는 방지하지 못합니다. 자세한 설명은 인스턴스 확장 보호를 사용하여 인스턴스 종료를 제어할 수 있습니다. 섹션을 참조하세요.
- Amazon EC2 Auto Scaling을 사용할 수 있는 모든 상용 AWS 리전 에서 용량 리밸런싱 지원이 가능합니다(중동(UAE) 지역은 예외).

용량 예약을 통해 특정 가용 영역의 용량을 예약할 수 있습니다.

Amazon EC2 온디맨드 용량 예약을 사용하면 특정 가용 영역의 컴퓨팅 용량을 예약할 수 있습니다. 용량 예약 사용을 시작하려면 특정 가용 영역에서 용량 예약을 생성합니다. 그런 다음, 인스턴스를 예약 용량으로 시작하거나 용량 사용률을 실시간으로 확인할 수 있으며, 필요 시 용량을 늘리거나 줄일 수 있습니다.

용량 예약은 `open` 또는 `targeted`로 구성됩니다. 용량 예약이 `open`인 경우 일치하는 속성이 있는 새 인스턴스 및 기존 인스턴스는 용량 예약의 용량으로 자동 실행됩니다. 용량 예약이 `targeted`이면 인스턴스를 예약된 용량으로 실행하도록 지정해야 합니다.

이 항목에서는 `targeted` 용량 예약에 온디맨드 인스턴스를 시작하는 Auto Scaling 그룹을 생성하는 방법을 보여줍니다. 특정 용량 예약을 사용할 때 유용합니다.

기본 단계는 다음과 같습니다.

1. 인스턴스 유형, 플랫폼, 인스턴스 수가 동일한 여러 가용 영역에 용량 예약을 생성합니다.

2. AWS Resource Groups를 사용하여 용량 예약을 그룹화합니다.
3. 용량 예약과 동일한 가용 영역을 사용하여 리소스 그룹을 대상으로 하는 시작 템플릿으로 Auto Scaling 그룹을 생성합니다.

내용

- 1단계: 용량 예약 생성
- 2단계: 용량 예약 그룹 생성
- 3단계: 시작 템플릿 생성
- 4단계: Auto Scaling 그룹 생성
- 관련 리소스

1단계: 용량 예약 생성

첫 번째 단계는 Auto Scaling 그룹이 배포될 각 가용 영역에 용량 예약을 생성하는 것입니다.

> **Note**
> 용량 예약을 처음 생성할 때만 `targeted` 예약을 생성할 수 있습니다.

Console

용량 예약을 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 용량 예약를 선택한 후 용량 예약 생성을 선택합니다.
3. 용량 예약 생성 페이지의 인스턴스 세부 정보 섹션에서 다음 설정에 주의하세요. 시작하는 인스턴스 유형, 플랫폼, 인스턴스의 가용 영역은 여기서 지정한 인스턴스 유형, 플랫폼, 가용 영역과 일치해야 합니다. 그렇지 않으면 용량 예약이 적용되지 않습니다.
   a. 인스턴스 유형의 경우 예약된 용량으로 시작할 인스턴스 유형을 선택합니다.
   b. 플랫폼은 사용자 인스턴스에 사용할 운영 체제를 선택합니다.
   c. 가용 영역의 경우 용량을 예약하려는 첫 번째 가용 영역을 선택합니다.
   d. 총 용량은 필요한 인스턴스 수를 선택합니다. Auto Scaling 그룹에 필요한 총 인스턴스 수를 사용하려는 가용 영역 수로 나눈 값을 계산합니다.

4. 용량 예약 세부 정보에서, 용량 예약 종료를 위해, 다음 옵션 중 하나를 선택합니다.
   - 특정 시간 - 지정된 날짜 및 시간에 용량 예약을 자동으로 취소합니다.
   - 수동 — 명시적으로 취소할 때까지 용량을 예약합니다.
5. 인스턴스 자격에서 대상: 용량 예약을 대상으로 하는 인스턴스만을 선택합니다.
6. (선택 사항) 태그의 경우 용량 예약과 연결할 태그를 지정합니다.
7. 생성을 선택합니다.
8. 새로 생성한 용량 예약의 ID를 기록해 둡니다. 용량 예약 그룹을 설정하는 데 필요합니다.

Auto Scaling 그룹에 사용할 각 가용 영역에 대해 이 절차를 반복하고 가용 영역 옵션의 값만 변경합니다.

AWS CLI

용량 예약을 생성하려면

다음 create-capacity-reservation명령을 사용하여 용량 예약을 생성합니다. `--availability-zone`, `--instance-type`, `--instance-platform`, `--instance-count`의 샘플 값을 바꿉니다.

```
aws ec2 create-capacity-reservation \
    --availability-zone us-east-1a \
    --instance-type c5.xlarge \
    --instance-platform Linux/UNIX \
    --instance-count 3 \
    --instance-match-criteria targeted
```

결과 용량 예약 ID의 예

```
{
    "CapacityReservation": {
        "CapacityReservationId": "cr-1234567890abcdef1",
        "OwnerId": "123456789012",
        "CapacityReservationArn": "arn:aws:ec2:us-east-1:123456789012:capacity-
reservation/cr-1234567890abcdef1",
        "InstanceType": "c5.xlarge",
        "InstancePlatform": "Linux/UNIX",
        "AvailabilityZone": "us-east-1a",
        "Tenancy": "default",
        "TotalInstanceCount": 3,
```

```
        "AvailableInstanceCount": 3,
        "EbsOptimized": false,
        "EphemeralStorage": false,
        "State": "active",
        "StartDate": "2023-07-26T21:36:14+00:00",
        "EndDateType": "unlimited",
        "InstanceMatchCriteria": "targeted",
        "CreateDate": "2023-07-26T21:36:14+00:00"
    }
}
```

새로 생성한 용량 예약의 ID를 기록해 둡니다. 용량 예약 그룹을 설정하는 데 필요합니다.

Auto Scaling 그룹에 사용할 각 가용 영역에 대해 이 명령을 반복하고 `--availability-zone` 옵션의 값만 변경합니다.

2단계: 용량 예약 그룹 생성

용량 예약 생성이 끝나면 AWS Resource Groups 서비스를 사용하여 용량 예약을 함께 그룹화할 수 있습니다. AWS Resource Groups는 용도에 따라 여러 가지 다른 유형의 그룹을 지원합니다. Amazon EC2는 서비스 연결 리소스 그룹이라는 특수 목적 그룹을 사용하여 용량 예약 그룹을 대상으로 합니다. 이 서비스 연결 리소스 그룹과 상호 작용하려면 콘솔이 아닌 AWS CLI 또는 SDK를 사용할 수 있습니다. 서비스 연결 리소스 그룹에 대한 자세한 내용은 AWS Resource Groups 사용 설명서의 리소스 그룹에 대한 서비스 구성을 참조하세요.

를 사용하여 용량 예약 그룹을 만들려면 AWS CLI

create-group 명령을 사용하여 용량 예약만 포함할 수 있는 리소스 그룹을 생성할 수 있습니다. 이 예제에서 리소스 그룹의 이름은 *my-cr-group*입니다.

```
aws resource-groups create-group \
    --name my-cr-group \
    --configuration '{"Type":"AWS::EC2::CapacityReservationPool"}'
 '{"Type":"AWS::ResourceGroups::Generic", "Parameters": [{"Name": "allowed-resource-
types", "Values": ["AWS::EC2::CapacityReservation"]}]}'
```

다음은 응답의 예입니다.

```
{
    "Group": {
        "GroupArn": "arn:aws:resource-groups:us-east-1:123456789012:group/my-cr-group",
```

```
        "Name": "my-cr-group"
    },
    "GroupConfiguration": {
        "Configuration": [
            {
                "Type": "AWS::EC2::CapacityReservationPool"
            },
            {
                "Type": "AWS::ResourceGroups::Generic",
                "Parameters": [
                    {
                        "Name": "allowed-resource-types",
                        "Values": [
                            "AWS::EC2::CapacityReservation"
                        ]
                    }
                ]
            }
        ],
        "Status": "UPDATE_COMPLETE"
    }
}
```

리소스 그룹의 ARN을 기록해 두세요. Auto Scaling 그룹에 대한 시작 템플릿을 설정할 때 필요합니다.

AWS CLI를 사용하여 용량 예약을 새로 생성된 그룹에 연결하려면

다음 group-resources 명령을 사용하여 용량 예약을 새로 생성된 용량 예약 그룹에 연결합니다. `--resource-arns` 옵션의 경우, 해당 ARN을 사용하여 용량 예약을 지정합니다. 관련 리전, 사용자의 계정 ID, 이전 단계에서 기록한 예약 ID를 사용하여 ARN을 구성합니다. 이 예시에서는 ID *cr-1234567890abcdef1* 및 *cr-54321abcdef567890*를 포함한 예약이 *my-cr-group*으로 이름이 지정된 그룹에서 함께 그룹화됩니다.

```
aws resource-groups group-resources \
    --group my-cr-group \
    --resource-arns \
      arn:aws:ec2:region:account-id:capacity-reservation/cr-1234567890abcdef1 \
      arn:aws:ec2:region:account-id:capacity-reservation/cr-54321abcdef567890
```

다음은 응답의 예입니다.

```
{
```

```
    "Succeeded": [
    "arn:aws:ec2:us-east-1:123456789012:capacity-reservation/cr-1234567890abcdef1",
    "arn:aws:ec2:us-east-1:123456789012:capacity-reservation/cr-54321abcdef567890"
    ],
    "Failed": [],
    "Pending": []
}
```

리소스 그룹의 수정 또는 삭제에 대한 자세한 내용은 AWS Resource Groups API 참조를 참조하세요.

3단계: 시작 템플릿 생성

Console

시작 템플릿을 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창의 Instances(인스턴스)에서 Launch Templates(시작 템플릿)을 선택합니다.
3. Create launch template(시작 템플릿 생성)을 선택합니다. 시작 템플릿의 이름을 입력하고 초기 버전에 대한 설명을 제공하세요.
4. Auto Scaling guidance(Auto Scaling 지침)에서 확인란을 선택합니다.
5. 시작 템플릿을 생성합니다. 사용할 용량 예약과 일치하는 AMI 및 인스턴스 유형을 선택하고 선택 사항으로 키 페어, 보안 그룹 1개 이상, 인스턴스에 대한 추가 EBS 볼륨 또는 인스턴스 스토어 볼륨을 선택할 수 있습니다.
6. 고급 세부 정보를 열고 다음을 수행합니다.
   a. 용량 예약은 그룹별 대상을 선택합니다.
   b. 용량 예약 - 그룹별 대상은 이전 섹션에서 생성한 용량 예약 그룹을 선택한 다음 저장을 선택합니다.
7. Create launch template(시작 템플릿 생성)을 선택합니다.
8. 확인 페이지에서 Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

AWS CLI

시작 템플릿 생성

다음 create-launch-template명령을 사용하여 용량 예약이 특정 리소스 그룹을 대상으로 하도록 지정하는 시작 템플릿을 생성합니다. `--launch-template-name`의 샘플 값을 바꿉니다.

*c5.xlarge*를 용량 예약에서 사용한 인스턴스 유형으로 대체하고, *ami-0123456789EXAMPLE*을 사용하려는 AMI의 ID로 대체합니다. *arn:aws:resource-groups:region:account-id:group/my-cr-group*을 이전 섹션의 시작 부분에서 생성한 리소스 그룹의 ARN으로 대체합니다.

```
aws ec2 create-launch-template \
    --launch-template-name my-launch-template \
    --launch-template-data \
        '{"InstanceType": "c5.xlarge",
          "ImageId": "ami-0123456789EXAMPLE",
          "CapacityReservationSpecification":
            {"CapacityReservationTarget":
                { "CapacityReservationResourceGroupArn": "arn:aws:resource-
groups:region:account-id:group/my-cr-group" }
            }
        }'
```

다음은 응답의 예입니다.

```
{
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-0dd77bd41dEXAMPLE",
        "LaunchTemplateName": "my-launch-template",
        "CreateTime": "2023-07-26T21:42:48+00:00",
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "DefaultVersionNumber": 1,
        "LatestVersionNumber": 1
    }
}
```

4단계: Auto Scaling 그룹 생성

Console

평소와 같이 Auto Scaling 그룹을 생성하되, VPC 서브넷을 선택할 때는 각 가용 영역에서 생성한 `targeted` 용량 예약과 일치하는 서브넷을 선택합니다. Auto Scaling 그룹이 이러한 가용 영역 중 하나에서 온디맨드 인스턴스를 시작하면 해당 가용 영역에 예약된 용량으로 인스턴스가 실행됩니다. 원하는 용량이 충족되기 전에 리소스 그룹의 용량 예약이 소진되면 예약 용량을 초과한 모든 용량을 일반 온디맨드 용량으로 시작합니다.

단순한 Auto Scaling 그룹을 생성하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택합니다.
2. 화면 상단의 탐색 표시줄에서 시작 템플릿을 생성할 때 사용한 것과 동일한 AWS 리전 것을 선택합니다.
3. Auto Scaling 그룹 생성을 선택합니다.
4. 시작 템플릿 또는 구성 선택 페이지에서 Auto Scaling group name(Auto Scaling 그룹 명칭)에 Auto Scaling 그룹의 이름을 입력합니다.
5. 시작 템플릿에서 기존 시작 템플릿을 선택합니다.
6. Launch template version(시작 템플릿 버전)에서 Auto Scaling 그룹이 스케일 아웃 시 시작 템플릿의 기본 버전을 사용할지, 최신 버전을 사용할지, 아니면 특정 버전을 사용할지를 선택합니다.
7. 인스턴스 시작 옵션 선택 페이지에서 인스턴스 유형 요구 사항 섹션을 건너뛰고 시작 템플릿에 지정된 EC2 인스턴스 유형을 사용합니다.
8. 네트워크(Network)의 VPC에서 VPC를 선택합니다. Auto Scaling 그룹은 시작 템플릿에서 지정한 보안 그룹과 동일한 VPC에 생성되어야 합니다. 시작 템플릿에서 보안 그룹을 지정하지 않은 경우, 용량 예약과 동일한 가용 영역에 서브넷이 있는 모든 VPC를 선택할 수 있습니다.
9. 가용 영역 및 서브넷의 경우, 용량 예약이 있는 가용 영역에 따라 포함할 각 가용 영역의 서브넷을 선택합니다.
10. 다음을 두 번 선택합니다
11. 그룹 크기 및 조정 정책 구성 페이지에서 원하는 용량에 시작할 인스턴스의 초기 수를 입력합니다. 이 수를 최소 또는 최대 용량 제한을 벗어나는 값으로 변경하는 경우 최소 용량 또는 최대 용량 값을 업데이트해야 합니다. 자세한 내용은 Auto Scaling 그룹에 대한 스케일링 제한 설정(을)를 참조하세요.
12. Skip to review(검토로 건너뛰기)를 선택합니다.
13. 검토(Review) 페이지에서 Auto Scaling 그룹 생성(Create Auto Scaling group)을 선택합니다.

AWS CLI

단순한 Auto Scaling 그룹을 생성하려면

다음 create-auto-scaling-group명령을 사용하고 시작 템플릿의 이름과 버전을 `--launch-template` 옵션 값으로 지정합니다. `--auto-scaling-group-name`, `--min-size`, `--max-size`, `--vpc-zone-identifier`의 샘플 값을 바꿉니다.

`--availability-zones` 옵션에는 용량 예약을 생성한 가용 영역을 지정합니다. 예를 들어, 용량 예약에서 `us-east-1a` 및 `us-east-1b` 가용 영역을 지정하는 경우 동일한 영역에 Auto Scaling 그룹을 생성해야 합니다. Auto Scaling 그룹이 이러한 가용 영역 중 하나에서 온디맨드 인스턴스를 시작하면 해당 가용 영역에 예약된 용량으로 인스턴스가 실행됩니다. 원하는 용량이 충족되기 전에 리소스 그룹의 용량 예약이 소진되면 예약 용량을 초과한 모든 용량을 일반 온디맨드 용량으로 시작합니다.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateName=my-launch-template,Version='1' \
    --min-size 6 \
    --max-size 6 \
    --vpc-zone-identifier "subnet-5f46ec3b,subnet-0ecac448" \
    --availability-zones us-east-1a us-east-1b
```

관련 리소스

예제 구현은 다음 AWS 샘플 GitHub 리포지토리의 AWS CloudFormation 템플릿을 참조하십시오. https://github.com/aws-samples/ aws-auto-scaling-backed - by-on-demand-capacity -reservations/.

용량 예약에 대해 알아볼 때 다음과 같은 관련 항목을 활용하세요.

- 온디맨드 용량 예약
  - Amazon EC2 사용 설명서에서 용량 예약 생성
  - Amazon EC2 사용 설명서의 온디맨드 용량 예약
  - AWS 클라우드 운영 및 마이그레이션 블로그에서 Amazon EC2 온디맨드 용량 예약 그룹을 타겟팅하세요.
- 용량 블록(기간이 정의된 용량 예약)
  - Amazon EC2 사용 설명서의 ML용 용량 블록
  - 사용 Capacity Blocks 기계 학습 워크로드용

명령줄에서 다음을 사용하여 Auto Scaling 그룹을 생성합니다. AWS CloudShell

지원되는 AWS 리전 경우 에서 직접 실행되는 브라우저 기반의 사전 인증된 셸을 사용하여 AWS CLI AWS CloudShell 명령을 실행할 수 있습니다. AWS Management Console선호하는 셸 (Bash 또는 Z 셸) 을 사용하여 서비스에 대해 AWS CLI 명령을 실행할 수 있습니다. PowerShell

다음 두 가지 방법 중 하나를 AWS Management Console 사용하여 AWS CloudShell 에서 시작할 수 있습니다.

- 콘솔 탐색 표시줄에서 AWS CloudShell 아이콘을 선택합니다. 검색 상자 오른쪽에 있습니다.
- 콘솔 탐색 표시줄의 검색 상자를 사용하여 CloudShell검색한 다음 CloudShell옵션을 선택합니다.

새 브라우저 창에서 처음 실행하면 AWS CloudShell 시작 패널이 표시되고 주요 기능이 나열됩니다. 이 패널을 닫으면 셸이 콘솔 자격 증명을 구성하고 전달하는 동안 상태 업데이트가 제공됩니다. 명령 프롬프트가 표시되면 셸이 상호 작용할 준비가 된 것입니다.

이 서비스에 대한 자세한 내용은 AWS CloudShell 사용 설명서를 참조하세요.

AWS CloudFormation을 이용한 Auto Scaling 그룹 생성

Amazon EC2 Auto Scaling은 리소스를 모델링하고 설정하는 데 도움이 되는 서비스인 와 통합되어 있으므로 AWS 리소스와 인프라를 생성하고 관리하는 데 드는 시간을 줄일 수 있습니다. AWS CloudFormation원하는 모든 AWS 리소스 (예: Auto Scaling 그룹) 를 설명하는 템플릿을 만들고 해당 리소스를 AWS CloudFormation 프로비저닝 및 구성합니다.

를 사용하면 템플릿을 재사용하여 AWS CloudFormation Amazon EC2 Auto Scaling 리소스를 일관되고 반복적으로 설정할 수 있습니다. 리소스를 한 번 설명한 다음 여러 AWS 계정 지역과 지역에서 동일한 리소스를 반복해서 프로비저닝하십시오.

Amazon EC2 Auto Scaling 및 템플릿 AWS CloudFormation

Amazon EC2 Auto Scaling 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 AWS CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이 템플릿은 AWS CloudFormation 스택에 프로비저닝하려는 리소스를 설명합니다. JSON이나 YAML에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하여 템플릿을 시작하는 데 도움

을 받을 수 있습니다. AWS CloudFormation 자세한 내용은 디자이너란 무엇입니까? 를 참조하십시오. AWS CloudFormation AWS CloudFormation 사용 설명서에서

Amazon EC2 Auto Scaling용 스택 템플릿 생성을 시작하려면 다음 작업을 완료하세요.

- 를 사용하여 시작 템플릿을 생성합니다 AWS::EC2::LaunchTemplate.
- 그룹 그룹을 사용하여 Auto Scaling AWS::AutoScaling::AutoScaling 생성합니다.

Application Load Balancer 뒤에 Auto Scaling 그룹을 배포하는 방법을 보여주는 연습은 AWS CloudFormation 사용 설명서의 연습: 확장 가능한 로드 밸런싱 웹 서버 생성을 참조하세요.

AWS CloudFormation 사용 설명서의 다음 섹션에서 Auto Scaling 그룹 및 관련 리소스를 생성하는 템플릿 스니펫의 유용한 추가 예를 찾을 수 있습니다.

- Amazon EC2 Auto Scaling 리소스 유형 참조: 리소스 유형 참조
- 다음을 사용하여 Amazon EC2 Auto Scaling 리소스를 구성합니다. AWS CloudFormation

에 대해 자세히 알아보십시오. AWS CloudFormation

자세히 AWS CloudFormation알아보려면 다음 리소스를 참조하십시오.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation API Reference
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

다음을 통해 인스턴스 유형 권장 사항을 확인하세요. AWS Compute Optimizer

AWS 에서 제공하는 기능을 사용하여 성능을 개선하거나, 비용을 절감하거나, 두 가지를 모두 수행할 수 있도록 Amazon EC2 인스턴스 유형 권장 사항을 제공합니다. AWS Compute Optimizer이러한 권장 사항을 사용하여 Auto Scaling 그룹의 새 인스턴스 유형으로 이동할지 여부를 결정할 수 있습니다.

권장 사항을 만들기 위해 Compute Optimizer는 기존 인스턴스 사양과 최근 지표 기록을 분석합니다. 그런 다음, 컴파일된 데이터를 사용하여 기존 성능 워크로드를 처리하는 데 가장 적합하게 최적화된 Amazon EC2 인스턴스 유형을 권장합니다. 권장 사항은 시간당 인스턴스 요금과 함께 반환됩니다.

> **Note**
>
> Compute Optimizer에서 권장 사항을 받으려면 먼저 Compute Optimizer를 옵트인해야 합니다. 자세한 내용은 AWS Compute Optimizer 사용 설명서의 AWS Compute Optimizer시작하기를 참조하세요.

내용

- 제한 사항
- 조사 결과
- 권장 사항 보기
- 권장 사항 평가를 위한 고려 사항

제한 사항

Compute Optimizer는 M, C, R, T 및 X 인스턴스 유형을 시작 및 실행하도록 구성된 Auto Scaling 그룹의 인스턴스를 위한 권장 사항을 생성합니다. 하지만 AWS Graviton2 프로세서 기반 -g 인스턴스 유형 (예: C6g) 및 네트워크 대역폭 성능이 더 높은 -n 인스턴스 유형 (예: M5n) 에 대한 권장 사항은 생성되지 않습니다.

Auto Scaling 그룹은 단일 인스턴스 유형(즉, 혼합 인스턴스 유형 아님)을 실행하도록 구성되어야 하고, 연결된 조정 정책이 없어야 하며, 원하는 용량, 최소 및 최대 용량의 값이 동일해야 합니다(예: 인스턴스 수가 고정된 Auto Scaling 그룹). Compute Optimizer는 이러한 구성 요구 사항을 모두 충족하는 Auto Scaling 그룹의 인스턴스에 대한 권장 사항을 생성합니다.

조사 결과

Compute Optimizer는 Auto Scaling 그룹에 대한 결과를 다음과 같이 분류합니다.

- 최적화되지 않음 - 워크로드에 더 나은 성능을 제공할 수 있는 권장 사항을 Compute Optimizer에서 확인한 경우 Auto Scaling 그룹이 최적화되지 않은 것으로 간주됩니다.
- 최적화됨 – 선택한 인스턴스 유형에 따라 워크로드를 실행하도록 그룹이 올바르게 프로비저닝되었다고 Compute Optimizer에서 판단한 경우 Auto Scaling 그룹이 최적화된 것으로 간주됩니다. 최적화된 리소스의 경우 Compute Optimizer에서 새로운 인스턴스 유형을 권장하는 경우가 있습니다.
- 없음 – 이 Auto Scaling 그룹에 대한 권장 사항이 없습니다. 이 문제는 Compute Optimizer를 옵트인한지 12시간이 지나지 않았거나, Auto Scaling 그룹이 실행된 지 30시간이 지나지 않았거나, Auto

Scaling 그룹 또는 인스턴스 유형이 Compute Optimizer에서 지원되지 않는 경우에 발생할 수 있습니다. 자세한 내용은 제한 사항(을)를 참조하세요.

권장 사항 보기

Compute Optimizer를 옵트인한 후 Auto Scaling 그룹에 대해 생성되는 결과 및 권장 사항을 볼 수 있습니다. 최근에 옵트인한 경우 권장 사항을 최대 12시간 동안 사용하지 못할 수 있습니다.

Auto Scaling 그룹에 대해 생성된 권장 사항을 보려면

1. https://console.aws.amazon.com/compute-optimizer/에서 Compute Optimizer 콘솔을 엽니다.

   대시보드 페이지가 열립니다.
2. 모든 Auto Scaling 그룹에 대한 권장 사항 보기를 선택합니다.
3. Auto Scaling 그룹을 선택합니다.
4. 세부 정보 보기를 선택합니다.

   기본 테이블 설정에 따라 미리 구성된 보기에서 최대 세 개의 서로 다른 인스턴스 권장 사항이 표시되도록 보기가 변경됩니다. 또한 Auto Scaling 그룹에 대한 최근 CloudWatch 지표 데이터 (평균 CPU 사용률, 평균 네트워크 입력, 평균 네트워크 출력) 를 제공합니다.

권장 사항 중 하나를 사용할지를 결정합니다. 성능 향상, 비용 절감 또는 이 두 가지 모두를 위해 최적화할 것인지를 결정합니다.

Auto Scaling 그룹의 인스턴스 유형을 변경하려면 새 시작 구성을 사용할 수 있도록 시작 템플릿 또는 Auto Scaling 그룹을 업데이트하세요. 기존 인스턴스는 이전 구성을 계속 사용합니다. 기존 인스턴스를 업데이트하려면 인스턴스를 종료해서 Auto Scaling 그룹으로 대체하도록 하거나 자동 크기 조정 기능이 종료 정책에 따라 최신 인스턴스로 이전 인스턴트를 점차 대체하도록 허용합니다.

> **Note**
>
> 최대 인스턴스 수명 및 인스턴스 새로 고침 기능을 사용하면 Auto Scaling 그룹의 기존 인스턴스를 대체하여 새 시작 템플릿 또는 시작 구성을 사용하는 새 인스턴스를 시작할 수도 있습니다. 자세한 내용은 최대 인스턴스 수명을 기준으로 Auto Scaling 인스턴스 교체 및 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다.(을)를 참조하세요.

권장 사항 평가를 위한 고려 사항

새 인스턴스 유형으로 이동하기 전에 다음 사항을 고려하세요.

- 권장 사항은 사용량을 예측하지 않습니다. 권장 사항은 최근 14일 기간의 사용량을 기준으로 합니다. 향후 사용량 요구 사항을 충족할 것으로 예상되는 인스턴스 유형을 선택해야 합니다.
- 그래프로 표시된 지표를 집중적으로 살펴보고 실제 사용량이 인스턴스 용량보다 낮은지 확인합니다. 또한 지표 데이터 (평균, 피크, 백분위수) 를 확인하여 EC2 인스턴스 CloudWatch 권장 사항을 더 자세히 평가할 수 있습니다. 예를 들어, CPU 백분율 지표가 하루 동안 어떻게 변화하고 수용해야 하는 피크가 있는지 확인합니다. 자세한 내용은 Amazon 사용 CloudWatch 설명서의 사용 가능한 지표 보기를 참조하십시오.
- Compute Optimizer는 T3, T3a, T2 인스턴스 등 성능 버스트가 가능한 인스턴스에 대한 권장 사항을 제공할 수 있습니다. 기준 이상으로 주기적으로 버스트하는 경우 새 인스턴스 유형의 vCPU에 따라 계속 버스트할 수 있어야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 CPU 크레딧 및 성능 저하 인스턴스의 기준 성능을 참조하십시오.
- 예약 인스턴스를 구매한 경우 온디맨드 인스턴스 요금이 예약 인스턴스로 청구될 수 있습니다. 현재 인스턴스 유형을 변경하기 전에 먼저 예약 인스턴스 사용률 및 적용 범위에 미치는 영향을 평가합니다.
- 가능한 경우 최신 세대 인스턴스로의 변환을 고려합니다.
- 다른 인스턴스 패밀리로 마이그레이션할 때 현재 인스턴스 유형과 새 인스턴스 유형이 가상화, 아키텍처 또는 네트워크 유형 측면에서 호환되어야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 크기 조정 호환성을 참조하십시오.
- 마지막으로 각 권장 사항에 대해 제공되는 성능 위험 등급을 고려합니다. 성능 위험은 권장 인스턴스 유형이 워크로드의 성능 요구 사항을 충족하는지 여부를 검증하기 위해 얼마나 많은 노력을 기울여야 하는지를 나타냅니다. 또한 변경 전후에 엄격한 로드 및 성능 테스트를 수행하는 것이 좋습니다.

추가 리소스

이 페이지의 주제 외에도 다음 리소스를 참조하세요.

- Amazon EC2 인스턴스 유형
- AWS Compute Optimizer 사용 설명서

Elastic Load Balancing을 사용하여 Auto Scaling 그룹에서 들어오는 애플리케이션 트래픽을 분산하십시오.

Elastic Load Balancing (Elastic Load Balancing) 은 들어오는 애플리케이션 트래픽을 실행 중인 모든 EC2 인스턴스에 자동으로 분산합니다. Elastic Load Balancing을 사용하면 단 하나의 인스턴스에도 부하가 걸리지 않도록 트래픽 라우팅을 최적화하여 들어오는 요청을 관리할 수 있습니다. Auto Scaling 그룹에서 Elastic Load Balancing 사용하려면 Auto Scaling 그룹에 로드 밸런서를 연결합니다. 그러면 Auto Scaling 그룹으로 들어오는 모든 웹 트래픽에 대하여 단일 접점의 역할을 하는 로드 밸런서에 해당 그룹을 등록합니다.

Auto Scaling 그룹에서 Elastic Load Balancing을 사용하는 경우 로드 밸런서에 개별 EC2 인스턴스를 등록할 필요가 없습니다. Auto Scaling 그룹에서 시작되는 인스턴스가 로드 밸런서에 자동으로 등록됩니다. 마찬가지로 Auto Scaling 그룹에 의해 종료된 인스턴스는 로드 밸런서에서 자동으로 등록 취소됩니다.

Auto Scaling 그룹에 로드 밸런서를 연결한 후 Elastic Load Balancing 지표(예: 대상당 Application Load Balancer 요청 수)를 사용하여 수요 변동에 따라 그룹의 인스턴스 수를 조정하도록 Auto Scaling 그룹을 구성할 수 있습니다.

선택적으로, Auto Scaling 그룹에 Elastic Load Balancing 상태 확인을 추가하여 Amazon EC2 Auto Scaling에서 이러한 추가 상태 확인을 기반으로 비정상 인스턴스를 식별하여 교체할 수 있도록 할 수 있습니다. 그렇지 않으면 대상 그룹의 정상 호스트 수가 허용된 수보다 적을 경우 알려주는 CloudWatch 경보를 생성할 수 있습니다.

내용

- Elastic Load Balancing 유형
- Elastic Load Balancing 로드 밸런서 연결 준비
- Auto Scaling 그룹에 Elastic Load Balancing 로드 밸런서를 연결합니다.
- 콘솔에서 애플리케이션 로드 밸런서 또는 네트워크 로드 밸런서를 구성합니다.
- 로드 밸런서의 연결 상태 확인
- 가용 영역 추가
- 가용 영역 제거
- Auto Scaling 그룹에서 대상 그룹 또는 Classic Load Balancer를 분리합니다.
- 다음을 사용하여 Elastic Load Balancing을 사용하는 예제 AWS CLI

Elastic Load Balancing 유형

Elastic Load Balancing에서는 Auto Scaling 그룹에서 사용할 수 있는 4가지 유형의 로드 밸런서 즉, Application Load Balancer, Network Load Balancer, Gateway Load Balancer 및 Classic Load Balancer를 제공합니다.

로드 밸런서 유형이 구성되는 방법에는 주요 차이점이 있습니다. Application Load Balancer, Network Load Balancer 및 Gateway Load Balancer를 사용하여 인스턴스를 대상 그룹에 대상으로 등록하고 트래픽을 대상 그룹에 라우팅합니다. Classic Load Balancer에서는 로드 밸런서에 인스턴스를 직접 등록합니다.

Application Load Balancer

애플리케이션 계층 (HTTP/HTTPS) 에서의 경로 및 부하 분산, 경로 기반 라우팅 지원 Application Load Balancer는 가상 사설 클라우드 () 에 있는 하나 이상의 등록된 대상 (VPC예: EC2 인스턴스) 에 있는 포트로 요청을 라우팅할 수 있습니다.

Network Load Balancer

Layer-4 헤더에서 추출한 주소 정보를 기반으로 전송 계층 (TCP/UDPLayer-4) 에서의 라우팅 및 로드 밸런싱. Network Load Balancer는 트래픽 급증을 처리하고, 클라이언트의 소스 IP를 유지하며, 로드 밸런서의 수명 동안 고정 IP를 사용할 수 있도록 합니다.

Gateway Load Balancer

트래픽을 어플라이언스 인스턴스 플릿으로 분산합니다. 방화벽, 침입 탐지 및 방지 시스템과 기타 어플라이언스와 같은 타사 가상 어플라이언스에 확장성, 가용성 및 단순성을 제공합니다. 게이트웨이 로드 밸런서는 프로토콜을 지원하는 가상 어플라이언스와 함께 작동합니다. GENEVE 추가 기술 통합이 필요하므로 Gateway Load Balancer를 선택하기 전에 사용 설명서를 참조하세요.

Classic Load Balancer

전송 계층 (TCP/SSL) 또는 애플리케이션 계층 (HTTP/HTTPS) 에서 경로와 부하를 분산합니다.

사용 가능한 다양한 유형의 로드 밸런서를 더 자세히 이해하려면 다음 리소스를 참조하십시오.

- Elastic Load Balancing이란 무엇인가요?
- Application Load Balancer란 무엇인가요?
- Network Load Balancer란 무엇인가요?
- Gateway Load Balancer란 무엇입니까?

- Classic Load Balancer란 무엇입니까?

Elastic Load Balancing 로드 밸런서 연결 준비

Elastic Load Balancing 로드 밸런서를 Auto Scaling 그룹에 연결하기 전에 다음 사전 요구 사항을 완료해야 합니다.

- 트래픽을 Auto Scaling 그룹으로 라우팅하는 데 사용되는 로드 밸런서와 대상 그룹을 이미 생성해야 합니다.

  로드 밸런서와 대상 그룹을 생성하는 방법은 두 가지가 있습니다.

  - Elastic Load Balancing 사용 — Auto Scaling 그룹을 생성하기 전에 Elastic Load Balancing 설명서의 절차에 따라 로드 밸런서와 대상 그룹을 생성하고 구성하십시오. Amazon EC2 인스턴스 등록 단계를 건너뛰십시오. 대상 그룹을 EC2 Auto Scaling 그룹에 연결하면 Amazon Auto Scaling에서 자동으로 인스턴스 등록 (및 등록 취소) 을 처리합니다. 자세한 설명은 Elastic Load Balancing 사용자 가이드의 Elastic Load Balancing 시작하기를 참조하세요.
  - Amazon EC2 Auto Scaling 사용 — Amazon EC2 Auto Scaling 콘솔의 기본 구성을 사용하여 로드 밸런서와 대상 그룹을 생성, 구성 및 연결합니다. 자세한 내용은 콘솔에서 애플리케이션 로드 밸런서 또는 네트워크 로드 밸런서를 구성합니다. 단원을 참조하십시오.
- 로드 밸런서를 생성하기 전에 필요한 로드 밸런서의 유형을 파악하십시오. 자세한 내용은 Elastic Load Balancing 유형 단원을 참조하십시오.
- 로드 밸런서와 대상 그룹은 Auto Scaling 그룹과 AWS 계정동일하고 지역은 동일해야 합니다. VPC
- 대상 그룹은 대상 유형으로 `instance`를 지정해야 합니다. Auto Scaling 그룹을 사용하는 경우에는 대상 유형으로 `ip`를 지정할 수 없습니다.
- Auto Scaling 그룹의 시작 템플릿에 로드 밸런서에서 필요한 인바운드 트래픽을 허용하는 올바른 보안 그룹이 포함되어 있지 않은 경우 시작 템플릿을 업데이트해야 합니다. 권장 규칙은 로드 밸런서의 유형과 로드 밸런서에서 사용하는 백엔드 유형에 따라 달라집니다. 예를 들어 트래픽을 웹 서버로 라우팅하려면 로드 밸런서의 포트 80을 통한 인바운드 HTTP 액세스를 허용하십시오. 시작 템플릿을 수정해도 기존 인스턴스는 새 설정으로 업데이트되지 않습니다. 기존 인스턴스를 업데이트하려면 인스턴스 새로 고침을 시작하여 인스턴스를 교체할 수 있습니다. 자세한 내용은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다. 단원을 참조하십시오.
- 또한 시작 템플릿의 보안 그룹은 Elastic Load Balancing이 상태 점검을 수행할 수 있도록 올바른 포트의 로드 밸런서에서의 액세스를 허용해야 합니다.
- 게이트웨이 로드 밸런서 뒤에 가상 어플라이언스를 배포하는 경우, 시작 템플릿의 AMI Amazon 머신 이미지 AMI () 에 프로토콜을 지원하는 GENEVE 서버의 ID를 지정해야 Auto Scaling 그룹이 게이

트웨이 로드 밸런서와 트래픽을 교환할 수 있습니다. 또한 시작 템플릿의 보안 그룹은 포트 6081을 통한 UDP 트래픽을 허용해야 합니다.

> **Tip**
>
> 완료하는 데 시간이 걸리는 부트스트래핑 스크립트가 있다면, Auto Scaling 그룹에 시작 수명 주기 후크를 추가하여 로드 밸런서의 뒤에 인스턴스가 등록된 다음에야 부트스트랩 스크립트가 성공적으로 완료되고 인스턴스의 애플리케이션이 트래픽 수락을 준비하도록 지연시킬 수 있습니다. Amazon EC2 Auto Scaling 콘솔에서 Auto Scaling 그룹을 처음 생성할 때는 수명 주기 후크를 추가할 수 없습니다. 하지만 그룹을 생성한 후에 라이프사이클 후크를 추가할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 수명 주기 후크 단원을 참조하십시오.

대상의 상태 확인을 구성하십시오.

Elastic Load Balancing 로드 밸런서에 등록된 대상에 대한 상태 확인을 구성하여 해당 대상이 트래픽을 제대로 처리할 수 있는지 확인할 수 있습니다. 구체적인 단계는 사용 중인 로드 밸런서 유형에 따라 달라집니다. 자세한 정보는 다음 자료를 참조하십시오.

- Application Load Balancer — 애플리케이션 로드 밸런서 사용 설명서에서 대상 그룹의 상태 확인을 참조하십시오.
- Network Load Balancer — Network Load Balancer 사용 설명서에서 대상 그룹의 상태 확인을 참조하십시오.
- 게이트웨이 로드 밸런서 - 게이트웨이 로드 밸런서 사용 설명서에서 대상 그룹의 상태 확인을 참조하십시오.
- Classic Load Balancer — 클래식 로드 밸런서 사용 설명서의 Classic Load Balancer의 상태 확인 구성을 참조하십시오.

기본적으로 Amazon EC2 Auto Scaling은 인스턴스를 비정상으로 간주하지 않고 Elastic Load Balancing 상태 확인에 실패할 경우 인스턴스를 교체합니다. Auto Scaling 그룹의 기본 상태 점검은 EC2 상태 확인만 가능합니다. 자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 단원을 참조하십시오.

Amazon EC2 Auto Scaling이 Elastic Load Balancing에서 비정상으로 보고된 인스턴스를 대체하도록 하려면 Elastic Load Balancing 상태 확인을 사용하도록 Auto Scaling 그룹을 구성할 수 있습니다. 이렇게 함으로써 Amazon EC2 Auto Scaling은 상태 확인 또는 Elastic Load EC2 Balancing 상태 확인

에 실패할 경우 인스턴스를 비정상으로 간주합니다. 여러 로드 밸런서 대상 그룹 또는 Classic Load Balancer를 그룹에 연결할 경우 인스턴스를 정상으로 간주하려면 모두 해당 인스턴스가 정상이라고 보고해야 합니다. 그 중 하나가 인스턴스를 비정상으로 보고하면 다른 곳에서 정상으로 보고하더라도 Auto Scaling 그룹은 인스턴스를 교체합니다.

Auto Scaling 그룹에 대해 이러한 상태 확인을 활성화하는 방법에 대한 자세한 내용은 을 참조하십시오Auto Scaling 그룹에 Elastic Load Balancing 로드 밸런서를 연결합니다..

> **Note**
>
> 이러한 상태 점검이 가능한 한 빨리 시작되도록 하려면 그룹의 상태 확인 유예 기간을 너무 높게 설정하지 말고 Elastic Load Balancing 상태 점검에서 대상이 요청을 처리할 수 있는지 여부를 판단할 수 있을 만큼 충분히 높게 설정해야 합니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.

Auto Scaling 그룹에 Elastic Load Balancing 로드 밸런서를 연결합니다.

이 항목에서는 Auto Scaling 그룹에 Elastic Load Balancing 로드 밸런서를 연결하는 방법을 설명합니다. 또한 Elastic Load Balancing에서 비정상으로 보고한 인스턴스를 Amazon EC2 Auto Scaling에서 대체하도록 Elastic Load Balancing 상태 확인을 활성화하는 방법도 설명합니다.

기본적으로 Amazon EC2 Auto Scaling은 Amazon 상태 확인을 기반으로 비정상이거나 연결할 수 없는 인스턴스만 대체합니다. EC2 Elastic Load Balancing 상태 확인을 활성화하면, EC2 Auto Scaling 그룹에 연결한 Elastic Load Balancing 로드 밸런서가 비정상으로 보고되면 Amazon Auto Scaling에서 실행 중인 인스턴스를 교체할 수 있습니다.

Auto Scaling 그룹에 애플리케이션 로드 밸런서를 연결하는 방법에 대한 자습서는 을 참조하십시오. 자습서: 조정 및 로드 밸런싱된 애플리케이션 설정

> **Important**
>
> 계속하기 전에 이전 섹션의 모든 사전 요구 사항을 완료하세요.

내용

- 대상 그룹 또는 Classic Load Balancer 연결
- 대상 그룹 또는 Classic Load Balancer 분리

대상 그룹 또는 Classic Load Balancer 연결

Auto Scaling 그룹을 생성하거나 업데이트할 때 하나 이상의 대상 그룹 또는 클래식 로드 밸런서를 연결할 수 있습니다. Application Load Balancer, Network Load Balancer 또는 게이트웨이 로드 밸런서를 연결할 때는 로드 밸런서 자체가 아닌 대상 그룹을 연결합니다.

이 섹션의 다음 단계를 따라 콘솔을 사용합니다.

- 대상 그룹 또는 Classic Load Balancer를 Auto Scaling 그룹에 연결
- Elastic Load Balancing에 대한 상태 점검 켜기

새 Auto Scaling 그룹을 생성할 때 기존 로드 밸런서 연결

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 화면 상단의 탐색 표시줄에서 로드 밸런서를 생성할 때 사용한 항목을 선택합니다. AWS 리전
3. Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.
4. 1단계와 2단계에서 원하는 옵션을 선택하고 3단계: 고급 옵션 구성으로 진행합니다.
5. 로드 밸런싱(Load balancing)에서기존 로드 밸런서에 연결(Attach to an existing load balancer)을 선택합니다.
6. 기존 로드 밸런서에 연결(Attach to an existing load balancer)에서 다음 중 하나를 수행합니다.

   a. Application Load Balancer, Network Load Balancer 및 Gateway Load Balancer의 경우:

      로드 밸런서 대상 그룹에서 선택(Choose from your load balancer target groups)을 선택한 다음 기존 로드 밸런서 대상 그룹(Existing load balancer target groups) 필드에서 대상 그룹을 선택합니다.

   b. Classic Load Balancer의 경우:

      Classic Load Balancer에서 선택(Choose from Classic Load Balancers)을 선택한 다음 Classic Load Balancer 필드에서 로드 밸런서를 선택합니다.
7. (옵션) 건전성 체크, 추가 건전성 체크 유형의 경우, Elastic Load Balancing 건전성 체크 켜기를 선택하십시오.
8. (옵션) 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.

9. 계속해서 Auto Scaling 그룹을 생성합니다. Auto Scaling 그룹이 생성되면 인스턴스가 로드 밸런서에 자동으로 등록됩니다.

Auto Scaling 그룹을 생성한 후 Auto Scaling 그룹에 기존 로드 밸런서를 연결하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 오토 스케일링 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 로드 밸런싱, 편집을 선택합니다.
4. 로드 밸런싱에서 다음 중 하나를 수행합니다.

   a. Application, Network or Gateway Load Balancer target groups(Application, Network 또는 Gateway Load Balancer 대상 그룹)에서 해당 확인란을 선택한 다음 대상 그룹을 선택합니다.

   b. Classic Load Balancer에서 해당 확인란을 선택하고 로드 밸런서를 선택합니다.
5. Update(업데이트)를 선택합니다.

로드 밸런서 연결을 완료하면 해당 로드 밸런서를 사용하는 상태 확인을 선택적으로 켤 수 있습니다.

Elastic Load Balancing 상태 검사를 켜려면

1. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
2. 상태 확인, 추가 상태 확인 유형의 경우 Elastic Load Balancing 상태 확인 활성화를 선택합니다.
3. 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.
4. 업데이트를 선택합니다.

> **Note**
>
> AWS CLI를 사용하여 로드 밸런서를 연결하는 동안 로드 밸런서의 상태를 모니터링할 수 있습니다. Amazon EC2 Auto Scaling에서 인스턴스를 성공적으로 등록하고 하나 이상의 등록된 인스턴스가 상태 확인을 통과하면 상태를 받게 `InService` 됩니다. 자세한 내용은 로드 밸런서의 연결 상태 확인 단원을 참조하십시오.

대상 그룹 또는 Classic Load Balancer 분리

더 이상 로드 밸런서가 필요하지 않으면 다음 절차에 따라 Auto Scaling 그룹에서 분리합니다.

그룹에서 로드 밸런서를 분리하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 로드 밸런싱, 편집을 선택합니다.
4. 로드 밸런싱에서 다음 중 하나를 수행합니다.

   a. Application, Network or Gateway Load Balancer target groups(Application, Network 또는 Gateway Load Balancer 대상 그룹)에서 대상 그룹 옆에 있는 삭제 아이콘(X)을 선택합니다.

   b. Classic Load Balancers에서 로드 밸런서 옆에 있는 삭제 아이콘(X)을 선택합니다.
5. 업데이트를 선택합니다.

대상 그룹 분리가 끝나면 Elastic Load Balancing 상태 확인을 해제할 수 있습니다.

Elastic Load Balancing 상태 검사를 끄려면

1. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
2. 상태 점검, 추가 상태 점검 유형의 경우 Elastic Load Balancing 상태 확인 켜기를 선택 취소하십시오.
3. 업데이트를 선택합니다.

콘솔에서 애플리케이션 로드 밸런서 또는 네트워크 로드 밸런서를 구성합니다.

Auto Scaling 그룹을 생성할 때 다음 절차를 따르면 Application Load Balancer 또는 Network Load Balancer를 생성하여 연결할 수 있습니다.

새 Auto Scaling 그룹을 생성할 때 새 로드 밸런서를 생성하여 연결

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.
3. 1단계와 2단계에서 원하는 옵션을 선택하고 3단계: 고급 옵션 구성으로 진행합니다.
4. Load balancing(로드 밸런싱)에서Attach to a new load balancer(새 로드 밸런서에 연결)를 선택합니다.
   a. Attach to a new load balancer(새 로드 밸런서에서 연결)의 Load balancer type(로드 밸런서 유형)에서 Application Load Balancer 또는 Network Load Balancer를 생성할지 선택합니다.
   b. Load balancer name(로드 밸런서 이름)에 로드 밸런서의 이름을 입력하거나 기본 이름을 유지합니다.
   c. Load balancer scheme(로드 밸런서 체계)에서 퍼블릭 인터넷 경계 로드 밸런서를 생성할지 또는 내부 로드 밸런서의 기본값을 유지할지 선택합니다.
   d. 가용 영역 및 서브넷의 경우, 인스턴스를 시작하기로 선택한 각 가용 영역의 퍼블릭 서브넷을 EC2 선택합니다. 해당 항목은 2단계에서 미리 채워집니다.
   e. Listeners and routing(리스너 및 라우팅)에서 (필요한 경우) 리스너의 포트 번호를 업데이트하고 Default routing(기본 라우팅)에서 Create a target group(대상 그룹 생성)을 선택합니다. 또는 드롭다운 목록에서 기존 대상 그룹을 선택할 수도 있습니다.
   f. 마지막 단계에서 대상 그룹 생성(Create a target group)을 선택한 경우 새 목표 그룹 이름(New target group name)에 대상 그룹의 이름을 입력하거나 기본 이름을 유지합니다.
   g. 로드 밸런서에 태그를 추가하려면 태그 추가(Add tag)를 선택하고 각 태그에 대한 태그 키와 값을 입력합니다.
5. (옵션) 건전성 체크, 추가 건전성 체크 유형의 경우, Elastic Load Balancing 건전성 체크 켜기를 선택하십시오.
6. (옵션) 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.
7. 계속해서 Auto Scaling 그룹을 생성합니다. Auto Scaling 그룹이 생성되면 인스턴스가 로드 밸런서에 자동으로 등록됩니다.

> **Note**
>
> Auto Scaling 그룹을 생성한 후 Elastic Load Balancing 콘솔을 사용하여 추가 리스너를 생성할 수 있습니다. 이는 리스너와 같은 HTTPS 보안 프로토콜을 사용하여 리스너를 생성해야 하는 UDP 경우에 유용합니다. 고유 포트를 사용하는 경우에는 기존 로드 밸런서에 더 많은 리스너를 추가할 수 있습니다.

로드 밸런서의 연결 상태 확인

로드 밸런서를 연결하면 그룹의 인스턴스를 등록하는 동안 인스턴스가 `Adding` 상태로 전환됩니다. 그룹의 모든 인스턴스가 등록된 후에는 인스턴스가 `Added` 상태로 전환됩니다. 등록된 인스턴스가 하나 이상 상태 확인을 통과한 후에는 인스턴스가 `InService` 상태로 전환됩니다. 로드 밸런서가 `InService` 상태에 있을 때 Amazon EC2 Auto Scaling은 비정상으로 보고된 모든 인스턴스를 종료하고 교체할 수 있습니다. 등록된 인스턴스 중 상태 확인을 통과한 인스턴스가 없는 경우(예: 잘못 구성된 상태 확인으로 인해), 로드 밸런서가 `InService` 상태로 전환되지 않습니다. Amazon EC2 Auto Scaling은 인스턴스를 종료하거나 교체하지 않습니다.

로드 밸런서를 분리하면 그룹의 인스턴스를 등록 해제하는 동안 인스턴스가 `Removing` 상태로 전환됩니다. 인스턴스는 등록 해제된 후에도 계속 실행됩니다. 기본적으로 Application Load Balancer, Network Load Balancer 및 Gateway Load Balancer에 대해 Connection Draining(등록 취소 지연)이 활성화되어 있습니다. Connection Draining이 활성화된 경우 Elastic Load Balancing은 인스턴스를 등록 취소하기 전에 진행 중인 요청이 완료되거나 최대 제한 시간이 만료될 때까지 (먼저 일어나는 쪽을) 기다립니다.

AWS Command Line Interface (AWS CLI) 또는 를 사용하여 첨부 상태를 확인할 수 AWS SDKs 있습니다. 콘솔에서는 연결 상태를 확인할 수 없습니다.

를 사용하여 첨부 파일 상태를 AWS CLI 확인하려면

다음 describe-traffic-sources명령은 지정된 Auto Scaling 그룹에 대한 모든 트래픽 소스의 첨부 파일 상태를 반환합니다.

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

이 예제에서는 Auto Scaling 그룹에 연결된 Elastic Load Balancing 대상 그룹을 `State` 요소에 있는 대상 그룹의 연결 상태와 함께 반환합니다. ARN

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/1234567890123456",
            "State": "InService",
            "Type": "elbv2"
        }
    ]
}
```

가용 영역 추가

안전하고 안정적인 지리적 이중화를 활용하기 위해서, 사용자가 작업 중인 리전의 여러 가용 영역에 걸쳐 Auto Scaling 그룹을 확장한 다음 로드 밸런서를 연결하여 들어오는 트래픽을 해당 영역에 분산시킵니다.

한 가용 영역이 비정상이거나 사용할 수 없게 되면 Amazon EC2 Auto Scaling은 영향을 받지 않는 가용 영역에서 새 인스턴스를 시작합니다. 비정상 가용 영역이 정상 상태로 돌아오면 Amazon EC2 Auto Scaling은 자동으로 애플리케이션 인스턴스를 Auto Scaling 그룹의 모든 가용 영역에 균등하게 재분배합니다. Amazon EC2 Auto Scaling은 가장 적은 수의 인스턴스로 가용 영역에서 새 인스턴스를 시작하려고 시도하여 이를 수행합니다. 그러나 시도가 실패할 경우 Amazon EC2 Auto Scaling은 성공할 때까지 다른 가용 영역에서 시작을 시도합니다.

Elastic Load Balancing은 로드 밸런서에 대해 활성화한 각 가용 영역에 대해 로드 밸런서 노드를 생성합니다. 로드 밸런서에 대해 교차 영역 로드 밸런싱을 활성화하면 각 로드 밸런서 노드가 활성화된 모든 가용 영역에 있는 등록된 인스턴스 간에 트래픽을 균등하게 분산합니다. 교차 영역 로드 밸런싱이 비활성화된 경우에는 각각의 로드 밸런서 노드가 해당 가용 영역에만 있는 등록된 인스턴스 간에 요청을 균등하게 분산합니다.

Auto Scaling 그룹을 생성할 때는 1개 이상의 가용 영역을 지정해야 합니다. Auto Scaling 그룹에 가용 영역을 추가한 다음 로드 밸런서에서 해당 가용 영역을 활성화하여 애플리케이션의 가용성을 늘릴 수 있습니다(로드 밸런서가 이 방식을 지원하는 경우).

제한 사항

로드 밸런서에 대해 어떤 가용 영역을 활성화할지 업데이트하려면 다음 제한 사항을 숙지해야 합니다.

- 로드 밸런서에 대해 가용 영역을 활성화할 때 해당 가용 영역에서 서브넷을 하나 지정합니다. 로드 밸런서에 대해 가용 영역당 최대 1개의 서브넷을 활성화할 수 있습니다.

- 인터넷 경계 로드 밸런서의 경우 로드 밸런서에 대해 사용자가 지정하는 서브넷에 사용 가능한 IP 주소가 8개 이상 있어야 합니다.
- Application Load Balancer의 경우 2개 이상의 가용 영역을 활성화해야 합니다.
- Network Load Balancer의 경우 활성화된 가용 영역을 비활성화할 수 없지만 추가 가용 영역을 활성화할 수 있습니다.
- 게이트웨이 로드 밸런서의 경우 활성화된 가용 영역을 비활성화할 수 없지만 추가 가용 영역은 활성화할 수 있습니다.

다음 절차를 수행하면 Auto Scaling 그룹과 로드 밸런서를 추가 가용 영역의 서브넷으로 확장할 수 있습니다.

가용 영역을 추가하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 네트워크, 편집을 선택합니다.
4. Subnets(서브넷)에서 Auto Scaling 그룹에 추가할 가용 영역에 해당하는 서브넷을 선택합니다.
5. 업데이트를 선택합니다.
6. 로드 밸런서의 가용 영역을 업데이트하여 Auto Scaling 그룹과 동일한 가용 영역을 공유하게 하려면 다음 단계를 완료합니다.

   a. 탐색 창의 로드 밸런싱(Load Balancing) 아래에서 로드 밸런서(Load Balancers)를 선택합니다.

   b. 로드 밸런서를 선택합니다.

   c. 다음 중 하나를 수행합니다.

      - Application Load Balancer 및 Network Load Balancer의 경우:

        1. Description(설명) 탭의 Availability Zones(가용 영역)에서 Edit subnets(서브넷 편집)를 선택합니다.
        2. Edit subnets(서브넷 편집) 페이지의 Availability Zones(가용 영역)에서 추가할 가용 영역에 해당하는 확인란을 선택합니다. 해당 영역에 대해 서브넷이 하나뿐인 경우 해당

서브넷이 선택됩니다. 해당 영역에 대해 서브넷이 두 개 이상 있는 경우 서브넷 중 하나를 선택합니다.

- VPCa에 있는 클래식 로드 밸런서의 경우:

  1. [Instances] 탭에서 [Edit Availability Zones]를 선택합니다.
  2. Add and Remove Subnets(서브넷 추가 및 제거) 페이지의 Available subnets(사용 가능한 서브넷)에서 추가 아이콘(+)을 사용하여 서브넷을 선택합니다. 그러면 서브넷이 Selected subnets 아래로 이동합니다.

d. 저장(Save)을 선택합니다.

관련 리소스

Amazon EC2 Auto Scaling은 가용 영역을 변경할 때 그룹을 재조정합니다. 이는 일부 인스턴스를 교체하고 재배포하는 것을 의미합니다. 자세한 내용은 예: 가용 영역 전반에 인스턴스 분산(을)를 참조하세요.

로드 밸런서가 활성화되지 않은 가용 영역에 대상을 등록한 경우 로드 밸런서는 해당 대상으로 트래픽을 라우팅하지 않습니다. 자세한 내용은 Elastic Load Balancing 사용 설명서의 Elastic Load Balancing 작동 방식을 참조하세요.

가용 영역 제거

Auto Scaling 그룹과 로드 밸런서에서 가용 영역을 제거하려면 다음 절차를 수행하세요.

가용 영역을 제거하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 네트워크, 편집을 선택합니다.
4. Subnets(서브넷)에서 Auto Scaling 그룹에서 제거할 가용 영역에 해당하는 서브넷의 삭제 아이콘(X)을 선택합니다. 해당 영역에 대해 서브넷이 두 개 이상 있는 경우 각 서브넷마다 삭제 아이콘(X)을 선택합니다.
5. 업데이트를 선택합니다.

6. 로드 밸런서의 가용 영역을 업데이트하여 Auto Scaling 그룹과 동일한 가용 영역을 공유하게 하려면 다음 단계를 완료합니다.

   a. 탐색 창의 로드 밸런싱(Load Balancing) 아래에서 로드 밸런서(Load Balancers)를 선택합니다.
   b. 로드 밸런서를 선택합니다.
   c. 다음 중 하나를 수행합니다.
      - 애플리케이션 로드 밸런서의 경우:
        1. Description(설명) 탭의 Availability Zones(가용 영역)에서 Edit subnets(서브넷 편집)를 선택합니다.
        2. Edit subnets(서브넷 편집) 페이지의 Availability Zones(가용 영역)에서 해당 확인란을 지워 해당 가용 영역의 서브넷을 제거합니다.
      - 클래식 로드 밸런서의 경우: VPC
        1. [Instances] 탭에서 [Edit Availability Zones]를 선택합니다.
        2. Add and Remove Subnets(서브넷 추가 및 제거) 페이지의 Available subnets(사용 가능한 서브넷)에서 삭제 아이콘(-)을 사용하여 서브넷을 제거합니다. 그러면 서브넷이 Available subnets(가용 서브넷) 아래로 이동합니다.
   d. 저장(Save)을 선택합니다.

Auto Scaling 그룹에서 대상 그룹 또는 Classic Load Balancer를 분리합니다.

더 이상 로드 밸런서가 필요하지 않으면 다음 절차에 따라 Auto Scaling 그룹에서 분리합니다.

그룹에서 로드 밸런서를 분리하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   Auto Scaling 그룹 페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 로드 밸런싱, 편집을 선택합니다.
4. 로드 밸런싱에서 다음 중 하나를 수행합니다.

   a. Application, Network or Gateway Load Balancer target groups(Application, Network 또는 Gateway Load Balancer 대상 그룹)에서 대상 그룹 옆에 있는 삭제 아이콘(X)을 선택합니다.

   b. Classic Load Balancers에서 로드 밸런서 옆에 있는 삭제 아이콘(X)을 선택합니다.
5. 업데이트를 선택합니다.

대상 그룹 분리가 끝나면 Elastic Load Balancing 상태 확인을 해제할 수 있습니다.

Elastic Load Balancing 상태 검사를 끄려면

1. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
2. 상태 점검, 추가 상태 점검 유형의 경우 Elastic Load Balancing 상태 확인 켜기를 선택 취소하십시오.
3. 업데이트를 선택합니다.

다음을 사용하여 Elastic Load Balancing을 사용하는 예제 AWS CLI

AWS Command Line Interface (AWS CLI) 를 사용하여 로드 밸런서와 대상 그룹을 연결, 분리 및 설명하고, Elastic Load Balancing 상태 점검을 추가 및 제거하고, 활성화된 가용 영역을 변경할 수 있습니다.

이 주제에서는 Amazon EC2 Auto Scaling의 일반적인 작업을 수행하는 AWS CLI 명령의 예를 보여줍니다.

> **Important**
>
> 추가 명령 예제는 AWS CLI 명령 참조의 `aws elbv2` 및 `aws elb`(을)를 참조하세요.

내용

- 대상 그룹 또는 Classic Load Balancer 연결
- 대상 그룹 또는 Classic Load Balancer 설명
- Elastic Load Balancing 상태 확인 추가
- 가용 영역 변경
- 대상 그룹 또는 Classic Load Balancer 분리
- Elastic Load Balancing 상태 확인 제거
- 레거시 명령

대상 그룹 또는 Classic Load Balancer 연결

다음 create-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 생성하고 Amazon 리소스 이름 (ARN) 을 지정하여 대상 그룹을 동시에 연결합니다. 대상 그룹은 Application Load Balancer, Network Load Balancer 또는 Gateway Load Balancer와 연결할 수 있습니다.

`--auto-scaling-group-name`, `--vpc-zone-identifier`, `--min-size`, `--max-size`의 샘플 값을 바꿉니다. `--launch-template` 옵션의 경우, *`my-launch-template`* 및 *`1`*를 Auto Scaling 그룹의 시작 템플릿 이름 및 버전으로 바꿉니다. `--traffic-sources`ARN옵션의 경우 샘플을 ARN Application Load Balancer, Network Load Balancer 또는 게이트웨이 로드 밸런서의 대상 그룹으로 대체하십시오.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE1"
```

Auto Scaling 그룹이 생성된 후 attach-traffic-sources명령을 사용하여 추가 대상 그룹을 연결합니다.

다음 명령은 다른 대상 그룹을 동일한 그룹에 추가합니다.

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE2"
```

또는 Classic Load Balancer를 그룹에 연결하려면 다음 예와 같이 create-auto-scaling-group 또는 attach-traffic-sources를 사용할 때 `--traffic-sources` 및 `--type` 옵션을 지정합니다. *`my-classic-load-balancer`*을 Classic Load Balancer의 이름으로 바꿉니다. `--type` 옵션의 경우, **`elb`**의 값을 지정합니다.

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

대상 그룹 또는 Classic Load Balancer 설명

Auto Scaling 그룹에 연결된 로드 밸런서 또는 대상 그룹을 설명하려면 다음 describe-traffic-sources명령을 사용합니다. *`my-asg`*을 사용자 그룹의 이름으로 바꿉니다.

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

이 예제는 Auto ARN Scaling 그룹에 연결한 Elastic Load Balancing 대상 그룹을 반환합니다.

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE1",
            "State": "InService",
            "Type": "elbv2"
        },
        {
            "Identifier": "arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/12345678EXAMPLE2",
            "State": "InService",
            "Type": "elbv2"
        }
    ]
}
```

출력에서 `State` 필드에 대한 설명은 로드 밸런서의 연결 상태 확인(을)를 참조하세요.

Elastic Load Balancing 상태 확인 추가

Auto Scaling 그룹이 인스턴스에 대해 수행하는 상태 확인에 Elastic Load Balancing 상태 확인을 추가하려면 다음 update-auto-scaling-group명령을 사용하고 `--health-check-type` 옵션의 **`ELB`** 값으로 지정합니다. *`my-asg`*을 사용자 그룹의 이름으로 바꿉니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "ELB"
```

새 인스턴스가 상태 확인을 통과하려면 잠시 워밍업을 해야 하는 경우가 많습니다. 유예 기간이 충분한 워밍업 시간을 제공하지 않는 경우 인스턴스가 트래픽을 처리할 준비가 되지 않은 것으로 보일 수 있습니다. Amazon EC2 Auto Scaling은 이러한 인스턴스를 비정상으로 간주하여 교체할 수 있습니다.

상태 확인 유예 기간을 업데이트하려면 다음 예와 같이 update-auto-scaling-group을 사용할 때 `--health-check-grace-period` 옵션을 사용합니다. Replace *`300`* 새 인스턴스가 비정상으로 확인될 경우 인스턴스를 종료하기 전에 서비스를 유지할 시간 (초) 도 포함됩니다.

```
--health-check-grace-period 300
```

자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 단원을 참조하십시오.

가용 영역 변경

가용 영역 변경과 관련하여 다음과 같은 몇 가지 제한 사항을 파악하고 있어야 합니다. 자세한 내용은 가용 영역 추가 단원을 참조하십시오.

Application Load Balancer 또는 Network Load Balancer의 가용 영역을 변경하려면

1. 로드 밸런서의 가용 영역을 변경하기 전에 먼저 Auto Scaling 그룹의 가용 영역을 업데이트하여 지정된 영역에서 인스턴스 유형을 사용할 수 있는지 확인하는 것이 좋습니다.

   Auto Scaling 그룹의 가용 영역을 업데이트하려면 다음 update-auto-scaling-group명령을 사용합니다. 샘플 서브넷을 활성화할 가용 영역에 있는 IDs 서브넷의 IDs 서브넷으로 교체하십시오. 지정된 서브넷이 이전에 활성화된 서브넷을 바꿉니다. *my-asg*을 사용자 그룹의 이름으로 바꿉니다.

   ```
   aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
     --vpc-zone-identifier "subnet-41767929,subnet-cb663da2,subnet-8360a9e7"
   ```

2. 다음 describe-auto-scaling-groups명령을 사용하여 새 서브넷의 인스턴스가 시작되었는지 확인합니다. 인스턴스가 시작된 경우 인스턴스 및 인스턴스 상태 목록이 표시됩니다. *my-asg*을 사용자 그룹의 이름으로 바꿉니다.

   ```
   aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
   ```

3. 다음 set-subnets 명령을 사용하여 로드 밸런서의 서브넷을 지정합니다. 샘플 서브넷을 활성화할 가용 영역에 있는 IDs 서브넷의 IDs 서브넷으로 교체하십시오. 가용 영역당 1개의 서브넷만 지정할 수 있습니다. 지정된 서브넷이 이전에 활성화된 서브넷을 바꿉니다. 로드 *my-lb-arn* ARN 밸런서의 것으로 교체하십시오.

   ```
   aws elbv2 set-subnets --load-balancer-arn my-lb-arn \
     --subnets subnet-41767929 subnet-cb663da2 subnet-8360a9e7
   ```

Classic Load Balancer의 가용 영역을 변경하려면

1. 로드 밸런서의 가용 영역을 변경하기 전에 먼저 Auto Scaling 그룹의 가용 영역을 업데이트하여 지정된 영역에서 인스턴스 유형을 사용할 수 있는지 확인하는 것이 좋습니다.

   Auto Scaling 그룹의 가용 영역을 업데이트하려면 다음 update-auto-scaling-group명령을 사용합니다. 샘플 서브넷을 활성화할 가용 영역에 있는 IDs 서브넷의 IDs 서브넷으로 교체하십시오. 지정된 서브넷이 이전에 활성화된 서브넷을 바꿉니다. *my-asg*을 사용자 그룹의 이름으로 바꿉니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --vpc-zone-identifier "subnet-41767929,subnet-cb663da2"
```

2. 다음 describe-auto-scaling-groups명령을 사용하여 새 서브넷의 인스턴스가 시작되었는지 확인합니다. 인스턴스가 시작된 경우 인스턴스 및 인스턴스 상태 목록이 표시됩니다. *my-asg*을 사용자 그룹의 이름으로 바꿉니다.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name my-asg
```

3. 다음 attach-load-balancer-to-subnets 명령을 사용하여 Classic Load Balancer의 새 가용 영역을 활성화합니다. 샘플 서브넷 ID를 활성화할 가용 영역의 서브넷 ID로 바꿉니다. *my-lb*를 로드 밸런서의 이름으로 바꿉니다.

```
aws elb attach-load-balancer-to-subnets --load-balancer-name my-lb \
  --subnets subnet-cb663da2
```

가용 영역을 비활성화하려면 다음 detach-load-balancer-from -subnets 명령을 사용합니다. 샘플 서브넷 ID를 비활성화할 가용 영역의 서브넷 ID로 바꿉니다. *my-lb*를 로드 밸런서의 이름으로 바꿉니다.

```
aws elb detach-load-balancer-from-subnets --load-balancer-name my-lb \
  --subnets subnet-8360a9e7
```

대상 그룹 또는 Classic Load Balancer 분리

다음 detach-traffic-sources명령은 대상 그룹이 더 이상 필요하지 않을 때 Auto Scaling 그룹에서 대상 그룹을 분리합니다.

`--auto-scaling-group-name` 옵션의 경우, *my-asg*를 그룹 이름으로 바꿉니다. `--traffic-sources`ARN옵션의 경우 샘플을 ARN Application Load Balancer, Network Load Balancer 또는 게이트웨이 로드 밸런서의 대상 그룹으로 대체하십시오.

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:elasticloadbalancing:region:account-
id:targetgroup/my-targets/1234567890123456"
```

그룹에서 Classic Load Balancer를 분리하려면 다음 예와 같이 `--traffic-sources` 및 `--type` 옵션을 지정합니다. *`my-classic-load-balancer`*을 Classic Load Balancer의 이름으로 바꿉니다. `--type` 옵션의 경우, **`elb`**의 값을 지정합니다.

```
--traffic-sources "Identifier=my-classic-load-balancer" --type elb
```

Elastic Load Balancing 상태 확인 제거

Auto Scaling 그룹에서 Elastic Load Balancing 상태 확인을 제거하려면 다음 update-auto-scaling-group명령을 사용하고 `--health-check-type` 옵션의 **`EC2`** 값으로 지정합니다. *`my-asg`*을 사용자 그룹의 이름으로 바꿉니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인(을)를 참조하세요.

레거시 명령

다음 예는 기존 CLI 명령을 사용하여 로드 밸런서와 대상 그룹을 연결, 분리 및 설명하는 방법을 보여줍니다. 이 설명서는 이 설명서를 사용하려는 모든 고객을 위한 참조 자료로 유지됩니다. 기존 CLI 명령은 계속 지원되지만 여러 트래픽 소스 유형을 연결하고 분리할 수 있는 새로운 “트래픽 소스” CLI 명령을 사용하는 것이 좋습니다. 동일한 Auto Scaling 그룹에서 기존 CLI 명령과 “트래픽 소스” CLI 명령을 모두 사용할 수 있습니다.

대상 그룹 또는 Classic Load Balancer(레거시) 연결

대상 그룹을 연결하려면

다음 create-auto-scaling-group명령을 실행하면 대상 그룹이 연결된 Auto Scaling 그룹이 생성됩니다. 애플리케이션 로드 밸런서, 네트워크 로드 밸런서 또는 게이트웨이 로드 밸런서에 대한 대상 그룹의 Amazon 리소스 이름 (ARN) 을 지정합니다.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-
targets/1234567890123456" \
  --min-size 1 --max-size 5
```

다음 attach-load-balancer-target-groups 명령은 대상 그룹을 기존 Auto Scaling 그룹에 연결합니다.

```
aws autoscaling attach-load-balancer-target-groups --auto-scaling-group-name my-asg \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

Classic Load Balancer를 연결하려면

다음 create-auto-scaling-group명령을 실행하면 Classic Load Balancer가 연결된 Auto Scaling 그룹이 생성됩니다.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-configuration-name my-launch-config \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --load-balancer-names "my-load-balancer" \
  --min-size 1 --max-size 5
```

다음 attach-load-balancers명령은 지정된 Classic Load Balancer를 기존 Auto Scaling 그룹에 연결합니다.

```
aws autoscaling attach-load-balancers --auto-scaling-group-name my-asg \
  --load-balancer-names my-lb
```

대상 그룹 또는 Classic Load Balancer(레거시) 설명

대상 그룹을 설명하려면

Auto Scaling 그룹과 관련된 대상 그룹을 설명하려면 describe-load-balancer-target-groups 명령을 사용하십시오. 다음 예에는 대상 그룹이 나열되어 있습니다.*my-asg*.

```
aws autoscaling describe-load-balancer-target-groups --auto-scaling-group-name my-asg
```

Classic Load Balancer를 설명하려면

Auto Scaling 그룹과 관련된 클래식 로드 밸런서를 describe-load-balancers설명하려면 명령을 사용하십시오. 다음 예시는 다음과 같은 클래식 로드 밸런서를 나열합니다.*my-asg*.

```
aws autoscaling describe-load-balancers --auto-scaling-group-name my-asg
```

대상 그룹 또는 Classic Load Balancer(레거시) 분리

대상 그룹을 분리하려면

다음 detach-load-balancer-target-groups 명령은 대상 그룹이 더 이상 필요하지 않을 때 Auto Scaling 그룹에서 대상 그룹을 분리합니다.

```
aws autoscaling detach-load-balancer-target-groups --auto-scaling-group-name my-asg \
  --target-group-arns "arn:aws:elasticloadbalancing:region:account-id:targetgroup/my-targets/1234567890123456"
```

Classic Load Balancer를 분리하려면

다음 detach-load-balancers명령은 더 이상 필요하지 않을 때 Auto Scaling 그룹에서 Classic Load Balancer를 분리합니다.

```
aws autoscaling detach-load-balancers --auto-scaling-group-name my-asg \
  --load-balancer-names my-lb
```

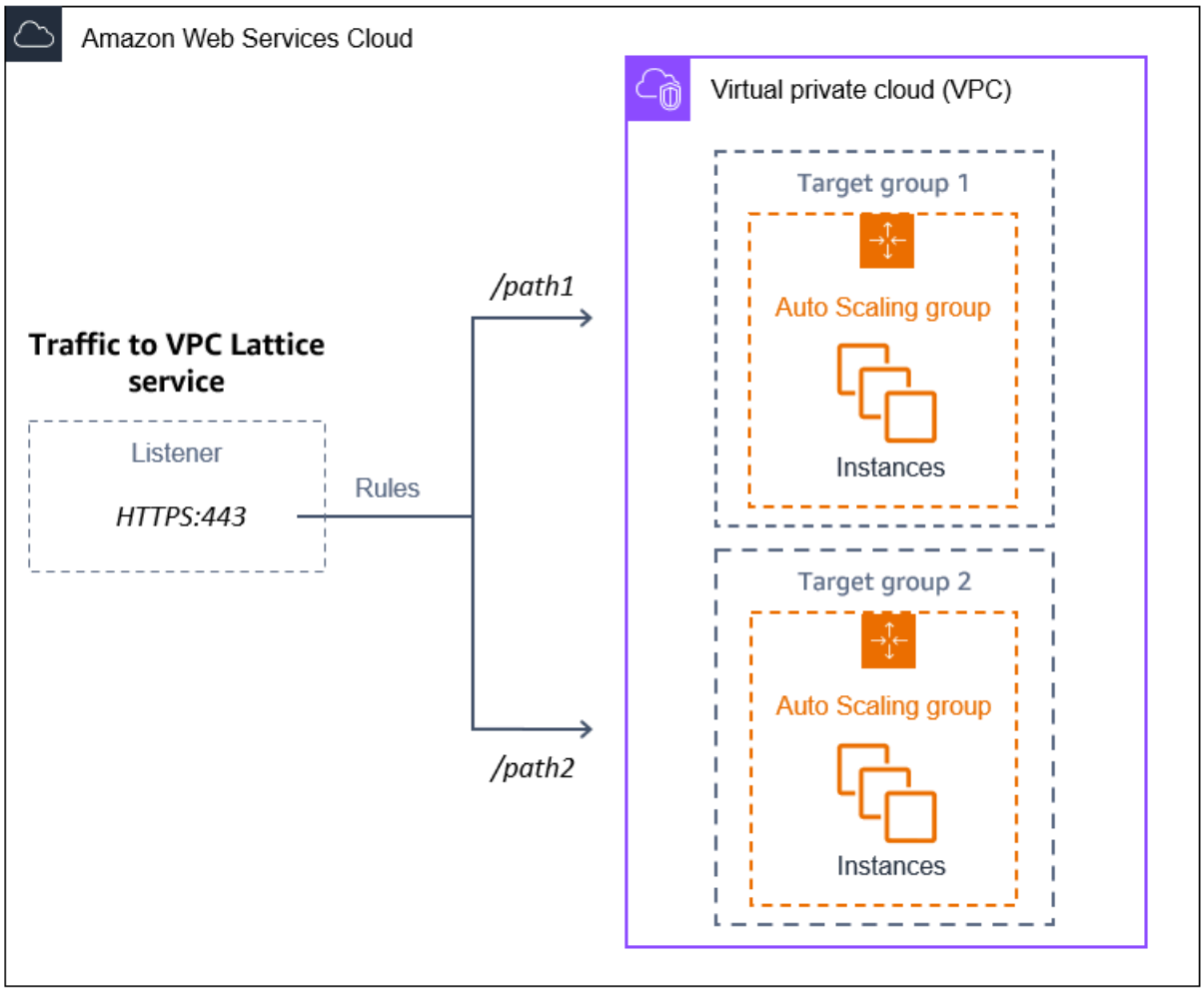
VPCLattice 대상 그룹을 통한 트래픽 흐름 관리

Amazon VPC Lattice를 사용하여 Auto Scaling 그룹 또는 Lambda 함수와 같은 별도의 리소스에서 실행되는 애플리케이션과 서비스 간의 트래픽 및 API 호출 흐름을 관리할 수 있습니다. VPCLattice는 여러 계정과 가상 사설 클라우드 () 에서 모든 서비스를 연결, 보호 및 모니터링할 수 있게 해주는 애플리케이션 네트워킹 서비스입니다. VPCs Lattice에 대해 자세히 알아보려면 VPC Lattice란 VPC 무엇입니까? 를 참조하십시오.

VPCLattice를 시작하려면 먼저 서비스 네트워크와 VPC 관련된 리소스가 서로 연결될 수 있도록 필요한 VPC Lattice 리소스를 생성해야 합니다. 이러한 리소스에는 서비스, 리스너, 리스너 규칙, 대상 그룹이 포함됩니다.

Auto Scaling 그룹을 VPC Lattice 서비스에 연결하려면, 요청을 인스턴스 ID로 등록된 인스턴스로 라우팅하는 서비스의 대상 그룹을 생성하고, 대상 그룹에 요청을 보내는 리스너를 서비스에 추가하십시오. 그런 다음, 대상 그룹을 Auto Scaling 그룹에 연결합니다. Amazon EC2 Auto Scaling은 자동으로 EC2 인스턴스를 대상 그룹의 대상으로 등록합니다. 나중에 Amazon EC2 Auto Scaling에서 인스턴스를 종료해야 하는 경우 종료 전에 대상 그룹에서 인스턴스를 자동으로 등록 취소합니다.

대상 그룹을 연결하면 Auto Scaling 그룹으로 들어오는 모든 요청의 진입점이 됩니다. 다음 다이어그램의 예에서 볼 수 있듯이 Lattice 서비스에 지정된 리스너 규칙을 사용하여 수신 요청을 적절한 대상 그룹으로 라우팅할 수 있습니다. VPC



트래픽이 VPC Lattice를 통해 Auto Scaling 그룹으로 라우팅되면 VPC Lattice는 라운드 로빈 로드 밸런싱을 사용하여 그룹 내 인스턴스 간에 요청을 밸런싱합니다. VPC또한 Lattice는 등록된 인스턴스의 상태를 모니터링하고 트래픽을 정상 인스턴스로만 라우팅할 수 있습니다.

수신 요청에 인스턴스를 계속 사용할 수 있도록 하려면 선택적으로 Auto Scaling 그룹에 VPC Lattice 상태 확인을 추가할 수 있습니다. 이렇게 하면 EC2 인스턴스 중 하나에 장애가 발생할 경우 Auto Scaling 그룹에서 새 인스턴스를 자동으로 시작하여 이를 대체합니다. VPCLattice 상태 확인의 동작은 Elastic Load Balancing 상태 확인의 동작과 유사합니다. Auto Scaling 그룹의 기본 상태 점검은 EC2 상태 확인만 가능합니다.

VPCLattice에 대해 자세히 알아보려면 Amazon Lattice를 통한 서비스 간 연결, 보안 및 모니터링 단순화 — 이제 블로그에서 VPC 정식 버전으로 제공됨을 참조하십시오. AWS

내용

- Auto Scaling 그룹에 VPC Lattice 대상 그룹을 연결할 준비를 하십시오.

- Auto Scaling 그룹에 VPC 래티스 대상 그룹을 연결합니다.
- VPCLattice 대상 그룹의 첨부 파일 상태를 확인하십시오.

Auto Scaling 그룹에 VPC Lattice 대상 그룹을 연결할 준비를 하십시오.

Auto Scaling 그룹에 VPC Lattice 대상 그룹을 연결하기 전에 다음 사전 요구 사항을 완료해야 합니다.

- VPCLattice 서비스 네트워크, 서비스, 리스너 및 대상 그룹을 이미 생성해야 합니다. 자세한 내용은 VPCLattice 사용 설명서의 다음 항목을 참조하십시오.
  - 서비스 네트워크
  - 서비스
  - 리스너
  - 대상 그룹
- 대상 그룹은 Auto Scaling 그룹과 동일한 AWS 계정 VPC, 및 지역에 속해야 합니다.
- 대상 그룹은 대상 유형으로 `instance`를 지정해야 합니다. Auto Scaling 그룹을 사용하는 경우에는 대상 유형으로 `ip`를 지정할 수 없습니다.
- 대상 그룹을 Auto Scaling 그룹에 연결할 수 있는 충분한 IAM 권한이 있어야 합니다. 다음 예제 정책은 대상 그룹을 연결 및 분리하는 데 필요한 최소 필수 권한을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:AttachTrafficSources",
                "autoscaling:DetachTrafficSources",
                "autoscaling:DescribeTrafficSources",
                "vpc-lattice:RegisterTargets",
                "vpc-lattice:DeregisterTargets"
            ],
            "Resource": "*"
        }
    ]
}
```

- Auto Scaling 그룹의 시작 템플릿에 호환되는 보안 그룹과 같은 VPC Lattice에 대한 올바른 설정이 포함되어 있지 않은 경우 시작 템플릿을 업데이트해야 합니다. 시작 템플릿을 수정해도 기존 인스

턴스는 새 설정으로 업데이트되지 않습니다. 기존 인스턴스를 업데이트하려면 인스턴스 새로 고침을 시작하여 인스턴스를 교체할 수 있습니다. 자세한 내용은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다. 단원을 참조하십시오.

- Auto Scaling 그룹에서 VPC Lattice 상태 확인을 활성화하기 전에 애플리케이션 기반 상태 점검을 구성하여 애플리케이션이 예상대로 응답하는지 확인할 수 있습니다. 자세한 내용은 VPCLattice 사용 설명서에서 대상 그룹의 건강 검진을 참조하십시오.

보안 그룹: 인바운드 및 아웃바운드 규칙

보안 그룹은 관련 EC2 인스턴스의 방화벽 역할을 하여 인스턴스 수준에서 인바운드 및 아웃바운드 트래픽을 모두 제어합니다.

> **Note**
>
> 네트워크 구성은 충분히 복잡하므로 Lattice와 함께 사용할 새 보안 그룹을 만드는 것이 좋습니다. VPC 또한 연락해야 AWS Support 하는 경우 더 쉽게 도움을 받을 수 있습니다. 다음 섹션은 사용자가 이 권장 사항을 따른다는 가정을 바탕으로 작성되었습니다.
> Auto Scaling 그룹에서 사용할 수 있는 VPC Lattice용 보안 그룹을 생성하는 방법에 대해 자세히 알아보려면 VPCLattice 사용 설명서의 보안 그룹을 사용한 트래픽 제어를 참조하십시오. 트래픽 흐름과 관련된 문제를 해결하려면 VPCLattice 사용 설명서에서 자세한 내용을 참조하십시오.

보안 그룹을 생성하는 방법에 대한 자세한 내용은 Amazon EC2 User Guide의 보안 그룹 생성을 참조하고 다음 표를 사용하여 선택할 옵션을 결정하십시오.

옵션	값
명칭	기억하기 쉬운 이름
설명	보안 그룹을 식별하는 데 도움이 되는 설명
VPC	Auto Scaling VPC 그룹과 동일합니다.

인바운드 규칙

보안 그룹을 만드는 경우에는 인바운드 규칙이 없습니다. 보안 그룹에 인바운드 규칙을 추가할 때까지 VPC Lattice 서비스 네트워크 내의 클라이언트에서 인스턴스로 들어오는 인바운드 트래픽은 허용되지 않습니다.

VPCLattice 서비스 네트워크 내의 클라이언트가 Auto Scaling 그룹의 인스턴스에 연결할 수 있도록 하려면 Auto Scaling 그룹의 보안 그룹을 올바르게 설정해야 합니다. 이 경우 특정 IP 주소 대신 VPC Lattice의 AWS 관리형 접두사 목록 이름에서 들어오는 트래픽을 허용하도록 인바운드 규칙을 지정하십시오. VPCLattice 접두사 목록은 Lattice에서 표기법으로 사용하는 IP 주소 범위입니다. VPC CIDR 자세한 내용은 Amazon VPC 사용 설명서의 AWS -managed 접두사 목록 사용을 참조하십시오.

보안 그룹에 규칙을 추가하는 방법에 대한 자세한 내용은 Amazon VPC User Guide의 보안 그룹에 규칙 추가를 참조하고 다음 표를 사용하여 선택할 옵션을 결정하십시오.

옵션	값
HTTP규칙	유형: HTTP 출처: com.amazonaws.*region*.vpc-lattice
HTTPS규칙	유형: HTTPS 출처: com.amazonaws.*region*.vpc-lattice

보안 그룹은 상태 저장 방식입니다. 즉, VPC Lattice 서비스 네트워크 내의 클라이언트에서 Auto Scaling 그룹의 인스턴스로의 트래픽을 허용한 다음 이전에 떠난 클라이언트로 응답을 다시 보냅니다.

아웃바운드 규칙

기본적으로 보안 그룹은 모든 아웃바운드 트래픽을 허용하는 아웃바운드 규칙을 포함합니다. 선택적으로 이 기본 규칙을 제거하고 아웃바운드 규칙을 추가하여 특정 보안 요구 사항을 수용할 수 있습니다.

제한 사항

- 혼합 인스턴스 그룹은 지원되지 않습니다. 혼합 인스턴스 정책이 있는 Auto Scaling 그룹에 VPC Lattice 대상 그룹을 연결하려고 하면 오류 메시지가 나타납니다. 현재 혼합 인스턴스가 포함된 Auto

Scaling 그룹은 VPC Lattice 서비스와 통합할 수 없습니다. 이는 로드 밸런싱 알고리즘이 사용 가능한 모든 리소스에 부하를 균등하게 분배하고 인스턴스가 동일한 부하를 처리할 수 있을 만큼 충분히 유사하다고 가정하기 때문입니다.

Auto Scaling 그룹에 VPC 래티스 대상 그룹을 연결합니다.

이 항목에서는 Auto Scaling 그룹에 VPC Lattice 대상 그룹을 연결하는 방법을 설명합니다. 또한 Lattice에서 VPC 비정상으로 보고한 인스턴스를 Amazon EC2 Auto Scaling에서 대체하도록 VPC Lattice 상태 확인을 활성화하는 방법도 설명합니다.

기본적으로 Amazon EC2 Auto Scaling은 Amazon 상태 확인을 기반으로 비정상이거나 연결할 수 없는 인스턴스만 대체합니다. EC2 VPCLattice 상태 확인을 활성화하면 EC2 Auto Scaling 그룹에 연결한 VPC Lattice 대상 그룹 중 하나라도 해당 인스턴스를 비정상으로 보고하면 Amazon Auto Scaling에서 실행 중인 인스턴스를 교체할 수 있습니다. 자세한 내용은 Auto Scaling 그룹의 인스턴스에 대한 상태 확인 단원을 참조하십시오.

> **Important**
>
> 계속하기 전에 이전 섹션의 모든 사전 요구 사항을 완료하세요.

Lattice 대상 그룹 VPC 연결

그룹을 생성하거나 업데이트할 때 하나 이상의 대상 그룹을 Auto Scaling 그룹에 연결할 수 있습니다.

Console

이 섹션의 다음 단계를 따라 콘솔을 사용합니다.

- VPC래티스 대상 그룹을 Auto Scaling 그룹에 연결
- Lattice의 상태 점검을 VPC 켜세요.

새 Auto Scaling 그룹에 VPC 래티스 대상 그룹을 연결하는 방법

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 화면 상단의 탐색 모음에서 대상 그룹을 생성한 AWS 리전 을 선택합니다.
3. Create Auto Scaling group(Auto Scaling 그룹 생성)을 선택합니다.

4. 1단계와 2단계에서 원하는 옵션을 선택하고 3단계: 고급 옵션 구성으로 진행합니다.
5. VPCLattice 통합 옵션의 경우 [VPCLattice 서비스에 연결] 을 선택합니다.
6. VPCLattice 대상 그룹 선택에서 대상 그룹을 선택합니다.
7. (선택 사항) 상태 점검, 추가 상태 점검 유형의 경우 VPCLattice 상태 점검 켜기를 선택합니다.
8. (옵션) 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. 이 시간은 Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.
9. 계속해서 Auto Scaling 그룹을 생성합니다. Auto Scaling 그룹이 생성되면 인스턴스가 VPC Lattice 대상 그룹에 자동으로 등록됩니다.

기존 Auto Scaling 그룹에 VPC 래티스 대상 그룹을 연결하는 방법

다음 절차를 사용해 기존 Auto Scaling 그룹에 서비스의 대상 그룹을 연결합니다.

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. Auto Scaling 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 VPCLattice 통합 옵션에서 편집을 선택합니다.
4. VPCLattice 통합 옵션에서 VPC Lattice 서비스에 연결을 선택합니다.
5. VPCLattice 대상 그룹 선택에서 대상 그룹을 선택합니다.
6. 업데이트를 선택합니다.

대상 그룹 연결을 완료하면 해당 그룹을 사용하는 상태 확인을 선택적으로 켤 수 있습니다.

VPCLattice 상태 검사를 켜려면

1. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
2. 상태 점검, 추가 상태 점검 유형의 경우 VPCLattice 상태 확인 켜기를 선택합니다.
3. 상태 확인 유예 기간에 시간을 초 단위로 입력합니다. 이 시간은 Amazon EC2 Auto Scaling이 상태가 된 후 인스턴스의 상태를 확인하기 전에 기다려야 하는 `InService` 시간입니다. 자세한 내용은 Auto Scaling 그룹의 상태 확인 유예 기간 설정 단원을 참조하십시오.
4. 업데이트를 선택합니다.

AWS CLI

이 섹션의 단계에 AWS CLI 따라 다음을 수행하십시오.

- VPC래티스 대상 그룹을 Auto Scaling 그룹에 연결
- Lattice의 상태 점검을 VPC 켜세요.

Auto Scaling 그룹에 VPC 래티스 타겟 그룹을 연결하는 방법

다음 create-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 생성하고 Amazon 리소스 이름 (ARN) 을 지정하여 VPC Lattice 대상 그룹을 동시에 연결합니다.

`--auto-scaling-group-name`, `--vpc-zone-identifier`, `--min-size`, `--max-size`의 샘플 값을 바꿉니다. `--launch-template`옵션의 경우 *`my-launch-template`* 및 를 VPC Lattice 대상 그룹에 등록된 인스턴스에 대해 생성한 시작 템플릿의 이름 및 버전으로 *`1`* 바꾸십시오. `--traffic-sources`옵션의 경우 샘플을 VPC Lattice 대상 그룹의 ARN ARN 샘플로 바꾸십시오.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name my-asg \
  --launch-template LaunchTemplateName=my-launch-template,Version='1' \
  --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782" \
  --min-size 1 --max-size 5 \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

Auto Scaling 그룹이 이미 생성된 후 다음 attach-traffic-sources명령을 사용하여 VPC Lattice 대상 그룹을 Auto Scaling 그룹에 연결합니다.

```
aws autoscaling attach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

Lattice의 상태 확인을 켜려면 VPC

VPCLattice 대상 그룹에 대해 애플리케이션 기반 상태 점검을 구성한 경우 이러한 상태 확인을 켤 수 있습니다. create-auto-scaling-groupor update-auto-scaling-group명령을 `--health-check-type` 옵션 및 값과 함께 사용합니다. **`VPC_LATTICE`** Auto Scaling 그룹에서 수행하는 상태 확인의 유예 기간을 지정하려면 `--health-check-grace-period` 옵션을 포함하고 값을 초 단위로 제공합니다.

```
--health-check-type "VPC_LATTICE" --health-check-grace-period 60
```

VPC래티스 대상 그룹을 분리합니다.

더 이상 VPC Lattice를 사용할 필요가 없는 경우 다음 절차를 사용하여 Auto Scaling 그룹에서 대상 그룹을 분리하십시오.

Console

이 섹션의 다음 단계를 따라 콘솔을 사용합니다.

- Auto Scaling 그룹에서 VPC 래티스 대상 그룹을 분리합니다.
- Lattice의 상태 확인을 끕니다. VPC

Auto Scaling 그룹에서 VPC 래티스 대상 그룹을 분리하려면

1. 에서 https://console.aws.amazon.com/ec2/Amazon EC2 콘솔을 열고 탐색 창에서 Auto Scaling Groups를 선택합니다.
2. 기존 그룹 옆의 확인란을 선택합니다.

   페이지 하단에 분할 창이 열립니다.
3. 세부 정보 탭에서 VPCLattice 통합 옵션에서 편집을 선택합니다.
4. VPCLattice 통합 옵션에서 대상 그룹 옆에 있는 삭제 (X) 아이콘을 선택합니다.
5. 업데이트를 선택합니다.

대상 그룹 분리가 끝나면 VPC Lattice 상태 확인을 끌 수 있습니다.

VPCLattice 상태 검사를 끄려면

1. 세부 정보 탭에서 상태 확인, 편집을 선택합니다.
2. 상태 점검, 추가 상태 점검 유형의 경우 VPC Lattice 상태 확인 켜기를 선택 취소하십시오.
3. 업데이트를 선택합니다.

AWS CLI

이 섹션의 단계에 따라 다음을 수행하십시오. AWS CLI

- Auto Scaling 그룹에서 VPC 래티스 대상 그룹을 분리합니다.
- Lattice의 상태 확인을 끕니다. VPC

대상 그룹이 더 이상 필요하지 않은 경우 detach-traffic-sources명령을 사용하여 Auto Scaling 그룹에서 대상 그룹을 분리합니다.

```
aws autoscaling detach-traffic-sources --auto-scaling-group-name my-asg \
  --traffic-sources "Identifier=arn:aws:vpc-lattice:region:account-id:targetgroup/
tg-0e2f2665eEXAMPLE"
```

Auto Scaling 그룹에서 VPC Lattice 상태 확인을 더 이상 사용하지 않도록 상태 확인을 update-auto-scaling-group업데이트하려면 명령을 사용하십시오. `--health-check-type` 옵션과 **`EC2`** 값을 포함합니다.

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name my-asg \
  --health-check-type "EC2"
```

VPCLattice 대상 그룹의 첨부 파일 상태를 확인하십시오.

Auto Scaling 그룹에 VPC Lattice 대상 그룹을 연결하면 그룹에 인스턴스를 등록하면서 `Adding` 상태가 됩니다. 그룹의 모든 인스턴스가 등록된 후에는 인스턴스가 `Added` 상태로 전환됩니다. 등록된 인스턴스가 하나 이상 상태 확인을 통과한 후에는 인스턴스가 `InService` 상태로 전환됩니다. 대상 그룹이 `InService` 상태에 있을 때 Amazon EC2 Auto Scaling은 비정상으로 보고된 모든 인스턴스를 종료하고 교체할 수 있습니다. 등록된 인스턴스 중 상태 확인을 통과한 인스턴스가 없는 경우(예: 잘못 구성된 상태 확인으로 인해), 대상 그룹이 `InService` 상태로 전환되지 않습니다. Amazon EC2 Auto Scaling은 인스턴스를 종료하거나 교체하지 않습니다.

대상 그룹을 서비스에서 분리하면 그룹의 인스턴스를 등록 해제하는 동안 인스턴스가 `Removing` 상태로 전환됩니다. 인스턴스는 등록 해제된 후에도 계속 실행됩니다. 기본적으로 Connection Draining(등록 취소 지연)이 활성화되어 있습니다. 연결 드레이닝이 활성화된 경우 VPC Lattice는 진행 중인 요청이 완료되거나 최대 제한 시간이 만료될 때까지 (둘 중 먼저 도래하는 날짜) 기다렸다가 인스턴스 등록을 취소합니다.

() 또는 를 사용하여 첨부 파일 상태를 확인할 수 있습니다. AWS Command Line Interface AWS CLI AWS SDKs 콘솔에서는 연결 상태를 확인할 수 없습니다.

를 사용하여 첨부 파일 상태를 AWS CLI 확인하려면

다음 describe-traffic-sources명령은 지정된 Auto Scaling 그룹에 대한 모든 트래픽 소스의 첨부 파일 상태를 반환합니다.

```
aws autoscaling describe-traffic-sources --auto-scaling-group-name my-asg
```

이 예제는 Auto Scaling 그룹에 연결된 VPC Lattice 대상 그룹을 `State` 요소에 있는 대상 그룹의 연결 상태와 함께 반환합니다. ARN

```
{
    "TrafficSources": [
        {
            "Identifier": "arn:aws:vpc-lattice:region:account-
id:targetgroup/tg-0e2f2665eEXAMPLE",
            "State": "InService",
            "Type": "vpc-lattice"
        }
    ]
}
```

Auto Scaling 이벤트를 처리하는 EventBridge 데 사용합니다.

Amazon은 EventBridge 이전에 CloudWatch Events라고 불렸으며 리소스를 모니터링하는 이벤트 기반 규칙을 설정하고 다른 서비스를 사용하는 대상 작업을 시작하도록 도와줍니다. AWS

Amazon EC2 Auto Scaling의 이벤트는 거의 EventBridge 실시간으로 전송됩니다. 이러한 다양한 이벤트에 대한 응답으로 프로그래밍 작업 및 알림을 호출하는 EventBridge 규칙을 설정할 수 있습니다. 예를 들어, 인스턴스가 시작되거나 종료되는 동안 AWS Lambda 함수를 호출하여 사전 구성된 작업을 수행할 수 있습니다.

EventBridge 규칙 대상에는 AWS Lambda 함수, Amazon SNS 주제, API 대상 AWS 계정, 기타 이벤트 버스 등이 포함될 수 있습니다. 지원되는 대상에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge 대상을 참조하십시오.

Amazon SNS 주제 및 EventBridge 규칙을 사용한 예제로 규칙을 생성하여 시작하십시오. EventBridge 그런 다음 사용자가 인스턴스 새로 고침을 시작하면 체크포인트에 도달할 때마다 Amazon SNS가 이메일로 알려줍니다. 자세한 설명은 인스턴스 새로 고침 이벤트에 대한 EventBridge 규칙 생성 섹션을 참조하세요.

내용

- Amazon EC2 Auto Scaling 이벤트 참조
- 웜 풀 예 이벤트 및 패턴
- Amazon EventBridge 규칙을 사용하여 작업을 자동화하세요

Amazon EC2 Auto Scaling 이벤트 참조

EventBridgeAmazon을 사용하면 수신 이벤트와 일치하는 규칙을 생성하고 처리 대상으로 라우팅하는 규칙을 생성할 수 있습니다.

내용

- 수명 주기 작업 이벤트
- 성공적인 스케일링 이벤트
- 실패한 스케일링 이벤트
- 인스턴스 새로 고침 이벤트

수명 주기 작업 이벤트

Auto Scaling 그룹에 수명 주기 후크를 추가하면 Amazon EC2 Auto Scaling은 인스턴스가 대기 상태로 전환되는 EventBridge 시점에 이벤트를 전송합니다. 이벤트는 최선의 작업에 근거하여 생성됩니다.

이벤트 유형

- 수명 주기 스케일 아웃 작업
- 수명 주기 스케일 인 작업

수명 주기 스케일 아웃 작업

이 예 이벤트에서는 시작 수명 주기 후크로 인해 Amazon EC2 Auto Scaling이 인스턴스를 `Pending:Wait` 상태로 이동했습니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "87654321-4321-4321-4321-210987654321",
```

```
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

수명 주기 스케일 인 작업

이 예 이벤트에서는 해지 수명 주기 후크로 인해 Amazon EC2 Auto Scaling이 인스턴스를 `Terminating:Wait` 상태로 이동했습니다.

> **Important**
>
> Auto Scaling 그룹이 스케일 인 시 인스턴스를 웜 풀로 반환하는 경우, 인스턴스를 웜 풀로 반환하면 `EC2 Instance-terminate Lifecycle Action` 이벤트도 생성할 수 있습니다. 인스턴스가 스케일 인 시 대기 상태로 전환될 때 전달되는 이벤트는 `Destination`의 값으로 `WarmPool`을 갖습니다. 자세한 설명은 Instance reuse policy 섹션을 참조하세요.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken":"87654321-4321-4321-4321-210987654321",
    "AutoScalingGroupName":"my-asg",
    "LifecycleHookName":"my-lifecycle-hook",
    "EC2InstanceId":"i-1234567890abcdef0",
    "LifecycleTransition":"autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata":"additional-info",
    "Origin": "AutoScalingGroup",
```

```
    "Destination": "EC2"
  }
}
```

성공적인 스케일링 이벤트

다음 예는 성공적인 스케일링 이벤트에 대한 이벤트 유형을 보여줍니다. 이벤트는 최선의 작업에 근거하여 생성됩니다.

이벤트 유형

- 성공적인 스케일 아웃 이벤트
- 성공적인 스케일 인 이벤트

성공적인 스케일 아웃 이벤트

다음 예 이벤트에서는 Amazon EC2 Auto Scaling이 인스턴스를 성공적으로 시작했습니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Launching a new EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
```

```
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

성공적인 스케일 인 이벤트

다음 예 이벤트에서는 Amazon EC2 Auto Scaling이 인스턴스를 성공적으로 해지했습니다.

> **Important**
>
> Auto Scaling 그룹이 스케일 인 시 인스턴스를 웜 풀로 반환하는 경우, 인스턴스를 웜 풀로 반환하면 `EC2 Instance Terminate Successful` 이벤트도 생성할 수 있습니다. 인스턴스가 웜 풀로 성공적으로 복귀할 때 전달되는 이벤트는 `Destination`의 값으로 `WarmPool`을 사용합니다. 자세한 설명은 Instance reuse policy 섹션을 참조하세요.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Successful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "InProgress",
    "Description": "Terminating EC2 instance: i-12345678",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "",
```

```
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

실패한 스케일링 이벤트

다음 예는 스케일링 이벤트 실패의 이벤트 유형을 보여줍니다. 이벤트는 최선의 작업에 근거하여 생성됩니다.

이벤트 유형

- 실패한 스케일 아웃 이벤트
- 실패한 스케일 인 이벤트

실패한 스케일 아웃 이벤트

다음 예 이벤트에서는 Amazon EC2 Auto Scaling이 인스턴스를 시작하지 못했습니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Launch Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
      "Subnet ID": "subnet-12345678"
```

```
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "EC2",
    "Destination": "AutoScalingGroup"
  }
}
```

실패한 스케일 인 이벤트

다음 예 이벤트에서는 Amazon EC2 Auto Scaling이 인스턴스를 해지하지 못했습니다.

> **Important**
>
> Auto Scaling 그룹이 스케일 인 시 인스턴스를 웜 풀로 반환하는 경우, 인스턴스를 웜 풀로 반환하지 못하면 `EC2 Instance Terminate Unsuccessful` 이벤트도 생성할 수 있습니다. 인스턴스가 웜 풀로 성공적으로 반환하지 못할 때 전달되는 이벤트는 `Destination`의 값으로 `WarmPool`을 갖습니다. 자세한 설명은 Instance reuse policy 섹션을 참조하세요.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance Terminate Unsuccessful",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn",
    "instance-arn"
  ],
  "detail": {
    "StatusCode": "Failed",
    "AutoScalingGroupName": "my-asg",
    "ActivityId": "87654321-4321-4321-4321-210987654321",
    "Details": {
      "Availability Zone": "us-west-2b",
```

```
      "Subnet ID": "subnet-12345678"
    },
    "RequestId": "12345678-1234-1234-1234-123456789012",
    "StatusMessage": "message-text",
    "EndTime": "yyyy-mm-ddThh:mm:ssZ",
    "EC2InstanceId": "i-1234567890abcdef0",
    "StartTime": "yyyy-mm-ddThh:mm:ssZ",
    "Cause": "description-text",
    "Origin": "AutoScalingGroup",
    "Destination": "EC2"
  }
}
```

인스턴스 새로 고침 이벤트

다음 예는 인스턴스 새로 고침 기능에 대한 이벤트를 보여줍니다. 이벤트는 최선의 작업에 근거하여 생성됩니다.

이벤트 유형

- 체크포인트 도달
- 인스턴스 새로 고침 시작됨
- 인스턴스 새로 고침 성공
- 인스턴스 새로 고침 실패
- 인스턴스 새로 고침 취소됨
- 인스턴스 새로 고침 롤백이 시작되었습니다.
- 인스턴스 새로 고침 롤백 성공
- 인스턴스 새로 고침 롤백 실패

체크포인트 도달

교체된 인스턴스의 수가 체크포인트에 정의된 백분율 임계값에 도달하면 Amazon EC2 Auto Scaling은 다음 이벤트를 보냅니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Checkpoint Reached",
  "source": "aws.autoscaling",
  "account": "123456789012",
```

```
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "ab00cf8f-9126-4f3c-8010-dbb8cad6fb86",
    "AutoScalingGroupName": "my-asg",
    "CheckpointPercentage": "50",
    "CheckpointDelay": "300"
  }
}
```

인스턴스 새로 고침 시작됨

인스턴스 새로 고침 상태가 `InProgress`로 바뀌면 Amazon EC2 Auto Scaling이 다음 이벤트를 보냅니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

인스턴스 새로 고침 성공

인스턴스 새로 고침 상태가 `Successful`로 바뀌면 Amazon EC2 Auto Scaling이 다음 이벤트를 보냅니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
```

```
  "detail-type": "EC2 Auto Scaling Instance Refresh Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

인스턴스 새로 고침 실패

인스턴스 새로 고침 상태가 `Failed`로 바뀌면 Amazon EC2 Auto Scaling이 다음 이벤트를 보냅니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

인스턴스 새로 고침 취소됨

인스턴스 새로 고침 상태가 `Cancelled`로 바뀌면 Amazon EC2 Auto Scaling이 다음 이벤트를 보냅니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
```

```
  "detail-type": "EC2 Auto Scaling Instance Refresh Cancelled",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

인스턴스 새로 고침 롤백이 시작되었습니다.

인스턴스 새로 고침 상태가 `RollbackInProgress`로 바뀌면 Amazon EC2 Auto Scaling이 다음 이벤트를 보냅니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Started",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

인스턴스 새로 고침 롤백 성공

인스턴스 새로 고침 상태가 `RollbackSuccessful`로 바뀌면 Amazon EC2 Auto Scaling이 다음 이벤트를 보냅니다.

```
{
  "version": "0",
```

```
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Succeeded",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

인스턴스 새로 고침 롤백 실패

인스턴스 새로 고침 상태가 `Failed`로 바뀌면 Amazon EC2 Auto Scaling이 다음 이벤트를 보냅니다.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Auto Scaling Instance Refresh Rollback Failed",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "yyyy-mm-ddThh:mm:ssZ",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "InstanceRefreshId": "c613620e-07e2-4ed2-a9e2-ef8258911ade",
    "AutoScalingGroupName": "my-asg"
  }
}
```

웜 풀 예 이벤트 및 패턴

Amazon EC2 Auto Scaling은 아마존에서 미리 정의된 여러 패턴을 지원합니다. EventBridge 이를 통해 이벤트 패턴이 생성되는 방법이 간소화됩니다. 양식에서 필드 값을 선택하고 패턴을 EventBridge 자동으로 생성합니다. 현재 Amazon EC2 Auto Scaling은 웜 풀이 있는 Auto Scaling 그룹에 의해 발생하는 이벤트에 대해 사전 정의된 패턴을 지원하지 않습니다. 패턴을 JSON 객체로 입력해야 합니다. 이 섹션

과 웜풀 이벤트에 대한 EventBridge 규칙 생성 주제에서는 이벤트 패턴을 사용하여 이벤트를 선택하고 대상으로 전송하는 방법을 보여줍니다.

Amazon EC2 Auto Scaling이 보내는 웜 풀 관련 이벤트를 필터링하는 EventBridge 규칙을 EventBridge 생성하려면 이벤트 섹션의 `Destination` 및 필드를 `Origin` `detail` 포함하십시오.

`Origin` 및 `Destination`의 값은 다음과 같을 수 있습니다.

`EC2` | `AutoScalingGroup` | `WarmPool`

내용

- 이벤트 예
- 이벤트 패턴 예

이벤트 예

Auto Scaling 그룹에 수명 주기 후크를 추가하면 Amazon EC2 Auto Scaling은 인스턴스가 대기 상태로 전환되는 EventBridge 시점에 이벤트를 전송합니다. 자세한 정보는 Auto Scaling 그룹의 웜 풀과 함께 라이프사이클 후크 사용을 참조하세요.

이 섹션에는 Auto Scaling 그룹에 웜 풀이 있는 경우, 이러한 이벤트의 예가 포함되어 있습니다. 이벤트는 최선의 작업에 근거하여 발생됩니다.

> **Note**
>
> 규모 조정 성공 EventBridge 시 Amazon EC2 Auto Scaling이 전송하는 이벤트에 대한 자세한 내용은 을 참조하십시오. 성공적인 스케일링 이벤트 스케일링이 실패할 때 발생하는 이벤트는 실패한 스케일링 이벤트을 참조하세요.

이벤트 예

- 수명 주기 스케일 아웃 작업
- 수명 주기 스케일 인 작업

수명 주기 스케일 아웃 작업

인스턴스가 스케일 아웃 이벤트에 대한 대기 상태로 전환될 때 전달되는 이벤트는 `detail-type`의 값이 `EC2 Instance-launch Lifecycle Action`입니다. `detail` 객체에서 `Origin` 및 `Destination` 속성의 값은 인스턴스가 어디에서 왔고 어디로 가는지를 보여줍니다.

이 예 스케일 아웃 이벤트에서는 새 인스턴스가 시작되고 웜 풀에 추가되어 상태가 `Warmed:Pending:Wait`(으)로 변경됩니다. 자세한 설명은 웜 풀의 인스턴스에 대한 수명 주기 상태 전환 섹션을 참조하세요.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-13T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "71514b9d-6a40-4b26-8523-05e7eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "EC2",
    "Destination": "WarmPool"
  }
}
```

이 예 스케일 아웃 이벤트에서는 인스턴스가 웜 풀로부터 Auto Scaling 그룹에 추가되었기 때문에 그 상태가 `Pending:Wait`으로 바뀝니다. 자세한 설명은 웜 풀의 인스턴스에 대한 수명 주기 상태 전환 섹션을 참조하세요.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-launch Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2021-01-19T00:35:52.359Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
```

```
  "detail": {
    "LifecycleActionToken": "19cc4d4a-e450-4d1c-b448-0de67EXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-launch-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
    "NotificationMetadata": "additional-info",
    "Origin": "WarmPool",
    "Destination": "AutoScalingGroup"
  }
}
```

수명 주기 스케일 인 작업

인스턴스가 스케일 인 이벤트에 대한 대기 상태로 전환될 때 전달되는 이벤트는 `detail-type`의 값이 `EC2 Instance-terminate Lifecycle Action`입니다. `detail` 객체에서 `Origin` 및 `Destination` 속성의 값은 인스턴스가 어디에서 왔고 어디로 가는지를 보여줍니다.

이 예 스케일 인 이벤트에서는 인스턴스가 웜 풀로 반환되었기 때문에 그 상태가 `Warmed:Pending:Wait`로 바뀝니다. 자세한 설명은 웜 풀의 인스턴스에 대한 수명 주기 상태 전환 섹션을 참조하세요.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "EC2 Instance-terminate Lifecycle Action",
  "source": "aws.autoscaling",
  "account": "123456789012",
  "time": "2022-03-28T00:12:37.214Z",
  "region": "us-west-2",
  "resources": [
    "auto-scaling-group-arn"
  ],
  "detail": {
    "LifecycleActionToken": "42694b3d-4b70-6a62-8523-09a1eEXAMPLE",
    "AutoScalingGroupName": "my-asg",
    "LifecycleHookName": "my-termination-lifecycle-hook",
    "EC2InstanceId": "i-1234567890abcdef0",
    "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
    "NotificationMetadata": "additional-info",
    "Origin": "AutoScalingGroup",
    "Destination": "WarmPool"
  }
```

```
}
```

이벤트 패턴 예

앞의 섹션에서는 Amazon EC2 Auto Scaling에서 발생하는 이벤트 예를 제공합니다.

EventBridge 이벤트 패턴은 일치하는 이벤트와 구조가 동일합니다. 패턴은 일치시키려는 필드를 인용하고 찾고 있는 값을 제공합니다.

다음과 같은 이벤트 필드는 작업을 호출하는 규칙에 정의된 이벤트 패턴을 형성합니다.

`"source": "aws.autoscaling"`

Amazon EC2 Auto Scaling에서 시작된 이벤트를 식별합니다.

`"detail-type": "EC2 Instance-launch Lifecycle Action"`

이벤트 유형을 식별합니다.

`"Origin": "EC2"`

인스턴스의 출처를 식별합니다.

`"Destination": "WarmPool"`

인스턴스가 이동하는 위치를 식별합니다.

다음 샘플 이벤트 패턴을 사용하면 웜 풀에 들어가는 인스턴스와 관련된 모든 `EC2 Instance-launch Lifecycle Action` 이벤트를 캡처할 수 있습니다.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
   }
}
```

다음 샘플 이벤트 패턴을 사용하여 스케일 아웃 이벤트로 인해 웜 풀에서 나가는 인스턴스와 연결된 모든 `EC2 Instance-launch Lifecycle Action` 이벤트를 캡처합니다.

```
{
```

```
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "WarmPool" ],
      "Destination": [ "AutoScalingGroup" ]
   }
}
```

다음 샘플 이벤트 패턴을 사용하면 Auto Scaling 그룹으로 직접 시작되는 인스턴스와 관련된 모든 `EC2 Instance-launch Lifecycle Action` 이벤트를 캡처할 수 있습니다.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "Origin": [ "EC2" ],
      "Destination": [ "AutoScalingGroup" ]
   }
}
```

다음 샘플 이벤트 패턴을 사용하여 축소 시 웜 풀로 반환되는 인스턴스와 연결된 모든 `EC2 Instance-terminate Lifecycle Action` 이벤트를 캡처합니다.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-terminate Lifecycle Action" ],
  "detail": {
      "Origin": [ "AutoScalingGroup" ],
      "Destination": [ "WarmPool" ]
   }
}
```

다음 샘플 이벤트 패턴을 사용하면 출발지 또는 목적지와 관계없이 `EC2 Instance-launch Lifecycle Action`과 관련된 모든 이벤트를 캡처할 수 있습니다.

```
{
  "source": [ "aws.autoscaling" ],
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ]
}
```

Amazon EventBridge 규칙을 사용하여 작업을 자동화하세요

Amazon EC2 Auto Scaling에서 이벤트가 발생하면 이벤트 알림이 JSON 파일로 EventBridge Amazon에 전송됩니다. 규칙을 작성하여 이벤트 패턴이 EventBridge 규칙과 일치할 때 취해야 할 작업을 자동화할 수 있습니다. 규칙에 정의된 패턴과 일치하는 이벤트 패턴을 EventBridge 탐지하면 규칙에 지정된 대상 (또는 대상) 을 EventBridge 호출합니다.

이 섹션의 예 절차를 시작점으로 사용할 수 있습니다.

다음 설명서도 유용할 수도 있습니다.

- 인스턴스가 시작될 때 또는 Lambda 함수를 사용하여 해지되기 전에 인스턴스에 대한 사용자 지정 작업을 수행하려면 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성 섹션을 참조하세요.
- CloudTrail로 로그인한 API 호출에서 Lambda 함수를 호출하려면 자습서: Amazon 사용 EventBridge 설명서에서 사용하는 API 호출 AWS 로깅을 참조하십시오. EventBridge
- 이벤트 규칙을 생성하는 방법에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 이벤트에 반응하는 Amazon EventBridge 규칙 생성을 참조하십시오.

주제

- 인스턴스 새로 고침 이벤트에 대한 EventBridge 규칙 생성
- 웜풀 이벤트에 대한 EventBridge 규칙 생성

인스턴스 새로 고침 이벤트에 대한 EventBridge 규칙 생성

다음 예시에서는 이메일 EventBridge 알림을 보내는 규칙을 만듭니다. 인스턴스를 새로 고치는 동안 체크포인트에 도달하면 Auto Scaling 그룹에 의해 이벤트가 발생할 때마다 이 작업이 수행됩니다. Amazon SNS를 사용하여 이메일 알림을 설정하는 절차가 포함되어 있습니다. Amazon SNS를 사용하여 이메일 알림을 전송하려면 먼저 주제를 생성한 다음 해당 주제에 이메일 주소를 구독해야 합니다.

인스턴스 새로 고침 기능에 대한 자세한 설명은 인스턴스 새로 고침을 사용하여 Auto Scaling 그룹의 인스턴스를 업데이트합니다. 섹션을 참조하세요.

SNS 주제 생성

SNS 주제는 논리적 액세스 지점으로 Auto Scaling 그룹에서 알림을 전송하는 데 사용하는 통신 채널입니다. 주제의 이름을 지정하여 주제를 생성합니다.

주제 이름은 다음 요건을 충족해야 합니다.

- 1~256자입니다.
- 대문자 및 소문자 ASCII 문자, 숫자, 밑줄 또는 하이픈을 사용합니다.

자세한 설명은 Amazon Simple Notification Service 개발자 안내서에서 Amazon SNS 주제 생성을 참조하세요.

Amazon SNS 주제 구독

Auto Scaling 그룹에서 주제로 전송하는 알림을 받으려면 엔드포인트가 해당 주제를 구독해야 합니다. 이 절차에서 엔드포인트에 Amazon EC2 Auto Scaling의 알림을 받을 이메일 주소를 지정합니다.

자세한 설명은 Amazon Simple Notification Service 개발자 안내서에서 Amazon SNS 주제 구독을 참조하세요.

Amazon SNS 구독 확인

Amazon SNS는 이전 단계에서 지정한 이메일 주소로 확인 이메일을 보냅니다.

다음 단계를 계속하기 전에 AWS 알림에서 이메일을 열고 링크를 선택하여 구독을 확인했는지 확인하십시오.

에서 승인 메시지를 받게 됩니다. AWS이제 Amazon SNS가 지정된 이메일 주소로 이메일 형식의 알림을 주고받을 수 있도록 구성됩니다.

Amazon SNS 주제로 이벤트 라우팅

선택한 이벤트와 일치하는 규칙을 생성하여 Amazon SNS 주제로 라우팅하여 구독한 이메일 주소에 알립니다.

Amazon SNS 주제로 알림을 보내는 규칙 생성

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙 세부 정보 정의(Define rule detail)에 대해 다음을 수행하십시오:

   a. 규칙의 이름을 입력하고 선택적으로 설명을 입력합니다.

      규칙은 동일한 지역과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.

b. 이벤트 버스(Event bus)에서 기본값(default)을 선택합니다. 계정의 AWS 서비스가 이벤트를 생성하면 해당 이벤트는 항상 계정의 기본 이벤트 버스로 이동합니다.

c. 규칙 유형(Rule type)에서 이벤트 패턴이 있는 규칙(Rule with an event pattern)을 생성합니다.

d. 다음을 선택합니다.

5. 이벤트 패턴 빌드(Build event pattern)에서 다음을 수행하십시오:

a. 이벤트 소스의 경우 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.

b. 이벤트 패턴(Event pattern)에서 다음을 수행하십시오:

i. 이벤트 소스(Event source)에서 AWS 서비스를 선택합니다.

ii. AWS 서비스에서 Auto Scaling을 선택합니다.

iii. Event type(이벤트 유형)에서 Instance Refresh(인스턴스 새로 고침)를 선택합니다.

iv. 기본적으로 이 규칙은 모든 인스턴스 새로 고침 이벤트와 일치합니다. 인스턴스를 새로 고치는 동안 체크포인트에 도달하면 알리는 규칙을 생성하려면 특정 인스턴스 이벤트(Specific instance event(s))를 선택하고 EC2 Auto Scaling 인스턴스 새로 고침 체크포인트에 도달함(EC2 Auto Scaling Instance Refresh Checkpoint Reached)을 선택합니다.

v. 기본적으로 규칙은 지역의 모든 Auto Scaling 그룹과 일치합니다. 규칙을 특정 Auto Scaling 그룹과 일치시키려면 Specific group name(s)(특정 그룹 명칭)를 선택하고 Auto Scaling 그룹을 하나 이상 선택합니다.

vi. 다음을 선택합니다.

6. 대상 선택(Select target(s))에서 다음을 수행하십시오:

a. 대상 유형(Target types)에서 AWS 서비스를 선택합니다.

b. 대상 선택(Select a target)에서 SNS 주제(SNS topic)를 선택합니다.

c. 주제(Topic)에서 Amazon SNS 주제를 선택합니다.

d. (옵션) 추가 설정(Additional settings)에서 선택적으로 추가 설정을 구성할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 이벤트에 반응하는 Amazon EventBridge 규칙 생성을 참조하십시오.

e. 다음을 선택합니다.

7. (옵션) 태그(Tags)에서 선택적으로 하나 이상의 태그를 규칙에 할당하고 다음(Next)을 선택할 수 있습니다.

8. 검토 및 생성(Review and create)에서 규칙의 세부 정보를 검토하고 필요에 따라 수정합니다. 그런 다음 규칙 생성(Create rule)을 선택합니다.

웜풀 이벤트에 대한 EventBridge 규칙 생성

다음 예제에서는 프로그래밍 작업을 호출하는 EventBridge 규칙을 생성합니다. 새 인스턴스가 웜 풀에 추가될 때 Auto Scaling 그룹에 의해 이벤트가 생성될 때마다 이 작업이 수행됩니다.

규칙을 만들기 전에 규칙에서 대상으로 사용할 AWS Lambda 함수를 만드십시오. 이 함수를 규칙의 대상으로 지정해야 합니다. 다음 절차는 새 인스턴스가 웜 풀에 들어갈 때 작동하는 EventBridge 규칙을 만드는 단계만 제공합니다. 수신 이벤트가 규칙과 일치할 때 호출할 간단한 Lambda 함수를 생성하는 방법을 보여주는 입문자용 자습서는 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성 섹션을 참조하세요.

웜 풀을 생성하고 사용하는 방법에 대한 자세한 설명은 웜 풀을 사용하여 부팅 시간이 긴 애플리케이션의 지연 시간을 줄입니다. 섹션을 참조하세요.

Lambda 함수를 호출하는 이벤트 규칙 생성

1. https://console.aws.amazon.com/events/ 에서 아마존 EventBridge 콘솔을 엽니다.
2. 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙 세부 정보 정의(Define rule detail)에 대해 다음을 수행하십시오:
   a. 규칙의 이름을 입력하고 선택적으로 설명을 입력합니다.

      규칙은 동일한 지역과 동일한 이벤트 버스의 다른 규칙과 동일한 이름을 가질 수 없습니다.
   b. 이벤트 버스(Event bus)에서 기본값(default)을 선택합니다. AWS 서비스 계정에서 이벤트를 생성하면 해당 이벤트는 항상 계정의 기본 이벤트 버스로 이동합니다.
   c. 규칙 유형(Rule type)에서 이벤트 패턴이 있는 규칙(Rule with an event pattern)을 생성합니다.
   d. 다음을 선택합니다.
5. 이벤트 패턴 빌드(Build event pattern)에서 다음을 수행하십시오:
   a. 이벤트 소스의 경우 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
   b. Event pattern(이벤트 패턴)에서 Custom pattern (JSON editor)사용자 지정 패턴(JSON 편집기)을 선택하고 다음 패턴을 Event pattern(이벤트 패턴) 상자에 붙여 넣고 #### 텍스트를 Auto Scaling 그룹 명칭으로 바꿉니다.

```
{
  "source": [ "aws.autoscaling" ],
```

```
  "detail-type": [ "EC2 Instance-launch Lifecycle Action" ],
  "detail": {
      "AutoScalingGroupName": [ "my-asg" ],
      "Origin": [ "EC2" ],
      "Destination": [ "WarmPool" ]
   }
}
```

다른 이벤트와 일치하는 규칙을 생성하려면 이벤트 패턴을 수정합니다. 자세한 설명은 이벤트 패턴 예 섹션을 참조하세요.

c. 다음을 선택하세요.

6. 대상 선택(Select target(s))에서 다음을 수행하십시오:

   a. 대상 유형(Target types)에서 AWS 서비스를 선택합니다.

   b. 대상 선택(Select a target)에서 Lambda 함수(Lambda function)를 선택합니다.

   c. 함수(Function)에서 이벤트를 보낼 함수를 선택합니다.

   d. (옵션) 버전/별칭 구성(Configure version/alias)에 대상 Lambda 함수의 버전 및 별칭 설정을 입력합니다.

   e. (옵션) 추가 설정(Additional settings)에 애플리케이션에 적합한 추가 설정을 입력합니다. 자세한 내용은 Amazon EventBridge 사용 설명서의 이벤트에 반응하는 Amazon EventBridge 규칙 생성을 참조하십시오.

   f. 다음을 선택합니다.

7. (옵션) 태그(Tags)에서 선택적으로 하나 이상의 태그를 규칙에 할당하고 다음(Next)을 선택할 수 있습니다.

8. 검토 및 생성(Review and create)에서 규칙의 세부 정보를 검토하고 필요에 따라 수정합니다. 그런 다음 규칙 생성(Create rule)을 선택합니다.

Amazon VPC를 사용하여 Auto Scaling 인스턴스에 네트워크 연결 제공

Amazon VPC (Virtual Private Cloud) 는 사용자가 정의한 논리적으로 격리된 가상 네트워크에서 Auto Scaling 그룹과 같은 AWS 리소스를 시작할 수 있게 해주는 서비스입니다.

Amazon VPC의 서브넷은 해당 VPC의 IP 주소 범위 중 한 부분에 따라 정의되는 가용 구역 이내의 하위분류 단위입니다. 서브넷을 사용하면 보안 및 운영상의 필요에 따라 인스턴스를 그룹화할 수 있습니

다. 서브넷은 서브넷이 만들어진 원래 가용 영역 내에서만 존재합니다. 서브넷 안에서 Auto Scaling 인스턴스를 시작합니다.

인터넷과 서브넷의 인스턴스 사이에 통신할 수 있도록 설정하려면 인터넷 게이트웨이를 만들어 VPC에 연결해야 합니다. 인터넷 게이트웨이를 사용하면 서브넷 안의 리소스가 Amazon EC2 네트워크 엣지를 통해 인터넷에 연결되도록 할 수 있습니다. 서브넷 트래픽이 인터넷 게이트웨이로 라우팅되는 경우 해당 서브넷을 퍼블릭 서브넷이라고 합니다. 서브넷 트래픽이 인터넷 게이트웨이로 라우팅되지 않는 경우 해당 서브넷을 프라이빗 서브넷이라고 합니다. 반드시 인터넷에 연결되어야 하는 리소스의 퍼블릭 서브넷과 인터넷에 연결되지 않아도 되는 리소스의 프라이빗 서브넷을 사용합니다. VPC의 인스턴스에 대한 인터넷 액세스 권한을 제공하는 방법에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터넷 액세스를 참조하세요.

내용

- 기본 VPC
- 기본이 아닌 VPC
- VPC 서브넷 선택 시 고려 사항
- VPC에서 IP 주소 지정
- VPC의 네트워크 인터페이스
- 인스턴스 배치 테넌시
- AWS Outposts
- VPC에 대해 자세히 알 수 있는 추가 리소스

기본 VPC

2013년 12월 4일 AWS 계정 이후에 Auto Scaling 그룹을 생성했거나 새 AWS 리전그룹에서 Auto Scaling 그룹을 생성하려는 경우 기본 VPC가 자동으로 생성됩니다. 기본 VPC는 가용 영역별로 기본 서브넷과 함께 제공됩니다. 기본 VPC가 있는 경우 기본적으로 기본 VPC에서 Auto Scaling 그룹이 만들어집니다.

Amazon VPC 콘솔의 VPC(Your VPCs) 페이지에서 VPC를 확인할 수 있습니다.

기본 VPC에 대한 자세한 내용은 Amazon VPC 사용 설명서의 기본 VPC를 참조하세요.

기본이 아닌 VPC

AWS Management Console 의 VPC 대시보드(VPC Dashboard) 페이지로 이동하고 VPC 생성(Create VPC)을 선택하여 VPC를 추가로 생성하도록 선택할 수 있습니다.

자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

> **Note**
>
> VPC는 AWS 리전의 모든 가용 영역에 적용됩니다. VPC에 서브넷을 추가할 때 여러 가용 영역을 선택하여 해당 서브넷에서 호스팅되는 애플리케이션의 고가용성을 보장합니다. 가용 영역은 AWS 리전에 중복 전원, 네트워킹 및 연결이 있는 하나 이상의 개별 데이터 센터입니다. 가용 영역을 통해 프로덕션 애플리케이션은 고가용성, 내결함성 및 확장성을 갖춥니다.

VPC 서브넷 선택 시 고려 사항

Auto Scaling 그룹에 대해 VPC 서브넷을 선택할 때 다음 사항을 고려하세요.

- Elastic Load Balancing 로드 밸런서를 Auto Scaling 그룹에 연결하는 경우 인스턴스를 퍼블릭 또는 프라이빗 서브넷으로 시작할 수 있습니다. 하지만, 로드 밸런서는 DNS 확인을 지원하기 위해 퍼블릭 서브넷에서 생성되어야 합니다.
- SSH를 통해 Auto Scaling 인스턴스에 직접 액세스하는 경우 퍼블릭 서브넷에서만 인스턴스를 시작할 수 있습니다.
- AWS Systems Manager 세션 관리자를 사용하여 무수신 Auto Scaling 인스턴스에 액세스하는 경우 인스턴스를 퍼블릭 또는 프라이빗 서브넷에서 시작할 수 있습니다.
- 프라이빗 서브넷을 사용하는 경우 Auto Scaling 인스턴스가 퍼블릭 NAT 게이트웨이를 사용하여 인터넷에 액세스하도록 허용할 수 있습니다.
- 기본적으로 기본 VPC PC의 기본 서브넷은 퍼블릭 서브넷입니다.

VPC에서 IP 주소 지정

VPC에서 Auto Scaling 인스턴스를 시작할 때 해당 인스턴스에는 인스턴스가 시작된 서브넷의 CIDR 범위에 속하는 프라이빗 IP 주소가 자동으로 할당됩니다. 이를 통해 인스턴스가 VPC의 다른 인스턴스와 통신할 수 있게 됩니다.

인스턴스에 퍼블릭 IPv4 주소를 할당하도록 시작 템플릿 또는 시작 구성을 설정할 수 있습니다. 인스턴스에 퍼블릭 IP 주소를 할당하면 인스턴스가 인터넷 또는 다른 서비스와 통신할 수 있습니다. AWS

IPv6 주소를 자동으로 할당하도록 구성된 서브넷으로 인스턴스를 시작하면 IPv4 및 IPv6 주소가 모두 수신됩니다. 그렇지 않으면 IPv4 주소만 수신합니다. 자세한 내용은 Amazon EC2 사용 설명서의의 IPv6 주소를 참조하세요.

VPC 또는 서브넷의 CIDR 범위 지정에 관한 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

추가 네트워크 인터페이스를 지정하는 시작 템플릿을 사용할 경우 Amazon EC2 Auto Scaling은 인스턴스 시작 시 추가 프라이빗 IP 주소를 자동으로 할당할 수 있습니다. 각 네트워크 인터페이스에는 인스턴스가 시작되는 서브넷의 CIDR 범위에서 단일 프라이빗 IP 주소가 할당됩니다. 이 경우 시스템은 더 이상 기본 네트워크 인터페이스에 퍼블릭 IPv4 주소를 자동 할당할 수 없습니다. 사용 가능한 탄력적 IP 주소를 Auto Scaling 인스턴스에 연결하지 않으면 퍼블릭 IPv4 주소를 통해 인스턴스에 연결할 수 없습니다.

VPC의 네트워크 인터페이스

VPC의 각 인스턴스에는 기본 네트워크 인터페이스가 있습니다. 주 네트워크 인터페이스는 인스턴스에서 분리할 수 없습니다. 추가 네트워크 인터페이스를 만들어 VPC의 모든 인스턴스에 연결할 수 있습니다. 연결 가능한 네트워크 인터페이스의 개수는 인스턴스 유형에 따라 다릅니다.

시작 템플릿을 사용하여 인스턴스를 시작할 때 추가 네트워크 인터페이스를 지정할 수 있습니다. 그러나, 여러 네트워크 인터페이스를 사용하여 Auto Scaling 인스턴스를 시작하면 인스턴스와 동일한 서브넷에 각 인터페이스가 자동으로 생성됩니다. 이는 Auto Scaling 그룹에 지정된 서브넷을 사용하기 위해 Amazon EC2 Auto Scaling이 시작 템플릿에 정의된 서브넷을 무시하기 때문입니다. 자세한 내용은 Auto Scaling 그룹을 위한 시작 템플릿 생성을 참조하세요.

동일한 서브넷에서 2개 이상의 네트워크 인터페이스를 생성하거나 인스턴스에 연결하면 특히, Amazon Linux 이외 변형을 사용하는 인스턴스에서 비대칭 라우팅과 같은 네트워킹 문제가 발생할 수 있습니다. 이러한 유형의 구성이 필요한 경우 OS 내에서 보조 네트워크 인터페이스를 구성해야 합니다. 예를 들어 Ubuntu EC2 인스턴스에서 보조 네트워크 인터페이스를 작동시키려면 어떻게 해야 합니까? 를 참조하십시오. 지식 센터에서. AWS

인스턴스 배치 테넌시

기본적으로 VPC의 모든 인스턴스는 공유 테넌시 인스턴스로 실행합니다. Amazon EC2 Auto Scaling은 전용 인스턴스 및 전용 호스트도 지원합니다. 자세한 정보는 고급 설정을 사용하여 시작 템플릿 생성을 참조하세요.

AWS Outposts

AWS Outposts 인터넷 게이트웨이, 가상 사설 게이트웨이, Amazon VPC 전송 게이트웨이, VPC 엔드포인트를 포함하여 지역에서 액세스할 수 있는 VPC 구성 요소를 사용하여 Amazon VPC를 지역에서 Outpost로 확장합니다. AWS Outpost는 리전의 가용 영역에 위치하며 복원력을 위해 사용할 수 있는 해당 가용 영역의 확장입니다.

자세한 내용은 AWS Outposts 사용 설명서를 참조하십시오.

Outpost 내에서 Application Load Balancer의 트래픽을 처리하는 Auto Scaling 그룹을 배포하는 방법의 예는 블로그 게시물 Configuring an Application Load Balancer on AWS Outposts(OUT에서 Application Load Balancer 구성)를 참조하세요.

VPC에 대해 자세히 알 수 있는 추가 리소스

다음 주제를 확인하면 VPC 및 서브넷에 대해 자세히 알아볼 수 있습니다.

- VPC의 프라이빗 서브넷
  - 예시: 프라이빗 서브넷과 NAT에 서버가 있는 VPC
  - NAT 게이트웨이
- VPC의 퍼블릭 서브넷
  - 예시: 테스트 환경을 위한 VPC
  - 예시: 웹 및 데이터베이스 서버용 VPC
- Application Load Balancer를 위한 서브넷
  - 로드 밸런서를 위한 서브넷
- 일반 VPC 정보
  - Amazon VPC User Guide
  - VPC 피어링을 사용하여 VPC 연결
  - 탄력적 네트워크 인터페이스
  - VPC 엔드포인트를 사용한 프라이빗 연결

Amazon EC2 Auto Scaling의 보안

클라우드 AWS 보안은 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 기업 간의 AWS 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. Amazon EC2 Auto Scaling에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내AWS 서비스 규정 준수 참조하십시오.
- 클라우드에서의 보안 — 사용하는 AWS 서비스에 따라 책임이 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon EC2 Auto Scaling을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 Amazon EC2 Auto Scaling을 구성하는 방법을 보여줍니다. 또한 Amazon EC2 Auto Scaling 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

- Amazon EC2 Auto Scaling의 인프라 보안
- Amazon EC2 Auto Scaling의 레질리언스
- Amazon EC2 Auto Scaling의 데이터 보호
- Amazon EC2 Auto Scaling의 Identity and Access Management
- Amazon EC2 Auto Scaling에 대한 규정 준수 검증
- Amazon EC2 Auto Scaling 및 인터페이스 VPC 엔드포인트

Amazon EC2 Auto Scaling의 인프라 보안

관리형 서비스인 Amazon EC2 Auto Scaling은 AWS 글로벌 네트워크 보안의 보호를 받습니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Amazon EC2 Auto Scaling에 액세스할 수 있습니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안 (TLS). TLS1.2가 필요하고 TLS 1.3을 권장합니다.
- (임시 디피-헬만) 또는 (타원 곡선 임시 디피-헬만PFS) 와 같이 완벽한 순방향 기밀성 DHE () 을 갖춘 암호 제품군. ECDHE Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 액세스 키 ID와 보안 주체와 연결된 비밀 액세스 키를 사용하여 요청에 서명해야 합니다. IAM 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

Amazon EC2 Auto Scaling에 가상 사설 클라우드 (VPC) 엔드포인트를 사용할 수도 있습니다. 인터페이스 VPC 엔드포인트를 사용하면 Amazon VPC 리소스가 퍼블릭 인터넷에 노출되지 않고도 프라이빗 IP 주소를 사용하여 Amazon EC2 Auto Scaling에 액세스할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 및 인터페이스 VPC 엔드포인트 단원을 참조하세요.

관련 리소스

Amazon에서 제공하는 서비스 트래픽을 격리하는 기능에 대한 자세한 내용은 Amazon EC2 EC2사용 설명서의 EC2 Amazon의 인프라 보안을 참조하십시오.

Amazon EC2 Auto Scaling의 레질리언스

AWS 글로벌 인프라는 가용 영역을 중심으로 구축됩니다 AWS 리전 . AWS 리전 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다. 이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 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.

가용 영역에 대한 AWS 리전 자세한 내용은 글로벌 인프라를 참조하십시오AWS .

가용 영역 설계의 지리적 중복성을 활용하려면 다음을 수행합니다.

- Auto Scaling 그룹을 여러 가용 영역으로 확장합니다.
- 각 가용 영역에 하나 이상의 인스턴스를 유지합니다.
- 로드 밸런서를 연결하여 수신 트래픽을 동일한 가용 영역에 걸쳐 분배합니다. Application Load Balancer를 사용하는 경우 영역 간 부하 분산을 활성화하여 각 EC2 인스턴스가 비슷한 양의 트래픽

을 수신하도록 하십시오. 이렇게 하면 장애 조치 이벤트 중에 기존 인스턴스에 대한 부하 증가의 영향을 제한할 수 있으며 교차 영역 로드 밸런싱을 사용하지 않을 때보다 복원력이 향상됩니다.

- Elastic Load Balancing 상태 확인이 올바르게 구성되었는지, 그리고 Auto Scaling 그룹에서 사용 설정되어 있는지 확인합니다. 그러면 인스턴스가 상태 확인에 실패하면 Elastic Load Balancing은 해당 인스턴스로의 트래픽 전송을 중단하고 트래픽을 정상 인스턴스로 다시 라우팅하는 반면 Amazon EC2 Auto Scaling은 비정상 인스턴스를 대체합니다.

Amazon EC2 Auto Scaling은 다음과 같은 방식으로 애플리케이션 복원력 요구 사항을 지원합니다.

- 인스턴스의 상태 및 연결성 문제를 확인합니다. 인스턴스가 비정상이 되면, 해당 인스턴스는 자동으로 종료되고 새 인스턴스가 시작됩니다.
- 동적 조정 정책이 적용되는 경우, 수신되는 트래픽에 따라 용량을 자동으로 조정합니다.
- 조정 정책을 지원하는 Amazon CloudWatch 지표의 안정성 문제를 감지하고 신뢰할 수 있는 지표를 사용할 수 없는 경우 (예: 데이터 포인트가 누락된 경우) 축소 활동을 일시 중지합니다.
- 그룹 조정에 따라 활성화된 각 가용 영역에서 동일한 수의 인스턴스를 유지하려고 시도합니다.
- 가용 영역을 사용하여 고가용성을 유지합니다. 가용 영역이 비정상 상태가 되면 Amazon EC2 Auto Scaling은 다음을 수행합니다.
  - Auto Scaling 그룹에 대해 활성화된 다른 가용 영역에서 새 인스턴스를 시작합니다.
  - 비정상 가용 영역이 정상 상태로 복귀하는 경우 활성화된 모든 가용 영역에 걸쳐 인스턴스를 재분배합니다.
- 특정 가용 영역에서 인스턴스 시작에 실패하는 경우 활성화된 다른 가용 영역에서 인스턴스 시작을 계속 시도합니다.
- Auto Scaling 그룹과 연결된 로드 밸런서를 통해 인스턴스를 자동으로 등록 및 등록 취소합니다. 이렇게 하면 인스턴스를 별도로 등록 및 등록 취소할 필요가 없습니다.
- Amazon EC2 Auto Scaling 서비스의 컨트롤 플레인 APIs 중단은 기존 Auto Scaling 그룹의 규모 조정에 영향을 주지 않습니다.

관련 리소스

Amazon에서 제공하는 데이터 복원력 요구 사항을 지원하는 데 도움이 되는 기능에 대한 자세한 내용은 Amazon EBS EBS사용 설명서의 Amazon Elastic Block Store의 복원력을 참조하십시오.

Amazon EC2 Auto Scaling의 데이터 보호

AWS 공동 책임 모델 Amazon EC2 Auto Scaling의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든 를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 섹션을 FAQ참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 책임 공유 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management ()를 사용하여 개별 사용자를 설정하는 것이 좋습니다IAM. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다단계 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2가 필요하며 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.
- AWS 암호화 솔루션과 내의 모든 기본 보안 제어를 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 FIPS 를 AWS 통해 에 액세스할 때 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 API사용합니다. 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 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon EC2 Auto Scaling 또는 콘솔, API AWS CLI또는 를 사용하는 기타 AWS 서비스 작업을 수행하는 경우가 포함됩니다 AWS SDKs. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL를 제공하는 경우 해당 서버에 대한 요청을 검증URL하기 위해 에 보안 인증 정보를 포함하지 않는 것이 좋습니다.

Amazon EC2 인스턴스를 시작할 때 인스턴스가 부팅될 때 추가 구성을 수행하기 위해 사용자 데이터를 인스턴스에 전달할 수 있습니다. 또한 인스턴스로 전달되는 사용자 데이터에 기밀 또는 민감한 정보를 절대 넣지 않는 것이 좋습니다.

Amazon EBS 볼륨을 암호화하는 AWS KMS keys 데 사용

를 사용하여 클라우드에 저장된 Amazon EBS 볼륨 데이터를 암호화하도록 Auto Scaling 그룹을 구성할 수 있습니다 AWS KMS keys. Amazon EC2 Auto Scaling은 AWS 관리형 키와 고객 관리형 키를 지원하여 데이터를 암호화합니다. 시작 구성을 사용할 때는 고객 관리형 키를 지정하는 `KmsKeyId` 옵션을 사용할 수 없습니다. 고객 관리형 키를 지정하려면 시작 템플릿을 대신 사용하세요. 자세한 내용은 Auto Scaling 그룹에 대한 시작 템플릿 생성 단원을 참조하십시오. AWS KMS 암호화 키를 생성, 저장 및 관리하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서를 참조하세요.

시작 템플릿 또는 시작 구성을 설정하기 AMI 전에 EBS지원 에서 고객 관리형 키를 구성하거나 기본적으로 암호화를 사용하여 생성한 새 EBS 볼륨 및 스냅샷 복사본의 암호화를 적용할 수도 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 EBS-backed로 암호화 사용 AMIs 및 Amazon EBS 사용 설명서의 기본적으로 암호화를 참조하세요.

> **Note**
>
> 암호화에 고객 관리형 키를 사용할 때 Auto Scaling 인스턴스를 시작하는 데 필요한 키 정책을 설정하는 방법에 대한 자세한 설명은 암호화된 볼륨과 함께 사용하는 데 필요한 AWS KMS 키 정책 섹션을 참조하세요.

관련 리소스

Amazon 에서 제공하는 데이터 보호 지침은 Amazon EBS 사용 설명서의 Amazon Elastic Block Store 의 데이터 보호를 EBS참조하세요.

암호화된 볼륨과 함께 사용하는 데 필요한 AWS KMS 키 정책

Amazon EC2 Auto Scaling은 서비스 연결 역할을 사용하여 다른 에 권한을 위임합니다 AWS 서비스. Amazon EC2 Auto Scaling 서비스 연결 역할은 미리 정의되어 있으며 Amazon EC2 Auto Scaling이 AWS 서비스 사용자를 대신하여 다른 를 호출하는 데 필요한 권한을 포함합니다. 사전 정의된 권한에는 에 대한 액세스도 포함됩니다 AWS 관리형 키. 그러나, 해당 권한에는 이러한 키를 완전히 제어할 수 있는 고객 관리형 키에 대한 액세스 권한은 포함되어 있지 않습니다.

이 주제에서는 Amazon EBS 암호화를 위한 고객 관리형 키를 지정할 때 Auto Scaling 인스턴스를 시작하는 데 필요한 키 정책을 설정하는 방법을 설명합니다.

> **Note**
>
> Amazon EC2 Auto Scaling은 계정에서 암호화된 볼륨을 보호하기 위해 기본값을 사용하는 AWS 관리형 키 데 추가 권한이 필요하지 않습니다.

내용

- 개요
- 키 정책 구성
- 예 1: 고객 관리형 키에 대한 액세스를 허용하는 키 정책 섹션
- 예 2: 고객 관리형 키에 대한 교차 계정 액세스를 허용하는 키 정책 섹션
- AWS KMS 콘솔에서 키 정책 편집

개요

Amazon EC2 Auto Scaling이 인스턴스를 시작할 때 Amazon EBS 암호화에 다음을 사용할 AWS KMS keys 수 있습니다.

- AWS 관리형 키 – Amazon이 EBS 생성, 소유 및 관리하는 계정의 암호화 키입니다. 이 키는 새 계정을 위한 기본 암호화 키입니다. AWS 관리형 키 는 고객 관리형 키를 지정하지 않는 한 암호화에 사용됩니다.
- 고객 관리형 키 - 생성, 소유 및 관리하는 사용자 지정 암호화 키입니다. 자세한 정보는 AWS Key Management Service 개발자 안내서의 키 생성을 참조하세요.

  참고: 키는 대칭이어야 합니다. AmazonEBS은 비대칭 고객 관리형 키를 지원하지 않습니다.

암호화된 볼륨을 지정하는 시작 템플릿 또는 암호화된 스냅샷을 생성하거나 기본적으로 암호화를 활성화할 때 고객 관리형 키를 구성합니다.

키 정책 구성

KMS 키에는 Amazon EC2 Auto Scaling이 고객 관리형 키로 암호화된 Amazon EBS 볼륨으로 인스턴스를 시작할 수 있도록 허용하는 키 정책이 있어야 합니다.

이 페이지의 예제를 사용하여 Amazon EC2 Auto Scaling이 고객 관리형 키에 액세스할 수 있도록 키 정책을 구성합니다. 키를 생성할 때 또는 나중에 고객 관리형 키의 키 정책을 수정할 수 있습니다.

Amazon EC2 Auto Scaling 을 사용하려면 최소한 두 개의 정책 문을 키 정책에 추가해야 합니다.

- 첫 번째 문을 사용하면 `Principal` 요소에 지정된 IAM 자격 증명이 고객 관리형 키를 직접 사용할 수 있습니다. 여기에는 키에 대해 AWS KMS `Encrypt`, `Decrypt`, `ReEncrypt*GenerateDataKey*`, 및 `DescribeKey` 작업을 수행할 수 있는 권한이 포함됩니다.
- 두 번째 문을 사용하면 `Principal` 요소에 지정된 IAM 자격 증명이 `CreateGrant` 작업을 사용하여 AWS KMS 또는 다른 보안 주체와 통합된 에 자체 권한의 하위 집합을 위임 AWS 서비스 하는 권한을 생성할 수 있습니다. 이를 통해 사용자는 키를 사용하여 사용자 대신 암호화된 리소스를 생성할 수 있습니다.

키 정책에 새 정책 명령문을 추가할 경우 정책의 기존 섹션을 변경하지 마세요.

다음 각 예제에서 키 ID 또는 서비스 연결 역할의 이름과 같이 교체해야 하는 인수는 로 표시됩니다.*`user placeholder text`*. 대부분의 경우 서비스 연결 역할의 이름을 Amazon EC2 Auto Scaling 서비스 연결 역할의 이름으로 바꿀 수 있습니다.

자세한 정보는 다음 자료를 참조하십시오.

- 를 사용하여 키를 생성하려면 create-key 를 AWS CLI참조하세요.
- 를 사용하여 키 정책을 업데이트하려면 섹션을 AWS CLI참조하세요put-key-policy.
- 키 ID 및 Amazon 리소스 이름(ARN)을 찾으려면 AWS Key Management Service 개발자 안내서의 키 ID 찾기 및 ARN 섹션을 참조하세요.
- Amazon EC2 Auto Scaling 서비스 연결 역할에 대한 자세한 내용은 섹션을 참조하세요Amazon EC2 Auto Scaling의 서비스 연결 역할.
- Amazon EBS 암호화 및 KMS 일반적으로 Amazon EBS 암호화에 대한 자세한 내용은 Amazon EBS 사용 설명서 및 AWS Key Management Service 개발자 안내서 를 참조하세요.

예 1: 고객 관리형 키에 대한 액세스를 허용하는 키 정책 섹션

고객 관리형 키의 키 정책에 다음 두 정책 문을 추가하여 예제를 키에 대한 액세스가 허용되는 적절한 서비스 연결 역할ARN의 ARN 로 바꿉니다. 이 예제에서는 정책 섹션에 고객 관리형 키를 사용할 수 있는 AWSServiceRoleForAutoScaling 권한이라는 서비스 연결 역할에 부여됩니다.

```
{
   "Sid": "Allow service-linked role use of the customer managed key",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
```

```
            "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

```
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
        ]
    },
    "Action": [
        "kms:CreateGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": true
        }
     }
}
```

예 2: 고객 관리형 키에 대한 교차 계정 액세스를 허용하는 키 정책 섹션

Auto Scaling 그룹과 다른 계정에 고객 관리형 키를 생성하는 경우 키에 대한 크로스 계정 액세스를 허용하는 키 정책과 함께 권한 부여를 사용해야 합니다.

다음 순서로 완료해야 하는 두 단계가 있습니다.

1. 먼저 고객 관리형 키의 키 정책에 다음 두 가지 정책 문을 추가합니다. 예제를 다른 계정ARN의 ARN로 바꿉니다.*111122223333* Auto Scaling 그룹을 생성 AWS 계정 하려는 의 실제 계정 ID를 사용합

니다. 이렇게 하면 지정된 계정의 IAM 사용자 또는 역할에 다음 CLI 명령을 사용하여 키에 대한 권한을 생성할 수 있는 권한을 부여할 수 있습니다. 그러나, 이것만으로는 사용자에게 키에 대한 액세스 권한이 부여되지 않습니다.

```
{
   "Sid": "Allow external account 111122223333 use of the customer managed key",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::111122223333:root"
       ]
   },
   "Action": [
       "kms:Encrypt",
       "kms:Decrypt",
       "kms:ReEncrypt*",
       "kms:GenerateDataKey*",
       "kms:DescribeKey"
   ],
   "Resource": "*"
}
```

```
{
   "Sid": "Allow attachment of persistent resources in external
 account 111122223333",
   "Effect": "Allow",
   "Principal": {
       "AWS": [
           "arn:aws:iam::111122223333:root"
       ]
   },
   "Action": [
       "kms:CreateGrant"
   ],
   "Resource": "*"
}
```

2. 그런 다음, Auto Scaling 그룹을 생성하려는 계정에서 관련 권한을 적절한 서비스 연결 역할에 위임하는 권한 부여를 생성합니다. 권한 부여의 `Grantee Principal` 요소는 적절한 서비스 연결 역할 ARN의 입니다. `key-id` 는 키ARN의 입니다.

다음은 계정에서 라는 서비스 연결 역할을 제공하는 create-grant CLI 명령의 예입니다AWSServiceRoleForAutoScaling.*111122223333* 계정에서 고객 관리형 키를 사용할 수 있는 권한 *444455556666*.

```
aws kms create-grant \
  --region us-west-2 \
  --key-id arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d \
  --grantee-principal arn:aws:iam::111122223333:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling \
  --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
 "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

이 명령이 성공하려면 요청을 하는 사용자에게 `CreateGrant` 작업에 대한 권한이 있어야 합니다.

다음 예제 IAM 정책은 계정의 IAM 자격 증명(사용자 또는 역할)을 허용합니다.*111122223333* 계정에서 고객 관리형 키에 대한 권한 부여 생성 *444455556666*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreationOfGrantForTheKMSKeyinExternalAccount444455556666",
      "Effect": "Allow",
      "Action": "kms:CreateGrant",
      "Resource": "arn:aws:kms:us-
west-2:444455556666:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d"
    }
  ]
}
```

다른 에서 KMS 키에 대한 권한 부여를 생성하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 에서 권한 부여 AWS KMS를 AWS 계정참조하세요.

> **Important**
>
> 피부여자 보안 주체로 지정된 서비스 연결 역할 이름은 기존 역할의 이름이어야 합니다. 권한 부여를 생성한 후 권한 부여가 Amazon EC2 Auto Scaling이 지정된 KMS 키를 사용하도록 허용하도록 하려면 서비스 연결 역할을 삭제하고 다시 생성하지 마십시오.

AWS KMS 콘솔에서 키 정책 편집

이전 섹션의 예에서는 키 정책을 변경하는 유일한 방법인 키 정책에 명령문을 추가하는 방법을 보여줍니다. 키 정책을 변경하는 가장 쉬운 방법은 키 정책에 콘솔 AWS KMS 의 기본 뷰를 사용하고 적절한 키 정책에 대한 키 사용자 중 한 명을 IAM 자격 증명(사용자 또는 역할)으로 만드는 것입니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS Management Console 기본 보기 사용을 참조하세요.

> **Important**
>
> 주의할 점이 있습니다. 콘솔의 기본 보기 정책 문에는 고객 관리형 키에 대한 작업을 수행할 AWS KMS Revoke 수 있는 권한이 포함되어 있습니다. 계정의 고객 관리형 키에 대한 AWS 계정 액세스 권한을 부여하고 이 권한을 부여한 권한을 실수로 취소한 경우 외부 사용자는 더 이상 암호화된 데이터 또는 데이터를 암호화하는 데 사용된 키에 액세스할 수 없습니다.

Amazon EC2 Auto Scaling의 Identity and Access Management

AWS Identity and Access Management (IAM) 은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와줍니다. IAM 관리자는 어떤 사용자가 Amazon EC2 Auto Scaling 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지를 제어합니다. IAM은 추가 AWS 서비스 비용 없이 사용할 수 있습니다.

Amazon EC2 Auto Scaling을 사용하려면 계정에 로그인하기 위한 보안 자격 증명이 필요합니다. AWS 계정 자세한 내용은 IAM 사용 설명서의AWS 보안 자격 증명을 참조하십시오.

IAM 설명서 전체 내용은 IAM 사용 설명서를 참조하세요.

액세스 제어

요청을 인증하는 데 유효한 자격 증명이 있더라도 권한이 없다면 Amazon EC2 Auto Scaling 리소스를 생성하거나 액세스할 수 없습니다. 예를 들어, Auto Scaling 그룹을 생성하고 시작 템플릿을 사용하여 인스턴스를 시작하는 등의 권한이 있어야 합니다.

다음 섹션에서는 IAM 관리자가 Amazon EC2 Auto Scaling 작업 수행 가능자를 제어하여 Amazon EC2 Auto Scaling 리소스를 보호할 수 있도록 IAM을 사용하는 방법에 대한 세부 정보를 제공합니다.

먼저 Amazon EC2 주제를 읽는 것이 좋습니다. Amazon EC2 사용 설명서에서 Amazon EC2의 자격 증명 및 액세스 관리를 참조하십시오. 이 섹션의 각 항목을 읽으면 Amazon EC2에서 제공하는 액세스 제어 권한은 무엇인지, Amazon EC2 Auto Scaling 리소스 권한과 어떻게 결합될 수 있는지 잘 알 수 있을 것입니다.

주제

- Amazon EC2 Auto Scaling과 함께 작동하는 방식 IAM
- Amazon EC2 Auto Scaling API 권한
- AWS Amazon EC2 Auto Scaling에 대한 관리형 정책
- Amazon EC2 Auto Scaling의 서비스 연결 역할
- Amazon EC2 Auto Scaling 자격 증명 기반 정책 예제
- 교차 서비스 혼동된 대리인 방지
- Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다.
- Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할

Amazon EC2 Auto Scaling과 함께 작동하는 방식 IAM

Amazon IAM Auto Scaling을 사용하여 액세스를 관리하기 전에 Amazon EC2 Auto Scaling에서 사용할 수 있는 IAM 기능에 대해 알아보십시오. EC2

IAMAmazon EC2 Auto Scaling과 함께 사용할 수 있는 기능

IAM기능:	아마존 EC2 Auto Scaling 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACLs	아니요
ABAC(정책의 태그)	부분

IAM기능:	아마존 EC2 Auto Scaling 지원
임시 보안 인증	예
서비스 역할	예
서비스 연결 역할	예

Amazon EC2 Auto Scaling 및 기타 IAM 기능이 대부분의 기능과 어떻게 AWS 서비스 작동하는지 자세히 알아보려면 IAM사용 설명서의AWS 서비스 해당 기능을 참조하십시오. IAM

Amazon EC2 Auto Scaling에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 첨부할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 사용 설명서의 IAM정책 생성을 참조하십시오. IAM

IAMID 기반 정책을 사용하면 허용 또는 거부된 작업 및 리소스는 물론 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 사용 IAM설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

Amazon EC2 Auto Scaling 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 첨부하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

계정 간 액세스를 활성화하려면 다른 계정의 전체 계정 또는 IAM 엔티티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할

수 있는 계정의 IAM 관리자는 보안 주체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM사용 설명서의 계정 간 리소스 액세스를 참조하십시오. IAM

Amazon EC2 Auto Scaling에 대한 정책 조치

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

정책 `Action` 요소는 JSON 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 작업이 없는 권한 전용 작업과 같은 몇 가지 예외가 있습니다. API 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

Amazon EC2 Auto Scaling 작업 목록을 보려면 서비스 인증 참조의 Amazon EC2 Auto Scaling에서 정의한 작업을 참조하십시오.

Amazon EC2 Auto Scaling의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
autoscaling
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
  "autoscaling:action1",
  "autoscaling:action2"
]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예컨대, `Describe`라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "autoscaling:Describe*"
```

Amazon EC2 Auto Scaling을 위한 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource`JSON정책 요소는 작업이 적용되는 하나 또는 여러 개의 객체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. Amazon 리소스 이름 (ARN) 을 사용하여 리소스를 지정하는 것이 가장 좋습니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

를 사용하여 ARNs Auto Scaling 그룹을 식별하고 IAM 정책이 적용되는 구성을 시작할 수 있습니다.

Auto Scaling 그룹에는 다음이 ARN 있습니다.

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/asg-name"
```

시작 구성에는 다음이 포함됩니다ARN.

```
"Resource": "arn:aws:autoscaling:region:account-
id:launchConfiguration:uuid:launchConfigurationName/lc-name"
```

`CreateAutoScalingGroup`작업에 Auto Scaling 그룹을 지정하려면 다음 UUID 예와 같이 를 와일드카드 (`*`) 로 바꿔야 합니다.

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/asg-name"
```

`CreateLaunchConfiguration`작업을 포함하는 시작 구성을 지정하려면 다음 UUID 예와 같이 와일드카드 (`*`) 로 바꿔야 합니다.

```
"Resource": "arn:aws:autoscaling:region:account-
id:launchConfiguration:*:launchConfigurationName/lc-name"
```

Amazon EC2 Auto Scaling 리소스 유형 및 해당 ARNs 유형에 대한 자세한 내용은 서비스 인증 참조의 Amazon EC2 Auto Scaling에서 정의한 리소스를 참조하십시오. 각 리소스에 어떤 작업을 지정할 수 있는지 알아보려면 Amazon EC2 Auto Scaling에서 정의한 작업을 참조하십시오. ARN

> **Note**
>
> Auto Scaling 그룹에 대한 액세스를 제어하는 데 사용하는 IAM ARNs 정책의 예는 을 참조하십시오삭제할 수 있는 Auto Scaling 그룹 제어.

모든 Amazon EC2 Auto Scaling 작업이 리소스 수준 권한을 지원하는 것은 아닙니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 리소스로 사용해야 합니다.

다음 Amazon EC2 Auto Scaling 작업은 리소스 수준 권한을 지원하지 않습니다.

- `DescribeAccountLimits`
- `DescribeAdjustmentTypes`
- `DescribeAutoScalingGroups`
- `DescribeAutoScalingInstances`
- `DescribeAutoScalingNotificationTypes`
- `DescribeInstanceRefreshes`
- `DescribeLaunchConfigurations`
- `DescribeLifecycleHooks`
- `DescribeLifecycleHookTypes`
- `DescribeLoadBalancers`
- `DescribeLoadBalancerTargetGroups`
- `DescribeMetricCollectionTypes`
- `DescribeNotificationConfigurations`
- `DescribePolicies`
- `DescribeScalingActivities`
- `DescribeScalingProcessTypes`
- `DescribeScheduledActions`
- `DescribeTags`
- `DescribeTerminationPolicyTypes`
- `DescribeWarmPool`

Amazon EC2 Auto Scaling의 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우, AWS 는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어 리소스에 IAM 사용자 이름이 태그가 지정된 경우에만 리소스에 대한 액세스 권한을 IAM 사용자에게 부여할 수 있습니다. 자세한 내용은 IAM사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하십시오.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM사용 설명서의AWS 글로벌 조건 컨텍스트 키를 참조하십시오.

Amazon EC2 Auto Scaling은 지원되는 작업에 대한 액세스를 제어하고 Auto Scaling 그룹의 구성을 적용하는 데 사용할 수 있는 다음과 같은 조건 키를 지원합니다.

- `autoscaling:InstanceTypes`
- `autoscaling:LaunchConfigurationName`
- `autoscaling:LaunchTemplateVersionSpecified`
- `autoscaling:LoadBalancerNames`
- `autoscaling:MaxSize`
- `autoscaling:MinSize`
- `autoscaling:ResourceTag/`*`key-name`*`: `*`tag-value`*
- `autoscaling:TargetGroupARNs`
- `autoscaling:VPCZoneIdentifiers`

다음 조건 키는 시작 구성 요청 생성에만 사용됩니다.

- `autoscaling:ImageId`
- `autoscaling:InstanceType`
- `autoscaling:MetadataHttpEndpoint`
- `autoscaling:MetadataHttpPutResponseHopLimit`
- `autoscaling:MetadataHttpTokens`
- `autoscaling:SpotPrice`

Amazon EC2 Auto Scaling은 요청의 태그 또는 Auto Scaling 그룹에 있는 태그를 기반으로 권한을 정의하는 데 사용할 수 있는 다음과 같은 글로벌 조건 키도 지원합니다. 자세한 내용은 Auto Scaling 그룹 및 인스턴스에 태그 지정 단원을 참조하십시오.

- `aws:RequestTag/`*`key-name`*`:` *`tag-value`*
- `aws:ResourceTag/`*`key-name`*`:` *`tag-value`*
- `aws:TagKeys:` *`[tag-key, ...]`*

조건 키를 사용할 수 있는 Amazon EC2 Auto Scaling API 작업에 대해서는 서비스 인증 참조의 Amazon EC2 Auto Scaling에서 정의한 작업을 참조하십시오. Amazon EC2 Auto Scaling 조건 키에 대한 자세한 내용은 Amazon EC2 Auto Scaling의 조건 키를 참조하십시오.

> **Note**
>
> 조건 키를 사용하여 지원되는 작업에 대한 액세스를 제어하고 Auto Scaling 그룹의 구성을 적용하는 IAM 정책의 예는 다음 리소스를 참조하십시오.
>
> - 시작 템플릿 및 버전 번호 필요— 이 예제에서는 Auto Scaling 그룹을 생성하거나 업데이트할 때 시작 템플릿과 시작 템플릿의 버전 번호를 지정해야 합니다.
> - 생성할 수 있는 Auto Scaling 그룹 크기 제어— 이 예제에서는 특정 태그를 사용하여 Auto Scaling 그룹을 생성하거나 업데이트할 때 `MinSize` 및 `MaxSize` 속성의 가능한 값에 제약을 적용합니다.
> - 삭제할 수 있는 조정 정책 제어— 이 예에서는 특정 태그가 없는 Auto Scaling 그룹에만 조정 정책을 삭제할 수 있도록 합니다.

ACLs아마존 EC2 오토 스케일링에서

지원ACLs: 아니요

액세스 제어 목록 (ACLs) 은 리소스에 액세스할 수 있는 권한을 가진 주체 (계정 구성원, 사용자 또는 역할) 를 제어합니다. ACLs정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 JSON 비슷합니다.

ABACAmazon EC2 Auto Scaling과 함께

지원 ABAC (정책의 태그): 부분

속성 기반 액세스 제어 (ABAC) 는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM엔티티 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. 의 ABAC 첫 번째 단계는 엔티티와 리소스에 태그를 지정하는 것입니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC빠르게 성장하는 환경에서 유용하며 정책 관리가 복잡해지는 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

에 대한 자세한 내용은 What is ABAC? 를 참조하십시오. ABAC IAM사용 설명서에서. 설정 ABAC 단계가 포함된 자습서를 보려면 사용 IAM설명서의 속성 기반 액세스 제어 사용 (ABAC) 을 참조하십시오.

ABAC태그를 지원하는 리소스에서는 가능하지만 모든 리소스에서 태그를 지원하는 것은 아닙니다. 시작 구성 및 조정 정책은 태그를 지원하지 않지만 Auto Scaling 그룹은 태그를 지원합니다.

자세한 내용은 Auto Scaling 그룹 및 인스턴스에 태그 지정 단원을 참조하십시오.

Amazon EC2 Auto Scaling에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명을 사용하여 로그인하면 작동하지 AWS 서비스 않는 것도 있습니다. 임시 자격 증명을 사용하는 AWS 서비스 방법을 비롯한 추가 정보는 IAM사용 설명서의 AWS 서비스 해당 자격 증명을 참조하십시오. IAM

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하는 경우 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 Single Sign-On (SSO) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM사용 설명서의 역할 전환 (콘솔) 을 참조하십시오.

AWS CLI 또는 를 사용하여 임시 자격 증명을 수동으로 생성할 수 AWS API 있습니다. 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 내용은 의 임시 보안 자격 증명을 참조하십시오. IAM

Amazon EC2 Auto Scaling의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하도록 맡는 IAM역할입니다. IAM관리자는 내부에서 IAM 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 사용 설명서의 역할 만들기를 참조하여 권한을 위임하십시오IAM. AWS 서비스

Amazon SNS 주제 또는 Amazon SQS 대기열에 알림을 보내는 수명 주기 후크를 생성할 때는 Amazon EC2 Auto Scaling이 사용자를 대신하여 Amazon SNS 또는 SQS Amazon에 액세스할 수 있도록 하는 역할을 지정해야 합니다. IAM콘솔을 사용하여 라이프사이클 후크의 서비스 역할을 설정합니다. 이 콘솔에서는 관리형 정책을 사용하여 충분한 권한 세트가 있는 역할을 생성할 수 있습니다. 자세한 내용은 Amazon SNS를 사용하여 알림 수신 및 Amazon SQS 사용하여 알림 수신 단원을 참조하세요.

Auto Scaling 그룹을 생성할 때 선택적으로 서비스 역할을 전달하여 Amazon EC2 인스턴스가 사용자를 대신하여 다른 AWS 서비스 인스턴스에 액세스하도록 허용할 수 있습니다. Amazon EC2 인스턴스의 서비스 역할 (시작 템플릿 또는 시작 구성의 Amazon EC2 인스턴스 프로필이라고도 함) 은 인스턴스 시작 시 Auto Scaling 그룹의 모든 EC2 인스턴스에 할당되는 특수한 유형의 서비스 역할입니다. IAM콘솔을 AWS CLI 사용하여 이 서비스 역할을 생성하거나 편집할 수 있습니다. 자세한 내용은 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 단원을 참조하십시오.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon EC2 Auto Scaling 기능이 작동하지 않을 수 있습니다. Amazon EC2 Auto Scaling에서 이에 대한 지침을 제공하는 경우에만 서비스 역할을 편집하십시오.

Amazon EC2 Auto Scaling의 서비스 연결 역할

서비스 링크 역할 지원: 예

서비스 연결 역할은 에 연결된 서비스 역할 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

Amazon EC2 Auto Scaling 서비스 연결 역할을 생성 또는 관리하는 방법에 대한 자세한 내용은 을 참조하십시오. Amazon EC2 Auto Scaling의 서비스 연결 역할

Amazon EC2 Auto Scaling API 권한

Amazon EC2 Auto Scaling에 대한 정책 조치에 설명된 대로 필요한 Amazon EC2 Auto Scaling API 작업을 호출할 수 있는 권한을 사용자에게 부여해야 합니다. 또한 일부 Amazon EC2 Auto Scaling 작업의 경우 사용자에게 다른 AWS API에서 특정 작업을 호출할 수 있는 권한을 부여해야 합니다.

다른 AWS API의 필수 권한

Amazon EC2 Auto Scaling API 권한 외에도 사용자는 다른 AWS API로부터 다음과 같은 권한을 가져야 관련 작업을 성공적으로 수행할 수 있습니다.

Auto Scaling 그룹 생성(`autoscaling:CreateAutoScalingGroup`)

- `iam:CreateServiceLinkedRole`— 기본 서비스 연결 역할 생성 (해당 역할이 아직 없는 경우)
- `iam:PassRole`— 시작 시 IAM 역할을 서비스나 EC2 인스턴스로 전달합니다. 기본이 아닌 서비스 연결 역할, 수명 주기 후크에 사용되는 IAM 역할 또는 인스턴스 프로파일을 지정하는 시작 템플릿(IAM 역할의 컨테이너)이 제공되는 경우에 필요합니다.
- `ec2:RunInstances`— 시작 템플릿이 제공될 때 인스턴스를 시작합니다.
- `ec2:CreateTags`— 태그 사양이 포함된 시작 템플릿이 제공된 경우 시작 시 인스턴스 및 볼륨에 태그를 지정합니다.

수명 주기 후크 생성(`autoscaling:PutLifecycleHook`)

- `iam:PassRole`— 서비스에 IAM 역할을 전달하는 것. IAM 역할이 제공된 경우에 필요합니다.

VPC Lattice 대상 그룹 연결(`autoscaling:AttachTrafficSources`)

- `vpc-lattice:RegisterTargets`— 대상 그룹에 인스턴스를 자동으로 등록합니다.

VPC Lattice 대상 그룹 분리(`autoscaling:DetachTrafficSources`)

- `vpc-lattice:DeregisterTargets`— 대상 그룹과의 인스턴스 등록을 자동으로 취소합니다.

시작 구성 생성(`autoscaling:CreateLaunchConfiguration`)

- `ec2:DescribeImages`
- `ec2:DescribeInstances`
- `ec2:DescribeInstanceAttribute`
- `ec2:DescribeKeyPairs`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSpotInstanceRequests`
- `ec2:DescribeVpcClassicLink`
- `iam:PassRole`— 시작 시 EC2 인스턴스에 IAM 역할을 전달합니다. 시작 구성에서 인스턴스 프로파일(IAM 역할의 컨테이너)을 지정할 때 필요합니다.

AWS Amazon EC2 Auto Scaling에 대한 관리형 정책

AWS 관리형 정책은 에서 생성하고 관리하는 독립 실행형 정책입니다. AWS AWS 관리형 정책은 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었으므로 사용자, 그룹 및 역할에 권한을 할당하기 시작할 수 있습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수도 있다는 점에 유의하세요. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

관리형 정책에 정의된 권한은 변경할 수 없습니다. AWS AWS 관리형 정책에 정의된 권한을 업데이트하는 경우 AWS 해당 업데이트는 정책이 연결된 모든 주체 ID (사용자, 그룹, 역할) 에 영향을 미칩니다. AWS 새 API 작업이 시작되거나 기존 서비스에 새 AWS 서비스 API 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 가장 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하십시오.

Amazon EC2 Auto Scaling 관리형 정책

다음과 같은 관리형 정책을 AWS Identity and Access Management (IAM) 자격 증명 (사용자 또는 역할) 에 연결할 수 있습니다. 각 정책은 Amazon EC2 Auto Scaling에 대한 API 작업 모두 또는 일부에 대한 액세스 권한을 부여합니다.

- AutoScalingConsoleFullAccess— 를 사용하여 Amazon EC2 Auto Scaling에 대한 전체 액세스 권한을 부여합니다. AWS Management Console이 정책은 시작 구성을 사용할 때 작동하고 시작 템플릿을 사용할 때는 작동하지 않습니다.
- AutoScalingConsoleReadOnlyAccess— 를 사용하여 Amazon EC2 Auto Scaling에 대한 읽기 전용 액세스 권한을 부여합니다. AWS Management Console이 정책은 시작 구성을 사용할 때 작동하고 시작 템플릿을 사용할 때는 작동하지 않습니다.
- AutoScalingFullAccess— AWS CLI 또는 SDK를 통한 전체 Amazon EC2 Auto Scaling 액세스가 필요하지만 액세스는 불가능할 IAM ID에 대해 Amazon EC2 Auto Scaling에 대한 전체 액세스 권한을 부여합니다. AWS Management Console
- AutoScalingReadOnlyAccess— 또는 SDK만 호출하는 IAM ID에 대해 Amazon EC2 Auto Scaling에 대한 읽기 전용 액세스 권한을 부여합니다. AWS CLI

콘솔에서 시작 템플릿을 사용하는 경우 시작 템플릿과 관련된 추가 권한을 부여해야 합니다. 이 내용은 Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다.에서 다룹니다. Amazon EC2 Auto Scaling 콘솔에는 `ec2` 작업에 대한 권한이 필요합니다. 그래야 시작 템플릿 및 시작 템플릿을 사용하는 시작 인스턴스에 대한 정보를 표시할 수 있습니다.

AutoScalingServiceRolePolicy AWS 관리형 정책

이 정책은 Amazon EC2 Auto Scaling이 사용자를 대신하여 작업을 수행하도록 허용하는 서비스 연결 역할에 연결됩니다. 자세한 정보는 Amazon EC2 Auto Scaling의 서비스 연결 역할을 참조하세요.

이 정책에 대한 권한을 보려면 AWS 관리형 정책 참조를 참조하십시오 AutoScalingServiceRolePolicy.

Amazon EC2 Auto Scaling에서 관리형 정책을 업데이트했습니다. AWS

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 Amazon EC2 Auto Scaling의 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인하십시오. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon EC2 Auto Scaling 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
Amazon EC2 Auto Scaling은 서비스 연결 역할에 권한을 추가합니다.	이 `AutoScalingServiceRolePolicy` 정책은 이제 VPC의 모든 보안 그룹이 검증을 개선하도록 Amazon EC2 GetSecurityGroupsForVpcAPI	2024년 2월 29일

변경 사항	설명	날짜
	작업을 호출하고, Amazon EC2 GetInstanceTypesFromInstanceRequirementsAPI 작업을 호출하여 특정 인스턴스 요구 사항 세트를 충족하는 인스턴스 유형에 대한 정보를 얻을 수 있는 권한을 부여합니다. 자세한 정보는 Amazon EC2 Auto Scaling의 서비스 연결 역할을 참조하세요.	

변경 사항	설명	날짜
Amazon EC2 Auto Scaling은 서비스 연결 역할에 권한을 추가합니다.	이제 `AutoScalingServiceRolePolicy` 정책은 VPC Lattice와의 통합에 필요한 API 작업에 액세스할 수 있는 권한을 서비스에 부여합니다. • `GetTargetGroup` 및 `ListTargetGroup` 작업. VPC Lattice 대상 그룹에 대한 정보를 검색하는 데 필요합니다. • `RegisterTargets` 및 `DeregisterTargets` 작업. VPC Lattice 대상 그룹에서 인스턴스를 등록 및 등록 취소하는 데 필요합니다. • `ListTargets` . Amazon EC2 Auto Scaling이 VPC Lattice 대상 그룹에 등록된 인스턴스의 상태 정보를 검색할 수 있도록 합니다. 자세한 내용은 Amazon EC2 Auto Scaling의 서비스 연결 역할(을)를 참조하세요.	2022년 12월 6일

변경 사항	설명	날짜
Amazon EC2 Auto Scaling은 서비스 연결 역할에 권한을 추가합니다.	시작 템플릿을 생성할 때 AWS Systems Manager 파라미터를 AMI ID의 별칭으로 사용할 수 있도록 이제 `AutoScalingServiceRolePolicy` 정책에서 AWS Systems Manager GetParametersAPI 작업을 호출할 권한을 부여합니다. 자세한 정보는 Amazon EC2 Auto Scaling의 서비스 연결 역할을 참조하세요.	2022년 3월 28일
Amazon EC2 Auto Scaling은 서비스 연결 역할에 권한을 추가합니다.	예측 확장을 지원하기 위해 이제 `AutoScalingServiceRolePolicy` 정책에 CloudWatch GetMetricDataAPI 작업을 호출할 수 있는 권한이 포함됩니다. 자세한 정보는 Amazon EC2 Auto Scaling의 서비스 연결 역할을 참조하세요.	2021년 5월 19일
Amazon EC2 Auto Scaling 변경 사항 추적 시작	Amazon EC2 Auto Scaling은 AWS 관리형 정책의 변경 사항을 추적하기 시작했습니다.	2021년 5월 19일

Amazon EC2 Auto Scaling의 서비스 연결 역할

Amazon EC2 Auto Scaling은 사용자를 대신하여 다른 AWS 서비스 를 자동으로 호출하는 데 필요한 권한에 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 IAM 역할과 직접 연결된 고유한 유형의 IAM 역할입니다. AWS 서비스

연결된 서비스만 서비스 연결 역할을 수임할 수 있으므로 서비스 연결 역할은 다른 AWS 서비스 에 권한을 위임하는 안전한 방법을 제공합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 사용을 참조하세요. 또한 서비스 연결 역할을 사용하면 모든 API 호출을 볼 수 있습니다. AWS CloudTrail이 사

용자를 대신하여 Amazon EC2 Auto Scaling이 수행하는 모든 작업을 추적할 수 있기 때문에 요구 사항 모니터링 및 감사에 유용합니다. 자세한 내용은 를 사용하여 Amazon EC2 Auto Scaling API 통화 로깅 AWS CloudTrail(을)를 참조하세요.

다음 섹션에서는 Amazon EC2 Auto Scaling 서비스 연결 역할을 생성하고 관리하는 방법에 대해 설명합니다. 먼저 IAM 자격 증명(사용자, 역할 등)이 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 권한을 구성해야 합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 사용을 참조하십시오.

내용

- 개요
- 서비스 연결 역할에 의해 부여된 권한
- Amazon EC2 Auto Scaling 서비스 연결 역할을 지원하는 리전
- 서비스 연결 역할 생성, 편집 및 삭제
  - 서비스 연결 역할 생성(자동)
  - 서비스 연결 역할 생성(수동)
  - 서비스 연결 역할 편집
  - 서비스 연결 역할 삭제

개요

Amazon EC2 Auto Scaling 서비스 연결 역할에는 두 가지 유형이 있습니다.

- 계정의 기본 서비스 연결 역할에는 이름이 지정되어 있습니다. AWSServiceRoleForAutoScaling 다른 서비스 연결 역할을 지정하지 않은 경우 이 역할이 Auto Scaling 그룹에 자동으로 할당됩니다.
- ***### ### # #### ### ## #### ## ### ## ## (#: _ mysuffix) AWSServiceRoleForAutoScaling***

사용자 지정 접미사 서비스 연결 역할의 권한은 기본 서비스 연결 역할의 권한과 동일합니다. 두 경우 모두 역할을 편집할 수 없으며 Auto Scaling 그룹에서 사용 중인 역할은 삭제할 수 없습니다. 유일한 차이점은 역할 이름 접미사입니다.

Amazon EC2 Auto Scaling에서 시작하는 인스턴스를 고객 관리 AWS Key Management Service 키로 암호화할 수 있도록 키 정책을 편집할 때 두 역할 중 하나를 지정할 수 있습니다. 그러나, 특정 고객 관리형 키에 세분화된 액세스를 제공하려는 경우 사용자 지정 접미사 서비스 연결 역할을 사용해야 합니다. 사용자 지정 접미사 서비스 연결 역할은 다음 기능을 제공합니다.

- 고객 관리형 키에 대한 향상된 제어 기능
- CloudTrail로그에서 API 호출을 수행한 Auto Scaling 그룹을 추적하는 기능

일부 사용자만 액세스할 수 있는 고객 관리형 키를 생성하는 경우 다음 단계를 통해 사용자 지정 접미사 서비스 연결 역할을 사용할 수 있습니다.

1. 사용자 지정 접미사를 사용하여 서비스 연결 역할을 생성합니다. 자세한 내용은 서비스 연결 역할 생성(수동)(을)를 참조하세요.
2. 서비스 연결 역할에 고객 관리형 키에 대한 액세스 권한을 부여합니다. 서비스 연결 역할에서 키를 사용할 수 있도록 허용하는 키 정책에 대한 자세한 정보는 암호화된 볼륨과 함께 사용하는 데 필요한 AWS KMS 키 정책(을)를 참조하세요.
3. 사용자에게 사용자가 생성한 서비스 연결 역할에 대한 액세스 권한을 부여합니다. IAM 정책 생성에 대한 자세한 정보는 어떤 서비스 연결 역할을 전달할 수 있는지 (사용) 제어할 수 있습니다. PassRole (을)를 참조하세요. 사용자가 해당 역할을 서비스에 전달할 권한 없이 서비스 연결 역할을 지정하려고 하면 오류가 발생합니다.

서비스 연결 역할에 의해 부여된 권한

Amazon EC2 Auto Scaling은 이름이 지정된 서비스 연결 역할 AWSServiceRoleForAutoScaling또는 사용자 지정 접미사 서비스 연결 역할을 사용합니다.

서비스 연결 역할은 그 역할을 위임하기 위해 다음 서비스를 신뢰합니다.

- `autoscaling.amazonaws.com`

역할 권한 정책은 Amazon EC2 Auto Scaling에서 다음 작업을 완료할 수 있도록 허용합니다. AutoScalingServiceRolePolicy

- `ec2`— EC2 인스턴스를 생성, 설명, 수정, 시작/중지 및 종료합니다.
- `iam`— 인스턴스에서 실행되는 애플리케이션이 해당 역할의 임시 자격 증명에 액세스할 수 있도록 IAM 역할을 EC2 인스턴스에 전달합니다.
- `iam`— Amazon EC2 Auto Scaling이 사용자를 대신하여 스팟 인스턴스를 시작할 수 있도록 AWSServiceRoleForEC2Spot서비스 연결 역할을 생성합니다.
- `elasticloadbalancing`— Elastic Load Balancing으로 인스턴스를 등록 및 등록 취소하고 등록된 대상의 상태를 확인합니다.

- `cloudwatch`— 규모 조정 정책에 대한 CloudWatch 경보를 생성, 설명, 수정 및 삭제하고 예측 규모 조정에 사용되는 지표를 검색합니다.
- `sns`— 인스턴스 시작 또는 종료 시 Amazon SNS에 알림을 게시합니다.
- `events`— 사용자를 대신하여 EventBridge 규칙을 생성, 설명, 업데이트 및 삭제합니다.
- `ssm`— Systems Manager 파라미터를 시작 템플릿의 AMI ID에 대한 별칭으로 사용하는 경우 파라미터 스토어에서 파라미터를 읽습니다.
- `vpc-lattice`— VPC Lattice에 인스턴스를 등록 및 등록 취소하고 등록된 대상의 상태를 확인합니다.

Amazon EC2 Auto Scaling 서비스 연결 역할을 지원하는 리전

Amazon EC2 Auto Scaling은 서비스가 제공되는 모든 지역에서 서비스 연결 역할을 사용할 수 있도록 AWS 리전 지원합니다.

서비스 연결 역할 생성, 편집 및 삭제

서비스 연결 역할 생성(자동)

사용자 지정 접미사 AWSServiceRoleForAutoScaling서비스 연결 역할을 수동으로 생성하고 그룹을 생성할 때 지정하지 않는 한, Amazon EC2 Auto Scaling은 Auto Scaling 그룹을 처음 생성할 때 서비스 연결 역할을 생성합니다.

> **Important**
>
> 서비스 연결 역할을 생성할 IAM 권한이 있어야 합니다. 그러한 권한이 없으면 자동 생성은 실패합니다. 자세한 정보는 IAM 사용 설명서의 서비스 연결 역할 권한 섹션 및 이 설명서의 서비스 연결 역할 생성(을)를 참조하세요.

Amazon EC2 Auto Scaling은 2018년 3월부터 서비스 연결 역할을 지원하기 시작했습니다. 그 전에 Auto Scaling 그룹을 만들었다면 Amazon EC2 Auto Scaling이 사용자 계정에 AWSServiceRoleForAutoScaling역할을 생성한 것입니다. 자세한 정보는 IAM 사용 설명서의 내 AWS 계정에 표시되는 새 역할을 참조하세요.

서비스 연결 역할 생성(수동)

서비스 연결 역할을 만들려면(콘솔 사용)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

2. 탐색 창에서 역할(Roles), 역할 생성(Create role)을 선택합니다.
3. 신뢰할 수 있는 엔터티 선택(Select trusted entity)에서 AWS 서비스(service)를 선택합니다.
4. Choose the service that will use this role(이 역할을 사용할 서비스 선택)에서 EC2 Auto Scaling 및 EC2 Auto Scaling 사용 사례를 선택합니다.
5. 다음: 권한, 다음: 태그, 다음 검토를 차례로 선택합니다. 참고: 생성하는 동안에는 서비스 연결 역할에 태그를 연결할 수 없습니다.
6. 리뷰 ***##### ## ### ## ## ## #### ### ## ### ##### #### #### # ## AWSServiceRoleForAutoScaling_ #### ### ### ## ### #####. AWSServiceRoleForAutoScaling***
7. (선택 사항) Role description(역할 설명)에서 서비스 연결 역할에 대한 설명을 편집합니다.
8. 역할 생성을 선택합니다.

서비스 연결 역할을 만들려면(AWS CLI)

create-service-linked-roleCLI ### #### ### _ #### ### Amazon EC2 Auto Scaling# ### ## ### #####. AWSServiceRoleForAutoScaling

```
aws iam create-service-linked-role --aws-service-name autoscaling.amazonaws.com --
custom-suffix suffix
```

이 명령의 출력에는 서비스 연결 역할의 ARN이 포함되며, 이 정보를 사용하여 고객 관리형 키에 대한 서비스 연결 역할 액세스를 허용할 수 있습니다.

```
{
    "Role": {
        "RoleId": "ABCDEF0123456789ABCDEF",
        "CreateDate": "2018-08-30T21:59:18Z",
        "RoleName": "AWSServiceRoleForAutoScaling_suffix",
        "Arn": "arn:aws:iam::123456789012:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_suffix",
        "Path": "/aws-service-role/autoscaling.amazonaws.com/",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
```

```
                    "Principal": {
                        "Service": [
                            "autoscaling.amazonaws.com"
                        ]
                    },
                    "Effect": "Allow"
                }
            ]
        }
    }
}
```

자세한 정보는 IAM 사용 설명서의 서비스 연결 역할 생성을 참조하세요.

서비스 연결 역할 편집

Amazon EC2 Auto Scaling용으로 생성된 서비스 연결 역할은 편집할 수 없습니다. 서비스 연결 역할을 생성한 후에는 역할의 이름 또는 해당 권한을 변경할 수 없습니다. 그러나, 역할의 설명은 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

서비스 연결 역할 삭제

Auto Scaling 그룹을 사용하지 않는 경우 해당 서비스 연결 역할을 삭제하는 것이 좋습니다. 역할을 삭제하면 사용되지 않거나 적극적으로 모니터링 및 유지 관리되지 않는 엔터티를 가질 수 없습니다.

먼저 관련 종속 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 따라서, 리소스에 대한 Amazon EC2 Auto Scaling 권한을 실수로 취소하는 것을 방지할 수 있습니다. 한 서비스 연결 역할을 여러 Auto Scaling 그룹에 사용하는 경우 서비스 연결 역할을 삭제하기 전에 해당 역할을 사용하는 모든 Auto Scaling 그룹을 삭제해야 합니다. 자세한 내용은 Auto Scaling 인프라 삭제(을)를 참조하세요.

IAM을 사용하여 서비스 연결 역할을 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 서비스에 연결 역할 삭제를 참조하세요.

AWSServiceRoleForAutoScaling서비스 연결 역할을 삭제하면 Auto Scaling 그룹을 생성하고 다른 서비스 연결 역할을 지정하지 않으면 Amazon EC2 Auto Scaling이 역할을 다시 생성합니다.

Amazon EC2 Auto Scaling 자격 증명 기반 정책 예제

기본적으로 새로 가입한 사용자는 아무 것도 할 수 있는 AWS 계정 권한이 없습니다. IAM 관리자는 Amazon EC2 Auto Scaling API 작업을 수행할 수 있는 IAM 자격 증명(예: 사용자 또는 역할) 권한을 부여하는 IAM 정책을 생성하고 할당해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하십시오.

다음은 권한 정책의 예입니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:UpdateAutoScalingGroup",
          "autoscaling:DeleteAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
      }
   },
   {
      "Effect": "Allow",
      "Action": "autoscaling:Describe*",
      "Resource": "*"
   }]
}
```

이 샘플 정책은 Auto Scaling 그룹이 **`purpose=testing`** 태그를 사용하는 경우에만 그룹을 생성, 업데이트, 삭제할 수 있는 권한을 부여합니다. `Describe` 작업은 리소스 수준 권한을 지원하지 않기 때문에 조건 없이 별도의 명령문에 지정해야 합니다. 시작 템플릿으로 인스턴스를 시작하려면 사용자에게도 `ec2:RunInstances` 권한이 있어야 합니다. 자세한 내용은 Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다.(을)를 참조하세요.

> **Note**
>
> IAM 자격 증명(사용자 또는 역할)이 Amazon EC2 Auto Scaling 작업을 수행할 수 있는 권한을 허용하거나 거부하는 고유한 사용자 지정 IAM 정책을 생성할 수 있습니다. 지정된 권한이 필요한 IAM 자격 증명에 이러한 사용자 지정 정책을 연결할 수 있습니다. 다음 예제에서는 몇 가지 일반적인 사용 사례의 권한을 보여 줍니다.
>
> 일부 Amazon EC2 Auto Scaling API 작업의 경우 작업을 통해 생성하거나 수정할 수 있는 특정 Auto Scaling 그룹을 정책에 포함할 수 있습니다. 개별 Auto Scaling 그룹 ARN을 지정하여 이

러한 작업에 대한 대상 리소스를 제한할 수 있습니다. 그러나, 특정 태그가 있는 Auto Scaling 그룹에서 작업을 허용(또는 거부)하는 태그 기반 정책을 사용하는 것이 좋습니다.

예

- 생성할 수 있는 Auto Scaling 그룹 크기 제어
- 사용할 수 있는 태그 키 및 태그 값 제어
- 삭제할 수 있는 Auto Scaling 그룹 제어
- 삭제할 수 있는 조정 정책 제어
- 인스턴스 새로 고침 작업에 대한 액세스 제어
- 서비스 연결 역할 생성
- 어떤 서비스 연결 역할을 전달할 수 있는지 (사용) 제어할 수 있습니다. PassRole

생성할 수 있는 Auto Scaling 그룹 크기 제어

다음 정책은 사용자에게 **environment=development** 태그가 있는 모든 Auto Scaling 그룹을 생성하고 업데이트할 수 있는 권한을 부여합니다. 단, 요청자가 최소 크기를 **1**보다 작게 지정하거나 최대 크기를 **10**보다 크게 지정하지 않아야 합니다. 가능할 때마다 태그를 사용하여 계정의 Auto Scaling 그룹에 대한 액세스 제어를 도울 수 있습니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:UpdateAutoScalingGroup"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/environment": "development" },
          "NumericGreaterThanEqualsIfExists": { "autoscaling:MinSize": 1 },
          "NumericLessThanEqualsIfExists": { "autoscaling:MaxSize": 10 }
      }
   }]
}
```

또는 태그를 사용하여 Auto Scaling 그룹에 대한 액세스를 제어하고 있지 않은 경우, ARN을 사용하여 IAM 정책이 적용되는 Auto Scaling 그룹을 식별할 수 있습니다.

Auto Scaling 그룹에는 다음 ARN이 있습니다.

```
"Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/my-asg"
```

여러 ARN을 목록에 포함시켜 지정할 수도 있습니다. `Resource` 요소에 Amazon EC2 Auto Scaling 리소스의 ARN을 지정하는 방법에 대한 자세한 내용은 Amazon EC2 Auto Scaling을 위한 정책 리소스(을)를 참조하세요.

사용할 수 있는 태그 키 및 태그 값 제어

IAM 정책에서 조건을 사용하여 Auto Scaling 그룹에 적용할 수 있는 태그 키와 태그 값을 제어할 수 있습니다. 요청자가 특정 태그를 지정한 경우에만 Auto Scaling 그룹을 생성하거나 태그를 지정할 수 있는 권한을 부여하려면 `aws:RequestTag` 조건 키를 사용하세요. 특정 태그 키만 허용하려면 `aws:TagKeys` 조건 키와 `ForAllValues` 변경자를 사용합니다.

다음 정책은 요청자에게 요청에 **`environment`** 키를 포함하는 태그를 지정하도록 요구합니다. **`"?*"`** 값은 태그 키에 대한 일부 값이 있음을 나타냅니다. 와일드카드를 사용하려면 `StringLike` 조건 연산자를 사용해야 합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "StringLike": { "aws:RequestTag/environment": "?*" }
      }
   }]
}
```

다음 정책은 요청자가 **`purpose=webserver`** 및 **`cost-center=cc123`** 태그가 있는 Auto Scaling 그룹에만 태그를 지정할 수 있도록 지정하고 **`purpose`** 및 **`cost-center`** 태그만 허용합니다(다른 태그는 지정할 수 없음).

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": {
              "aws:RequestTag/purpose": "webserver",
              "aws:RequestTag/cost-center": "cc123"
          },
          "ForAllValues:StringEquals": { "aws:TagKeys": ["purpose", "cost-center"] }
      }
   }]
}
```

다음 정책은 요청자에게 요청에 태그를 한 개 이상 지정하도록 요구하고, **`cost-center`** 키와 **`owner`** 키만 허용합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:CreateAutoScalingGroup",
          "autoscaling:CreateOrUpdateTags"
      ],
      "Resource": "*",
      "Condition": {
          "ForAnyValue:StringEquals": { "aws:TagKeys": ["cost-center", "owner"] }
      }
   }]
}
```

> **Note**
>
> 조건의 경우 조건 키는 대소문자를 구분하지 않고 조건 값은 대소문자를 구분합니다. 따라서, 태그 키의 대소문자 구별을 설정하려면 태그 키가 조건의 값으로 지정된 `aws:TagKeys` 조건 키를 사용합니다.

삭제할 수 있는 Auto Scaling 그룹 제어

다음 정책은 그룹에 **`environment=development`** 태그가 있는 경우에만 Auto Scaling 그룹의 삭제를 허용합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "*",
      "Condition": {
          "StringEquals": { "aws:ResourceTag/environment": "development" }
      }
   }]
}
```

또는 조건 키를 사용하여 Auto Scaling 그룹에 대한 액세스를 제어하고 있지 않은 경우, `Resource` 요소에서 리소스의 ARN을 지정하여 대신 액세스를 제어할 수 있습니다.

다음 정책은 사용자에게 `DeleteAutoScalingGroup` API 작업을 사용할 수 있는 권한을 부여하지만 이름이 **`devteam-`**로 시작하는 Auto Scaling 그룹에만 해당됩니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeleteAutoScalingGroup",
      "Resource": "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/devteam-*"
   }]
}
```

여러 ARN을 목록에 포함시켜 지정할 수도 있습니다. UUID를 포함하면 특정 Auto Scaling 그룹에 대한 액세스 권한을 부여할 수 있습니다. 새 그룹의 UUID는 동일한 이름을 가진 삭제된 그룹의 UUID와 다릅니다.

```
"Resource": [
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-1",
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-2",
    "arn:aws:autoscaling:region:account-
id:autoScalingGroup:uuid:autoScalingGroupName/devteam-3"
]
```

삭제할 수 있는 조정 정책 제어

다음 정책은 `DeletePolicy` 작업을 사용하여 조정 정책을 삭제할 수 있는 권한을 부여합니다. 그러나, 작업 대상 Auto Scaling 그룹에 **`environment=production`** 태그가 있는 경우 작업을 거부합니다. 가능할 때마다 태그를 사용하여 계정의 Auto Scaling 그룹에 대한 액세스 제어를 도울 수 있습니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*"
   },
   {
      "Effect": "Deny",
      "Action": "autoscaling:DeletePolicy",
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/environment": "production" }
      }
   }]
}
```

인스턴스 새로 고침 작업에 대한 액세스 제어

다음 정책은 작업 대상 Auto Scaling 그룹에 **`environment=testing`** 태그가 있는 경우에만 인스턴스 새로 고침을 시작, 롤백, 취소할 수 있는 권한을 부여합니다. `Describe` 작업은 리소스 수준 권한을 지원하지 않기 때문에 조건 없이 별도의 명령문에 지정해야 합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
          "autoscaling:StartInstanceRefresh",
          "autoscaling:CancelInstanceRefresh",
          "autoscaling:RollbackInstanceRefresh"
      ],
      "Resource": "*",
      "Condition": {
          "StringEquals": { "autoscaling:ResourceTag/environment": "testing" }
      }
   },
   {
      "Effect": "Allow",
      "Action": "autoscaling:DescribeInstanceRefreshes",
      "Resource": "*"
   }]
}
```

`StartInstanceRefresh` 호출에서 원하는 구성을 지정하려면 사용자에게 다음과 같은 몇 가지 관련 권한이 필요할 수도 있습니다.

- ec2: RunInstances — 시작 템플릿을 사용하여 EC2 인스턴스를 시작하려면 사용자에게 IAM 정책의 `ec2:RunInstances` 권한이 있어야 합니다. 자세한 정보는 Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다.을 참조하세요.
- ec2: CreateTags — 생성 시 인스턴스와 볼륨에 태그를 추가하는 시작 템플릿에서 EC2 인스턴스를 시작하려면 사용자에게 IAM 정책에 대한 `ec2:CreateTags` 권한이 있어야 합니다. 자세한 정보는 인스턴스 및 볼륨에 태그를 지정하려면 필요한 권한을 참조하세요.
- iam: PassRole — 인스턴스 프로필 (IAM 역할을 위한 컨테이너) 이 포함된 시작 템플릿에서 EC2 인스턴스를 시작하려면 사용자에게 IAM 정책에 대한 권한도 있어야 합니다. `iam:PassRole` 자세한 내용과 예 IAM 정책은 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 섹션을 참조하세요.
- ssm: GetParameters — AWS Systems Manager 파라미터를 사용하는 시작 템플릿에서 EC2 인스턴스를 시작하려면 사용자에게 IAM 정책에 대한 권한도 있어야 합니다. `ssm:GetParameters` 자세한 정보는 AMI IDs 시작 템플릿 대신 AWS Systems Manager 파라미터 사용을 참조하세요.

서비스 연결 역할 생성

Amazon EC2 Auto Scaling에는 사용자 중 누군가가 처음으로 Amazon AWS 계정 EC2 Auto Scaling API 작업을 호출할 때 서비스 연결 역할을 생성할 수 있는 권한이 필요합니다. 서비스 연결 역할이 아직 존재하지 않으면 Amazon EC2 Auto Scaling에서 해당 역할을 계정에 생성합니다. 서비스 연결 역할은 Amazon EC2 Auto Scaling에 권한을 부여하여 사용자를 대신하여 다른 사람을 AWS 서비스 호출할 수 있도록 합니다.

역할 자동 생성이 성공하려면 사용자가 `iam:CreateServiceLinkedRole` 작업에 대한 권한을 보유해야 합니다.

```
"Action": "iam:CreateServiceLinkedRole"
```

다음은 사용자가 Amazon EC2 Auto Scaling을 위한 Amazon EC2 Auto Scaling 서비스 연결 역할을 생성할 수 있도록 허용하는 권한 정책의 예입니다.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
      "Condition": {
          "StringLike": { "iam:AWSServiceName":"autoscaling.amazonaws.com" }
      }
   }]
}
```

어떤 서비스 연결 역할을 전달할 수 있는지 (사용) 제어할 수 있습니다. PassRole

Auto Scaling 그룹을 만들거나 업데이트하고 요청에 사용자 지정 접미사 서비스 연결 역할을 지정하는 사용자에게는 `iam:PassRole` 권한이 필요합니다.

서비스 연결 역할마다 다른 키에 대한 액세스 `iam:PassRole` 권한을 부여하면 권한을 사용하여 AWS KMS 고객 관리 키의 보안을 보호할 수 있습니다. 조직의 필요에 따라 개발 팀을 위한 키, 품질 보증 팀을 위한 키 및 재무 팀을 위한 키가 있을 수 있습니다. 먼저 필수 키에 액세스할 수 있는 서비스 연결 역할 (예: 이름이 지정된 서비스 연결 역할) 을 생성합니다. AWSServiceRoleForAutoScaling_devteamkeyaccess 그런 다음, 사용자 또는 역할과 같은 IAM 자격 증명에 정책을 연결합니다.

다음 정책은 이름이 **devteam-**로 시작하는 모든 Auto Scaling 그룹에 **AWSServiceRoleForAutoScaling_devteamkeyaccess** 역할을 전달할 수 있는 권한을 부여합니다. Auto Scaling 그룹을 생성하는 IAM 자격 증명이 다른 서비스 연결 역할을 지정하려고 하면 오류가 발생합니다. 서비스 연결 역할을 지정하지 않기로 선택하면 기본 역할이 대신 사용됩니다. AWSServiceRoleForAutoScaling

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::account-id:role/aws-service-role/
autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling_devteamkeyaccess",
      "Condition": {
          "StringEquals": { "iam:PassedToService": [ "autoscaling.amazonaws.com" ] },
          "StringLike": { "iam:AssociatedResourceARN":
 [ "arn:aws:autoscaling:region:account-
id:autoScalingGroup:*:autoScalingGroupName/devteam-*" ] }
      }
   }]
}
```

사용자 지정 접미사 서비스 연결 역할에 대한 자세한 정보는 Amazon EC2 Auto Scaling의 서비스 연결 역할(을)를 참조하세요.

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 AWS크로스 서비스 사칭으로 인해 대리인 문제가 발생할 수 있습니다. 교차 서비스 가장은 한 서비스(호출하는 서비스)가 다른 서비스(호출되는 서비스)를 호출할 때 발생할 수 있습니다. 직접적으로 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다.

이를 방지하기 위해 계정 내 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 통해 모든 서비스의 데이터를 보호하는 데 도움이 되는 도구를 AWS 제공합니다. Amazon EC2 Auto Scaling 서비스 역할에 대한 신뢰 정책에 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하는 것이 좋습니다. 이러한 키는 Amazon EC2 Auto Scaling이 리소스에 대해 다른 서비스에 부여하는 권한을 제한합니다.

`SourceArn`및 `SourceAccount` 필드의 값은 Amazon EC2 Auto Scaling에서 AWS Security Token Service (AWS STS) 를 사용하여 사용자를 대신하여 역할을 맡을 때 설정됩니다.

`aws:SourceArn` 또는 `aws:SourceAccount` 전역 조건 키를 사용하려면 값을 Amazon EC2 Auto Scaling이 저장하는 리소스의 Amazon 리소스 이름(ARN) 또는 계정으로 설정합니다. 가능하면 더 구체적인 `aws:SourceArn`을 사용합니다. 값을 ARN 또는 ARN 패턴(ARN의 알 수 없는 부분에 대해 와일드카드(`*`) 사용)으로 설정합니다. 리소스의 ARN을 모르면 `aws:SourceAccount`를 대신 사용합니다.

다음 예제에서는 Amazon EC2 Auto Scaling에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지하는 방법을 보여줍니다.

예제: **`aws:SourceArn`** 및 **`aws:SourceAccount`** 조건 키 사용

서비스가 사용자를 대신하여 작업을 수행하기 위해 수임한 역할을 서비스 역할이라고 합니다. Amazon이 아닌 다른 곳으로 알림을 보내는 수명 주기 후크를 생성하려는 경우 EventBridge, Amazon EC2 Auto Scaling이 사용자를 대신하여 Amazon SNS 주제 또는 Amazon SQS 대기열에 알림을 보내도록 허용하는 서비스 역할을 생성해야 합니다. 교차 서비스 액세스에 Auto Scaling 그룹을 하나만 연결하려면 서비스 역할의 신뢰 정책을 다음과 같이 지정합니다.

이 예제 신뢰 정책은 조건문을 사용하여 서비스 역할에 대한 `AssumeRole` 기능을 지정된 계정의 지정된 Auto Scaling 그룹에 영향을 주는 작업으로만 제한합니다. `aws:SourceArn` 및 `aws:SourceAccount` 조건은 독립적으로 평가됩니다. 서비스 역할을 사용하려는 모든 요청은 두 조건을 모두 충족해야 합니다.

이 정책을 사용하기 전에 리전, 계정 ID, UUID 및 그룹 이름을 계정의 유효한 값으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "autoscaling.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn":
 "arn:aws:autoscaling:region:account_id:autoScalingGroup:uuid:autoScalingGroupName/my-asg"
```

```
        },
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        }
      }
    }
}
```

이전 예제에서:

- `Principal` 요소는 서비스(`autoscaling.amazonaws.com`)의 서비스 보안 주체를 지정합니다.
- `Action` 요소는 `sts:AssumeRole` 작업을 지정합니다.
- `Condition` 요소는 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 키를 지정합니다. 소스의 ARN에는 계정 ID가 포함되어 있으므로 `aws:SourceAccount`와 함께 `aws:SourceArn`을 사용할 필요가 없습니다.

추가 정보

자세한 정보는 IAM 사용 설명서의 AWS 글로벌 조건 컨텍스트 키, 혼동된 대리자 문제 및 역할 신뢰 정책 수정(콘솔)을 참조하세요.

Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다.

Amazon EC2 Auto Scaling은 Auto Scaling 그룹에서 Amazon EC2 시작 템플릿을 사용할 수 있도록 지원합니다. 사용자가 시작 템플릿에서 Auto Scaling 그룹을 생성하도록 허용하는 것이 좋습니다. 이렇게 하면 Amazon EC2 Auto Scaling 및 Amazon EC2의 최신 기능을 사용할 수 있기 때문입니다. 예를 들어, 사용자가 혼합 인스턴스 정책을 사용하려면 시작 템플릿을 지정해야 합니다.

`AmazonEC2FullAccess` 정책을 사용하여 사용자에게 계정의 Amazon EC2 Auto Scaling 리소스, 시작 템플릿 및 기타 EC2 리소스로 작업할 수 있는 완전한 액세스 권한을 부여할 수 있습니다. 또는 이 주제에 설명된 대로 고유한 사용자 정의 IAM 정책을 생성하여 사용자에게 시작 템플릿을 사용할 수 있는 세분화된 권한을 부여할 수 있습니다.

자신의 용도에 맞춰 조정할 수 있는 정책 샘플

다음은 사용자의 용도에 맞춰 조정할 수 있는 기본 권한 정책의 예제입니다. 이 정책은 Auto Scaling 그룹이 **`purpose=testing`** 태그를 사용하는 경우에만 모든 그룹을 생성, 업데이트, 삭제할 수 있는 권한을 부여합니다. 그런 다음, 모든 `Describe` 작업에 대한 권한을 부여합니다. `Describe` 작업은 리소스 수준 권한을 지원하지 않기 때문에 조건 없이 별도의 명령문에 지정해야 합니다.

이 정책이 적용되는 IAM 자격 증명(사용자 또는 역할)은 `ec2:RunInstances` 작업을 사용할 수 있는 권한도 받았기 때문에 시작 템플릿을 사용하여 Auto Scaling 그룹을 생성하거나 업데이트할 수 있는 권한이 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:DeleteAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": { "autoscaling:ResourceTag/purpose": "testing" }
            }
         },
         {
            "Effect": "Allow",
            "Action": [
                "autoscaling:Describe*",
                "ec2:RunInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

Auto Scaling 그룹을 생성하거나 업데이트하는 사용자에게는 다음과 같은 몇 가지 관련 권한이 필요할 수 있습니다.

- ec2: CreateTags — 생성 시 인스턴스와 볼륨에 태그를 추가하려면 사용자에게 IAM 정책의 `ec2:CreateTags` 권한이 있어야 합니다. 자세한 정보는 인스턴스 및 볼륨에 태그를 지정하려면 필요한 권한을 참조하세요.
- iam: PassRole — 인스턴스 프로필 (IAM 역할의 컨테이너) 이 포함된 시작 템플릿에서 EC2 인스턴스를 시작하려면 사용자에게 IAM 정책에 대한 `iam:PassRole` 권한도 있어야 합니다. 자세한 내용과 예 IAM 정책은 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할 섹션을 참조하세요.

- ssm: GetParameters — AWS Systems Manager 파라미터를 사용하는 시작 템플릿에서 EC2 인스턴스를 시작하려면 사용자에게 IAM 정책에 대한 권한도 있어야 합니다. `ssm:GetParameters` 자세한 정보는 AMI IDs 시작 템플릿 대신 AWS Systems Manager 파라미터 사용을 참조하세요.

인스턴스를 시작할 때 완료해야 하는 작업에 대한 이러한 권한은 사용자가 Auto Scaling 그룹과 상호작용할 때 확인됩니다. 자세한 내용은 `ec2:RunInstances` 및 `iam:PassRole`에 대한 권한 검증(을)를 참조하세요.

다음 예제는 시작 템플릿 사용 시 IAM 사용자의 액세스 권한을 제어하는 데 사용할 수 있는 정책 명령문을 보여 줍니다.

주제

- 특정 태그가 있는 시작 템플릿 필요
- 시작 템플릿 및 버전 번호 필요
- 인스턴스 메타데이터 서비스 버전 2(IMDSv2) 사용 필요
- Amazon EC2 리소스에 대한 액세스 제한
- 인스턴스 및 볼륨에 태그를 지정하려면 필요한 권한
- 추가 시작 템플릿 권한
- ec2:RunInstances 및 iam:PassRole에 대한 권한 검증
- 관련 리소스

특정 태그가 있는 시작 템플릿 필요

`ec2:RunInstances` 권한을 부여할 때 사용자가 특정 태그 또는 특정 ID의 시작 템플릿만 사용하여 시작 템플릿으로 인스턴스를 시작할 때 권한을 제한할 수 있도록 지정할 수 있습니다. 또한 `RunInstances` 호출에 대한 리소스 수준 권한을 추가로 지정하여 시작 템플릿을 사용하는 모든 사용자가 인스턴스를 시작할 때 참조하고 사용할 수 있는 AMI 및 기타 리소스를 제어할 수도 있습니다.

다음 예제에서는 지정된 리전에 위치하고 **`purpose=testing`** 태그가 있는 템플릿을 시작하는 `ec2:RunInstances` 작업에 대한 권한을 제한합니다. 또한 사용자는 AMI, 인스턴스 유형, 볼륨, 키 페어, 네트워크 인터페이스, 보안 그룹 등 시작 템플릿에 지정된 리소스에 액세스할 수 있습니다.

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

```
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:account-id:launch-template/*",
            "Condition": {
                "StringEquals": { "aws:ResourceTag/purpose": "testing" }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region::image/ami-*",
                "arn:aws:ec2:region:account-id:instance/*",
                "arn:aws:ec2:region:account-id:subnet/*",
                "arn:aws:ec2:region:account-id:volume/*",
                "arn:aws:ec2:region:account-id:key-pair/*",
                "arn:aws:ec2:region:account-id:network-interface/*",
                "arn:aws:ec2:region:account-id:security-group/*"
            ]
        }
    ]
}
```

시작 템플릿과 함께 태그 기반 정책을 사용하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 IAM 권한으로 시작 템플릿에 대한 액세스 제어를 참조하십시오.

시작 템플릿 및 버전 번호 필요

또한 IAM 권한을 사용하여 Auto Scaling 그룹을 생성하거나 업데이트할 때 시작 템플릿과 시작 템플릿의 버전 번호를 지정하도록 강제할 수 있습니다.

다음 예제에서는 시작 템플릿 및 시작 템플릿 버전 번호가 지정된 경우에만 사용자가 Auto Scaling 그룹을 생성하고 업데이트할 수 있습니다. 이 정책이 적용되는 사용자가 버전 번호를 생략하고 `$Latest` 또는 `$Default` 시작 템플릿 버전을 지정하거나 시작 구성을 대신 사용하는 시도를 할 경우 작업이 실패합니다.

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

```
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": { "autoscaling:LaunchTemplateVersionSpecified": "true" }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:UpdateAutoScalingGroup"
            ],
            "Resource": "*",
            "Condition": {
                "Null": { "autoscaling:LaunchConfigurationName": "false" }
            }
        }
    ]
}
```

인스턴스 메타데이터 서비스 버전 2(IMDSv2) 사용 필요

보안을 강화하기 위해 IMDSv2가 필요한 시작 템플릿 사용을 요구하도록 사용자의 권한을 설정할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 서비스 구성을 참조하십시오.

다음 예에서는 IMDSv2(`"ec2:MetadataHttpTokens":"required"`로 표시) 사용을 요구하도록 인스턴스도 옵트인되지 않으면 사용자가 `ec2:RunInstances` 작업을 호출할 수 없도록 지정합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireImdsV2",
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringNotEquals": { "ec2:MetadataHttpTokens": "required" }
            }
```

```
        }
    ]
}
```

> **Tip**
> 새 시작 템플릿을 사용하거나 인스턴스 메타데이터 옵션이 구성된 새 버전의 시작 템플릿을 사용하는 대체 Auto Scaling 인스턴스를 강제로 시작하려면 인스턴스 새로 고침을 시작할 수 있습니다. 자세한 내용은 Auto Scaling 인스턴스 업데이트(을)를 참조하세요.

Amazon EC2 리소스에 대한 액세스 제한

다음 예제에서는 Amazon EC2 리소스에 대한 액세스를 제한하여 사용자가 시작할 수 있는 인스턴스의 구성을 제어합니다. 시작 템플릿에 지정된 리소스에 대한 리소스 수준 권한을 지정하려면 `RunInstances` 작업 명령문에 해당 리소스를 포함해야 합니다

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region:account-id:launch-template/*",
                "arn:aws:ec2:region::image/ami-04d5cc9b88example",
                "arn:aws:ec2:region:account-id:subnet/subnet-1a2b3c4d",
                "arn:aws:ec2:region:account-id:volume/*",
                "arn:aws:ec2:region:account-id:key-pair/*",
                "arn:aws:ec2:region:account-id:network-interface/*",
                "arn:aws:ec2:region:account-id:security-group/sg-903004f88example"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:region:account-id:instance/*",
            "Condition": {
                "StringEquals": { "ec2:InstanceType": ["t2.micro", "t2.small"] }
            }
        }
```

```
    ]
}
```

이 예에는 2개의 명령문이 있습니다.

- 첫 번째 명령문에서는 사용자가 인스턴스를 특정 서브넷(**subnet-1a2b3c4d**)으로 시작하여 특정 보안 그룹(**sg-903004f88example**)을 사용하고 특정 AMI(**ami-04d5cc9b88example**)를 사용해야 합니다. 또한 사용자는 네트워크 인터페이스, 키 페어, 볼륨 등 시작 템플릿에 지정된 리소스에 액세스할 수 있습니다.
- 두 번째 명령문에서는 사용자가 **t2.micro** 및 **t2.small** 인스턴스 유형만 사용하여 인스턴스를 시작하도록 허용하므로 비용 통제에 도움이 됩니다.

  하지만 시작 템플릿으로 인스턴스를 시작할 권한이 있는 사용자가 다른 인스턴스 유형을 시작하는 것을 완전히 차단할 수 있는 효과적인 방법은 현재로서는 없습니다. 이는 시작 템플릿에 지정된 인스턴스 유형이 속성 기반 인스턴스 유형 선택을 사용하여 정의된 인스턴스 유형을 사용하도록 재정의될 수 있기 때문입니다.

사용자가 시작할 수 있는 인스턴스의 구성을 제어하는 데 사용할 수 있는 리소스 수준 권한의 전체 목록은 서비스 권한 부여 참조의 Amazon EC2에 사용되는 작업, 리소스, 조건 키를 참조하세요.

인스턴스 및 볼륨에 태그를 지정하려면 필요한 권한

다음 예제에서는 사용자가 생성 시 인스턴스와 볼륨에 태그를 지정하도록 허용합니다. 이 정책은 시작 템플릿에 지정된 태그가 있는 경우에 필요합니다. 자세한 내용은 Amazon EC2 사용 설명서의 생성 시 리소스에 태그를 지정할 권한 부여를 참조하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:region:account-id:*/*",
            "Condition": {
                "StringEquals": { "ec2:CreateAction": "RunInstances" }
            }
        }
    ]
}
```

추가 시작 템플릿 권한

콘솔 사용자에게 `ec2:DescribeLaunchTemplates` 및 `ec2:DescribeLaunchTemplateVersions` 작업에 대한 권한을 부여해야 합니다. 이러한 권한이 없으면 시작 템플릿과 네트워크 옵션을 Auto Scaling 그룹 마법사에 로드할 수 없으며 사용자는 시작 템플릿을 사용하여 인스턴스를 시작하기 위해 마법사를 단계별로 실행할 수 없습니다. IAM 정책 명령문의 `Action` 요소에서 이러한 추가 작업을 지정할 수 있습니다.

`ec2:RunInstances` 및 `iam:PassRole`에 대한 권한 검증

사용자는 Auto Scaling 그룹에서 사용하는 시작 템플릿 버전을 지정할 수 있습니다. 권한에 따라 번호가 지정된 특정 버전 또는 시작 템플릿의 `$Default` 버전 또는 `$Latest` 버전이 될 수 있습니다. 후자의 경우 특히 주의해야 합니다. 이렇게 하면 제한하려는 `ec2:RunInstances` 및 `iam:PassRole`에 대한 권한이 재정의될 수 있습니다.

이 섹션에서는 Auto Scaling 그룹에서 최신 또는 기본 버전의 시작 템플릿을 사용하는 시나리오를 설명합니다.

사용자가 `CreateAutoScalingGroup`, `UpdateAutoScalingGroup` 또는 `StartInstanceRefresh` API를 호출하면 Amazon EC2 Auto Scaling은 요청을 진행하기 전에 해당 시점의 최신 버전 또는 기본 버전인 시작 템플릿 버전을 기준으로 권한을 확인합니다. 이를 통해 인스턴스 시작 시 완료해야 할 작업(예: `ec2:RunInstances` 및 `iam:PassRole` 작업)에 대한 권한을 검증합니다. 이를 위해 Amazon RunInstancesEC2 테스트 실행 호출을 실행하여 실제로 요청하지 않고도 사용자에게 작업에 필요한 권한이 있는지 확인합니다. 응답이 반환되면 Amazon EC2 Auto Scaling에서 해당 응답을 읽습니다. 사용자의 권한이 지정된 작업을 허용하지 않는 경우 Amazon EC2 Auto Scaling은 요청에 실패하고 누락된 권한에 대한 정보가 포함된 오류를 사용자에게 반환합니다.

초기 확인 및 요청이 완료되면 인스턴스를 시작할 때마다 Amazon EC2 Auto Scaling은 인스턴스가 변경되었더라도 서비스 연결 역할의 권한을 사용하여 인스턴스를 최신 버전 또는 기본 버전으로 시작합니다. 따라서, 시작 템플릿을 사용하는 사용자는 `iam:PassRole` 권한이 없더라도 잠재적으로 시작 템플릿을 업데이트하여 IAM 역할을 인스턴스에 전달할 수 있습니다.

구성 그룹에 액세스하고 `$Latest` 또는 `$Default` 버전을 사용할 수 있는 사용자를 제한하려면 `autoscaling:LaunchTemplateVersionSpecified` 조건 키를 사용합니다. 이렇게 하면 Auto Scaling 그룹은 사용자가 `CreateAutoScalingGroup` 및 `UpdateAutoScalingGroup` API를 호출할 때 번호가 지정된 특정 버전만 수락하도록 합니다. 이 조건 키를 IAM 정책에 추가하는 방법을 보여주는 예제는 시작 템플릿 및 버전 번호 필요(을)를 참조하세요.

`$Latest` 또는 `$Default` 시작 템플릿 버전을 사용하도록 구성된 Auto Scaling 그룹의 경우, 사용자가 기본 시작 템플릿 버전을 지정하도록 허용하는 `ec2:ModifyLaunchTemplate` 작업을 포함하여 시작

템플릿의 버전을 생성하고 관리할 수 있는 사람을 제한하는 것을 고려하는 것이 좋습니다. 자세한 내용은 Amazon EC2 사용 설명서의 버전 관리 권한 제어를 참조하십시오.

관련 리소스

시작 템플릿과 시작 템플릿 버전을 보고, 생성하고, 삭제할 수 있는 권한에 대한 자세한 내용은 Amazon EC2 사용 설명서의 IAM 권한으로 시작 템플릿에 대한 액세스 제어를 참조하십시오.

`RunInstances` 호출에 대한 액세스 제어에 사용할 수 있는 리소스 수준 권한에 대한 자세한 내용은 서비스 권한 부여 참조의 Amazon EC2에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할

Amazon EC2 인스턴스에서 실행되는 애플리케이션이 다른 AWS 서비스에 액세스하려면 자격 증명이 필요하며, 이러한 자격 증명을 안전하게 제공하려면 IAM 역할을 사용합니다. 이 역할은 애플리케이션이 다른 AWS 리소스에 액세스할 때 사용할 수 있는 임시 권한을 제공합니다. 역할의 권한에 따라 애플리케이션에서 수행할 수 있는 작업이 결정됩니다.

Auto Scaling 그룹 인스턴스의 경우, 시작 템플릿 또는 시작 구성을 생성하고 인스턴스와 연결할 인스턴스 프로파일을 선택해야 합니다. 인스턴스 프로파일은 인스턴스가 시작될 때 Amazon EC2가 인스턴스에 IAM 역할을 전달하도록 허용하는 IAM 역할을 위한 컨테이너입니다. 먼저, 리소스에 액세스하는 데 필요한 모든 권한을 가진 IAM 역할을 생성합니다. AWS 그런 다음, 인스턴스 프로파일을 생성하고 여기에 그 역할을 할당합니다.

> **Note**
>
> 모범 사례로, 애플리케이션에 필요한 다른 AWS 서비스 역할에 대한 최소한의 권한을 갖도록 역할을 생성하는 것이 좋습니다.

내용

- 사전 조건
- 시작 템플릿 생성
- 다음 사항도 참조하세요.

사전 조건

Amazon EC2에서 실행 중인 애플리케이션이 수임할 수 있는 IAM 역할을 생성합니다. 나중에 역할을 부여받은 애플리케이션이 필요한 API 호출을 할 수 있도록 적절한 권한을 선택하세요.

SDK AWS CLI 또는 AWS SDK 중 하나 대신 IAM 콘솔을 사용하는 경우 콘솔은 자동으로 인스턴스 프로필을 생성하여 해당하는 역할과 동일한 이름을 지정합니다.

IAM 역할을 생성하려면(콘솔)

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽의 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다.
4. 신뢰할 수 있는 엔터티 선택(Select trusted entity)에서 AWS 서비스(service)를 선택합니다.
5. 사용 사례에 대해 EC2를 선택하고 다음(Next)을 선택합니다.
6. 가능하다면, 권한 정책을 사용하기 위한 정책을 선택하거나 정책 생성을 선택하여 새 브라우저 탭을 열고 완전히 새로운 정책을 생성합니다. 자세한 내용은 IAM 사용자 설명서에서 IAM 정책 생성을 참조하세요. 정책을 생성하면 탭을 닫고 원래 탭으로 돌아갑니다. 서비스에게 부여하려는 권한 정책 옆의 확인란을 선택합니다.
7. (선택 사항) 권한 경계를 선택합니다. 이는 서비스 역할에 사용할 수 있는 고급 기능입니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
8. 다음을 선택합니다.
9. 이름 지정, 검토 및 생성(Name, review, and create) 페이지에서 역할 이름(Role name)에 이 역할의 목적을 식별하는 데 도움이 되는 역할 이름을 입력합니다. 이 이름은 AWS 계정내에서 고유해야 합니다. 다른 AWS 리소스가 역할을 참조할 수 있으므로 역할을 생성한 후에는 역할 이름을 편집할 수 없습니다.
10. 역할을 검토한 다음 역할 생성을 선택합니다.

IAM 권한

IAM 자격 증명 기반 정책을 사용하여 새 IAM 역할에 대한 액세스를 제어합니다. `iam:PassRole` 권한은 인스턴스 프로파일을 지정하는 시작 템플릿을 사용하여 Auto Scaling 그룹을 생성 또는 업데이트하는 IAM 자격 증명(사용자 또는 역할)에 필요합니다.

다음 예제 정책에서는 이름이 **`qateam-`**로 시작하는 IAM 역할만 전달할 수 있는 권한을 부여합니다.

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

```
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/qateam-*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn"
                    ]
                }
            }
        }
    ]
}
```

> **Important**
>
> Amazon EC2 Auto Scaling에서 시작 템플릿을 사용하는 Auto Scaling 그룹에 대한 `iam:PassRole` 작업 권한을 검증하는 방법에 대한 자세한 내용은 `ec2:RunInstances` 및 `iam:PassRole`에 대한 권한 검증(을)를 참조하세요.

시작 템플릿 생성

를 사용하여 시작 템플릿을 생성할 때는 고급 세부 정보 섹션의 IAM 인스턴스 프로필에서 역할을 선택합니다. AWS Management Console자세한 정보는 고급 설정을 사용하여 시작 템플릿 생성을 참조하세요.

에서 create-launch-template명령을 사용하여 시작 템플릿을 생성할 때는 다음 AWS CLI예와 같이 IAM 역할의 인스턴스 프로필 이름을 지정하십시오.

```
aws ec2 create-launch-template --launch-template-name my-lt-with-instance-profile --
version-description version1 \
--launch-template-data
 '{"ImageId":"ami-04d5cc9b88example","InstanceType":"t2.micro","IamInstanceProfile":
{"Name":"my-instance-profile"}}'
```

다음 사항도 참조하세요.

Amazon EC2에 대한 IAM 역할을 알아보고 사용하는 데 도움이 되는 자세한 정보는 다음을 참조하세요.

- Amazon EC2 사용 설명서에 나와 있는 Amazon EC2의 IAM 역할
- IAM 사용 설명서의 인스턴스 프로파일 사용 및 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여

Amazon EC2 Auto Scaling에 대한 규정 준수 검증

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의AWS 서비스 범위별, 규정 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 적합한 애플리케이션을 만드는 AWS HIPAA 데 사용할 수 있는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스 사람이 자격이 있는 것은 아닙니다. HIPAA 자세한 내용은 HIPAA적격 서비스 참조를 참조하십시오.

- AWS 규정AWS 준수 리소스 — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (국립 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (), 국제 표준화 기구 ()) 를 포함한PCI) 전반의 보안 제어에 대한 지침을 매핑합니다. ISO
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에

대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.

- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하는 PCI DSS 등 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

PCIDSS규정 준수

Amazon EC2 Auto Scaling은 판매자 또는 서비스 공급자가 신용 카드 데이터를 처리, 저장 및 전송할 수 있도록 지원하며 결제 카드 업계 (PCI) 데이터 보안 표준 () 을 준수하는 것으로 검증되었습니다. DSS AWS PCICompliance Package 사본 요청 방법을 PCI DSS 비롯한 자세한 내용은 PCIDSS레벨 1을 참조하십시오.

AWS 워크로드의 PCI DSS 규정 준수를 달성하는 방법에 대한 자세한 내용은 다음 규정 준수 가이드를 참조하십시오.

- 결제 카드 업계 데이터 보안 표준 (PCIDSS) 3.2.1 on AWS

Amazon EC2 Auto Scaling 및 인터페이스 VPC 엔드포인트

인터페이스 VPC 엔드포인트를 사용하도록 Amazon EC2 Auto Scaling을 구성하여 VPC의 보안 상태를 향상시킬 수 있습니다. 인터페이스 엔드포인트는 VPC와 Amazon EC2 Auto Scaling 간의 모든 네트워크 트래픽을 네트워크로 제한하여 Amazon EC2 Auto Scaling API에 비공개로 액세스할 수 있는 기술을 기반으로 합니다. AWS PrivateLink AWS 인터페이스 엔드포인트를 사용하면 인터넷 게이트웨이, NAT 디바이스 또는 가상 프라이빗 게이트웨이가 필요 없습니다.

반드시 구성해야 할 필요는 없지만 구성하는 것이 좋습니다. AWS PrivateLink AWS PrivateLink 및 VPC 엔드포인트에 대한 자세한 내용은 What is? 를 참조하십시오. AWS PrivateLink 가이드에서.AWS PrivateLink

주제

- 인터페이스 VPC 엔드포인트 생성
- VPC 엔드포인트 정책 생성

인터페이스 VPC 엔드포인트 생성

다음 서비스 이름을 사용하여 Amazon EC2 Auto Scaling에 대한 엔드포인트를 생성합니다.

```
com.amazonaws.region.autoscaling
```

자세한 내용은 가이드의 인터페이스 VPC 엔드포인트를 사용한 AWS 서비스 액세스를 참조하십시오.AWS PrivateLink

Amazon EC2 Auto Scaling 설정은 변경할 필요가 없습니다. Amazon EC2 Auto Scaling은 AWS 서비스 엔드포인트 또는 프라이빗 인터페이스 VPC 엔드포인트 중 사용 중인 엔드포인트를 사용하여 다른 서비스를 호출합니다.

VPC 엔드포인트 정책 생성

VPC 엔드포인트에 정책을 연결하여 Amazon EC2 Auto Scaling API에 대한 액세스를 제어할 수 있습니다. 이 정책은 다음을 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

다음 예에서는 엔드포인트를 통해 조정 정책을 삭제할 수 있는 모든 사용자 권한을 거부하는 VPC 엔드포인트 정책을 보여줍니다. 또한 이 정책 예에서는 모든 사용자에게 다른 모든 작업을 수행할 수 있는 권한을 부여합니다.

```
{
   "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "autoscaling:DeleteScalingPolicy",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": "*"
```

```
        }
    ]
}
```

자세한 정보는 AWS PrivateLink 가이드의 엔드포인트 정책을 사용하여 VPC 엔드포인트에 대한 액세스 제어를 참조하세요.

에서 이 서비스 사용 AWS SDK

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 많은 프로그래밍 언어에 사용할 수 있습니다. 각 는 개발자가 원하는 언어로 애플리케이션을 더 쉽게 빌드할 수 있도록 API, 코드 예제 및 설명서를 SDK 제공합니다.

SDK 설명서	코드 예시
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	PowerShell 코드 예제용 도구
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

이 서비스 관련 예시는 를 사용한 Auto Scaling의 코드 예제 AWS SDKs를 참조하세요.

예제 사용 가능 여부

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

를 사용한 Auto Scaling의 코드 예제 AWS SDKs

다음 코드 예제에서는 AWS 소프트웨어 개발 키트()와 함께 Auto Scaling을 사용하는 방법을 보여줍니다SDK.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작하기

Auto Scaling 시작

다음 코드 예제에서는 Auto Scaling을 사용하여 시작하는 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
```

```
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();

        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

        if (response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
        }
    }
}
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_autoscaling.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
```

```
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for C++ API

Java

SDK Java 2.x용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
```

```
        });
    }
}
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for Java 2.x API

PHP

PHP용 SDK

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for PHP API

Python

SDK Python용(Boto3)

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
 client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- API 자세한 내용은 DescribeAutoScalingGroups의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Ruby

SDK Ruby용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
```

```
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for Ruby API

Rust

SDK Rust용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
```

```
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API 자세한 내용은 Rust 참조 DescribeAutoScalingGroups 의 섹션을 참조하세요. AWS SDK API

코드 예시

- 를 사용한 Auto Scaling의 기본 예제 AWS SDKs
  - Auto Scaling 시작
  - 를 사용하여 Auto Scaling의 기본 사항 알아보기 AWS SDK
  - 를 사용한 Auto Scaling 작업 AWS SDKs
    - 또는 와 AttachInstancesAWS SDK 함께 사용 CLI
    - 또는 와 AttachLoadBalancerTargetGroupsAWS SDK 함께 사용 CLI
    - 또는 와 AttachLoadBalancersAWS SDK 함께 사용 CLI
    - 또는 와 CompleteLifecycleActionAWS SDK 함께 사용 CLI
    - 또는 와 CreateAutoScalingGroupAWS SDK 함께 사용 CLI
    - 또는 와 CreateLaunchConfigurationAWS SDK 함께 사용 CLI
    - 또는 와 CreateOrUpdateTagsAWS SDK 함께 사용 CLI
    - 또는 와 DeleteAutoScalingGroupAWS SDK 함께 사용 CLI
    - 또는 와 DeleteLaunchConfigurationAWS SDK 함께 사용 CLI
    - 또는 와 DeleteLifecycleHookAWS SDK 함께 사용 CLI
    - 또는 와 DeleteNotificationConfigurationAWS SDK 함께 사용 CLI
    - 또는 와 DeletePolicyAWS SDK 함께 사용 CLI
    - 또는 와 DeleteScheduledActionAWS SDK 함께 사용 CLI

- 또는 와 DeleteTagsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAccountLimitsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAdjustmentTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingGroupsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingInstancesAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingNotificationTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeLaunchConfigurationsAWS SDK 함께 사용 CLI
- 또는 와 DescribeLifecycleHookTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeLifecycleHooksAWS SDK 함께 사용 CLI
- 또는 와 DescribeLoadBalancersAWS SDK 함께 사용 CLI
- 또는 와 DescribeMetricCollectionTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeNotificationConfigurationsAWS SDK 함께 사용 CLI
- 또는 와 DescribePoliciesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScalingActivitiesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScalingProcessTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScheduledActionsAWS SDK 함께 사용 CLI
- 또는 와 DescribeTagsAWS SDK 함께 사용 CLI
- 또는 와 DescribeTerminationPolicyTypesAWS SDK 함께 사용 CLI
- 또는 와 DetachInstancesAWS SDK 함께 사용 CLI
- 또는 와 DetachLoadBalancersAWS SDK 함께 사용 CLI
- 또는 와 DisableMetricsCollectionAWS SDK 함께 사용 CLI
- 또는 와 EnableMetricsCollectionAWS SDK 함께 사용 CLI
- 또는 와 EnterStandbyAWS SDK 함께 사용 CLI
- 또는 와 ExecutePolicyAWS SDK 함께 사용 CLI
- 또는 와 ExitStandbyAWS SDK 함께 사용 CLI
- 또는 와 PutLifecycleHookAWS SDK 함께 사용 CLI
- 또는 와 PutNotificationConfigurationAWS SDK 함께 사용 CLI
- 또는 와 PutScalingPolicyAWS SDK 함께 사용 CLI
- 또는 와 PutScheduledUpdateGroupActionAWS SDK 함께 사용 CLI
- 또는 와 RecordLifecycleActionHeartbeatAWS SDK 함께 사용 CLI

    - 또는 와 ResumeProcessesAWS SDK 함께 사용 CLI
    - 또는 와 SetDesiredCapacityAWS SDK 함께 사용 CLI
    - 또는 와 SetInstanceHealthAWS SDK 함께 사용 CLI
    - 또는 와 SetInstanceProtectionAWS SDK 함께 사용 CLI
    - 또는 와 SuspendProcessesAWS SDK 함께 사용 CLI
    - 또는 와 TerminateInstanceInAutoScalingGroupAWS SDK 함께 사용 CLI
    - 또는 와 UpdateAutoScalingGroupAWS SDK 함께 사용 CLI
- 를 사용한 Auto Scaling 시나리오 AWS SDKs
  - 를 사용하여 탄력적 서비스 구축 및 관리 AWS SDK

를 사용한 Auto Scaling의 기본 예제 AWS SDKs

다음 코드 예제에서는 에서 Amazon EC2 Auto Scaling AWS 의 기본 사항을 사용하는 방법을 보여줍니다SDKs.

예시

- Auto Scaling 시작
- 를 사용하여 Auto Scaling의 기본 사항 알아보기 AWS SDK
- 를 사용한 Auto Scaling 작업 AWS SDKs
  - 또는 와 AttachInstancesAWS SDK 함께 사용 CLI
  - 또는 와 AttachLoadBalancerTargetGroupsAWS SDK 함께 사용 CLI
  - 또는 와 AttachLoadBalancersAWS SDK 함께 사용 CLI
  - 또는 와 CompleteLifecycleActionAWS SDK 함께 사용 CLI
  - 또는 와 CreateAutoScalingGroupAWS SDK 함께 사용 CLI
  - 또는 와 CreateLaunchConfigurationAWS SDK 함께 사용 CLI
  - 또는 와 CreateOrUpdateTagsAWS SDK 함께 사용 CLI
  - 또는 와 DeleteAutoScalingGroupAWS SDK 함께 사용 CLI
  - 또는 와 DeleteLaunchConfigurationAWS SDK 함께 사용 CLI
  - 또는 와 DeleteLifecycleHookAWS SDK 함께 사용 CLI
  - 또는 와 DeleteNotificationConfigurationAWS SDK 함께 사용 CLI
  - 또는 와 DeletePolicyAWS SDK 함께 사용 CLI

- 또는 와 DeleteScheduledActionAWS SDK 함께 사용 CLI
- 또는 와 DeleteTagsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAccountLimitsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAdjustmentTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingGroupsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingInstancesAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingNotificationTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeLaunchConfigurationsAWS SDK 함께 사용 CLI
- 또는 와 DescribeLifecycleHookTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeLifecycleHooksAWS SDK 함께 사용 CLI
- 또는 와 DescribeLoadBalancersAWS SDK 함께 사용 CLI
- 또는 와 DescribeMetricCollectionTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeNotificationConfigurationsAWS SDK 함께 사용 CLI
- 또는 와 DescribePoliciesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScalingActivitiesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScalingProcessTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScheduledActionsAWS SDK 함께 사용 CLI
- 또는 와 DescribeTagsAWS SDK 함께 사용 CLI
- 또는 와 DescribeTerminationPolicyTypesAWS SDK 함께 사용 CLI
- 또는 와 DetachInstancesAWS SDK 함께 사용 CLI
- 또는 와 DetachLoadBalancersAWS SDK 함께 사용 CLI
- 또는 와 DisableMetricsCollectionAWS SDK 함께 사용 CLI
- 또는 와 EnableMetricsCollectionAWS SDK 함께 사용 CLI
- 또는 와 EnterStandbyAWS SDK 함께 사용 CLI
- 또는 와 ExecutePolicyAWS SDK 함께 사용 CLI
- 또는 와 ExitStandbyAWS SDK 함께 사용 CLI
- 또는 와 PutLifecycleHookAWS SDK 함께 사용 CLI
- 또는 와 PutNotificationConfigurationAWS SDK 함께 사용 CLI
- 또는 와 PutScalingPolicyAWS SDK 함께 사용 CLI

- 또는 와 PutScheduledUpdateGroupActionAWS SDK 함께 사용 CLI

- 또는 와 RecordLifecycleActionHeartbeatAWS SDK 함께 사용 CLI
- 또는 와 ResumeProcessesAWS SDK 함께 사용 CLI
- 또는 와 SetDesiredCapacityAWS SDK 함께 사용 CLI
- 또는 와 SetInstanceHealthAWS SDK 함께 사용 CLI
- 또는 와 SetInstanceProtectionAWS SDK 함께 사용 CLI
- 또는 와 SuspendProcessesAWS SDK 함께 사용 CLI
- 또는 와 TerminateInstanceInAutoScalingGroupAWS SDK 함께 사용 CLI
- 또는 와 UpdateAutoScalingGroupAWS SDK 함께 사용 CLI

Auto Scaling 시작

다음 코드 예제에서는 Auto Scaling을 사용하여 시작하는 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;

public class HelloAutoScaling
{
    /// <summary>
    /// Hello Amazon EC2 Auto Scaling. List EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="args"></param>
    /// <returns>Async Task.</returns>
    static async Task Main(string[] args)
    {
        var client = new AmazonAutoScalingClient();
```

```
        Console.WriteLine("Welcome to Amazon EC2 Auto Scaling.");
        Console.WriteLine("Let's get a description of your Auto Scaling
 groups.");

        var response = await client.DescribeAutoScalingGroupsAsync();

        response.AutoScalingGroups.ForEach(autoScalingGroup =>
        {

 Console.WriteLine($"{autoScalingGroup.AutoScalingGroupName}\t{autoScalingGroup.Availabil
        });

        if (response.AutoScalingGroups.Count == 0)
        {
            Console.WriteLine("Sorry, you don't have any Amazon EC2 Auto Scaling
 groups.");
        }
    }
}
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
```

```
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                       # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_autoscaling.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 *  A "Hello Autoscaling" starter application which initializes an Amazon EC2
 Auto Scaling client and describes the
 *  Amazon EC2 Auto Scaling groups.
 *
 *  main function
 *
 *  Usage: 'hello_autoscaling'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                    autoscalingClient.DescribeAutoScalingGroups(request);

            if (outcome.IsSuccess()) {
```

```
                const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
 &autoScalingGroups =
                        outcome.GetResult().GetAutoScalingGroups();
                for (auto &group: autoScalingGroups) {
                    groupNames.push_back(group.GetAutoScalingGroupName());
                }
                nextToken = outcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
 std::endl;
        for (auto &groupName: groupNames) {
            std::cout << "AutoScaling group: " << groupName << std::endl;
        }

    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for C++ API

Java

SDK Java 2.x용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingGroups {
    public static void main(String[] args) throws InterruptedException {
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeGroups(autoScalingClient);
    }

    public static void describeGroups(AutoScalingClient autoScalingClient) {
        DescribeAutoScalingGroupsResponse response =
 autoScalingClient.describeAutoScalingGroups();
        List<AutoScalingGroup> groups = response.autoScalingGroups();
        groups.forEach(group -> {
            System.out.println("Group Name: " + group.autoScalingGroupName());
            System.out.println("Group ARN: " + group.autoScalingGroupARN());
```

```
        });
    }
}
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for Java 2.x API

PHP

PHP용 SDK

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for PHP API

Python

SDK Python용(Boto3)

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import boto3


def hello_autoscaling(autoscaling_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon EC2 Auto Scaling
 client and list
    some of the Auto Scaling groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client object.
    """
    print(
        "Hello, Amazon EC2 Auto Scaling! Let's list up to ten of you Auto Scaling
 groups:"
    )
    response = autoscaling_client.describe_auto_scaling_groups()
    groups = response.get("AutoScalingGroups", [])
    if groups:
        for group in groups:
            print(f"\t{group['AutoScalingGroupName']}:
 {group['AvailabilityZones']}")
    else:
        print("There are no Auto Scaling groups in your account.")


if __name__ == "__main__":
    hello_autoscaling(boto3.client("autoscaling"))
```

- API 자세한 내용은 DescribeAutoScalingGroups의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Ruby

SDK Ruby용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-autoscaling'
require 'logger'

# AutoScalingManager is a class responsible for managing AWS Auto Scaling
 operations
# such as listing all Auto Scaling groups in the current AWS account.
class AutoScalingManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Gets and prints a list of Auto Scaling groups for the account.
  def list_auto_scaling_groups
    paginator = @client.describe_auto_scaling_groups
    auto_scaling_groups = []
    paginator.each_page do |page|
      auto_scaling_groups.concat(page.auto_scaling_groups)
    end

    if auto_scaling_groups.empty?
      @logger.info('No Auto Scaling groups found for this account.')
    else
      auto_scaling_groups.each do |group|
        @logger.info("Auto Scaling group name: #{group.auto_scaling_group_name}")
        @logger.info("  Group ARN:             #{group.auto_scaling_group_arn}")
```

```
        @logger.info("  Min/max/desired:       #{group.min_size}/
#{group.max_size}/#{group.desired_capacity}")
        @logger.info("\n")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  autoscaling_client = Aws::AutoScaling::Client.new
  manager = AutoScalingManager.new(autoscaling_client)
  manager.list_auto_scaling_groups
end
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for Ruby API

Rust

SDK Rust용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
```

```
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API 자세한 내용은 Rust 참조 DescribeAutoScalingGroups 의 섹션을 참조하세요. AWS SDK API

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

를 사용하여 Auto Scaling의 기본 사항 알아보기 AWS SDK

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 시작 템플릿과 가용 영역으로 Amazon EC2 Auto Scaling 그룹을 생성하고 인스턴스 실행에 대한 정보를 가져옵니다.
- Amazon CloudWatch 지표 수집을 활성화합니다.
- 그룹의 원하는 용량을 업데이트하고 인스턴스가 시작될 때까지 기다립니다.
- 그룹에서 인스턴스를 종료합니다.
- 사용자 요청 및 용량 변경에 따라 발생하는 조정 활동을 나열합니다.
- CloudWatch 지표에 대한 통계를 가져온 다음 리소스를 정리합니다.

.NET

AWS SDK for .NET

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
global using Amazon.AutoScaling;
global using Amazon.AutoScaling.Model;
global using Amazon.CloudWatch;
global using AutoScalingActions;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.EC2;
using Microsoft.Extensions.Configuration;
using Host = Microsoft.Extensions.Hosting.Host;

namespace AutoScalingBasics;

public class AutoScalingBasics
{

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 Auto Scaling, Amazon
        // CloudWatch, and Amazon EC2.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonCloudWatch>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalingWrapper>()
                .AddTransient<CloudWatchWrapper>()
                .AddTransient<EC2Wrapper>()
                .AddTransient<UIWrapper>()
            )
            .Build();


        var autoScalingWrapper =
host.Services.GetRequiredService<AutoScalingWrapper>();
        var cloudWatchWrapper =
host.Services.GetRequiredService<CloudWatchWrapper>();
        var ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var imageId = configuration["ImageId"];
        var instanceType = configuration["InstanceType"];
        var launchTemplateName = configuration["LaunchTemplateName"];

        launchTemplateName += Guid.NewGuid().ToString();

        // The name of the Auto Scaling group.
        var groupName = configuration["GroupName"];

        uiWrapper.DisplayTitle("Auto Scaling Basics");
        uiWrapper.DisplayAutoScalingBasicsDescription();

        // Create the launch template and save the template Id to use when
deleting the
        // launch template at the end of the application.
        var launchTemplateId = await
ec2Wrapper.CreateLaunchTemplateAsync(imageId!, instanceType!,
launchTemplateName);
```

```
        // Confirm that the template was created by asking for a description of
it.
        await ec2Wrapper.DescribeLaunchTemplateAsync(launchTemplateName);

        uiWrapper.PressEnter();

        var availabilityZones = await ec2Wrapper.ListAvailabilityZonesAsync();

        Console.WriteLine($"Creating an Auto Scaling group named {groupName}.");
        await autoScalingWrapper.CreateAutoScalingGroupAsync(
            groupName!,
            launchTemplateName,
            availabilityZones.First().ZoneName);

        // Keep checking the details of the new group until its lifecycle state
        // is "InService".
        Console.WriteLine($"Waiting for the Auto Scaling group to be active.");

        List<AutoScalingInstanceDetails> instanceDetails;

        do
        {
            instanceDetails = await
autoScalingWrapper.DescribeAutoScalingInstancesAsync(groupName!);
        }
        while (instanceDetails.Count <= 0);

        Console.WriteLine($"Auto scaling group {groupName} successfully
created.");
        Console.WriteLine($"{instanceDetails.Count} instances were created for
the group.");

        // Display the details of the Auto Scaling group.
        instanceDetails.ForEach(detail =>
        {
            Console.WriteLine($"Group name: {detail.AutoScalingGroupName}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Metrics collection");
        Console.WriteLine($"Enable metrics collection for {groupName}");
        await autoScalingWrapper.EnableMetricsCollectionAsync(groupName!);
```

```
        // Show the metrics that are collected for the group.

        // Update the maximum size of the group to three instances.
        Console.WriteLine("--- Update the Auto Scaling group to increase max size
to 3 ---");
        int maxSize = 3;
        await autoScalingWrapper.UpdateAutoScalingGroupAsync(groupName!,
launchTemplateName, maxSize);

        Console.WriteLine("--- Describe all Auto Scaling groups to show the
current state of the group ---");
        var groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);

        uiWrapper.DisplayGroupDetails(groups!);

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Describe account limits");
        await autoScalingWrapper.DescribeAccountLimitsAsync();

        uiWrapper.WaitABit(60, "Waiting for the resources to be ready.");

        uiWrapper.DisplayTitle("Set desired capacity");
        int desiredCapacity = 2;
        await autoScalingWrapper.SetDesiredCapacityAsync(groupName!,
desiredCapacity);

        Console.WriteLine("Get the two instance Id values");

        // Empty the group before getting the details again.
        groups!.Clear();
        groups = await
autoScalingWrapper.DescribeAutoScalingGroupsAsync(groupName!);
        if (groups is not null)
        {
            foreach (AutoScalingGroup group in groups)
            {
                Console.WriteLine($"The group name is
{group.AutoScalingGroupName}");
                Console.WriteLine($"The group ARN is
{group.AutoScalingGroupARN}");
                var instances = group.Instances;
```

```
                foreach (Amazon.AutoScaling.Model.Instance instance in instances)
                {
                    Console.WriteLine($"The instance id is
{instance.InstanceId}");
                    Console.WriteLine($"The lifecycle state is
{instance.LifecycleState}");
                }
            }
        }

        uiWrapper.DisplayTitle("Scaling Activities");
        Console.WriteLine("Let's list the scaling activities that have occurred
for the group.");
        var activities = await
autoScalingWrapper.DescribeScalingActivitiesAsync(groupName!);
        if (activities is not null)
        {
            activities.ForEach(activity =>
            {
                Console.WriteLine($"The activity Id is {activity.ActivityId}");
                Console.WriteLine($"The activity details are
{activity.Details}");
            });
        }

        // Display the Amazon CloudWatch metrics that have been collected.
        var metrics = await
cloudWatchWrapper.GetCloudWatchMetricsAsync(groupName!);
        Console.WriteLine($"Metrics collected for {groupName}:");
        metrics.ForEach(metric =>
        {
            Console.Write($"Metric name: {metric.MetricName}\t");
            Console.WriteLine($"Namespace: {metric.Namespace}");
        });

        var dataPoints = await
cloudWatchWrapper.GetMetricStatisticsAsync(groupName!);
        Console.WriteLine("Details for the metrics collected:");
        dataPoints.ForEach(detail =>
        {
            Console.WriteLine(detail);
        });

        // Disable metrics collection.
```

```
        Console.WriteLine("Disabling the collection of metrics for
{groupName}.");
        var success = await
autoScalingWrapper.DisableMetricsCollectionAsync(groupName!);

        if (success)
        {
            Console.WriteLine($"Successfully stopped metrics collection for
{groupName}.");
        }
        else
        {
            Console.WriteLine($"Could not stop metrics collection for
{groupName}.");
        }

        // Terminate all instances in the group.
        uiWrapper.DisplayTitle("Terminating Auto Scaling instances");
        Console.WriteLine("Now terminating all instances in the Auto Scaling
group.");

        if (groups is not null)
        {
            groups.ForEach(group =>
            {
                // Only delete instances in the AutoScaling group we created.
                if (group.AutoScalingGroupName == groupName)
                {
                    group.Instances.ForEach(async instance =>
                    {
                        await
autoScalingWrapper.TerminateInstanceInAutoScalingGroupAsync(instance.InstanceId);
                    });
                }
            });
        }

        // After all instances are terminated, delete the group.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Deleting the Auto Scaling group.");
        await autoScalingWrapper.DeleteAutoScalingGroupAsync(groupName!);

        // Delete the launch template.
```

```
        var deletedLaunchTemplateName = await
 ec2Wrapper.DeleteLaunchTemplateAsync(launchTemplateId);

        if (deletedLaunchTemplateName == launchTemplateName)
        {
            Console.WriteLine("Successfully deleted the launch template.");
        }

        Console.WriteLine("The demo is now concluded.");
    }
}


namespace AutoScalingBasics;

/// <summary>
/// A class to provide user interface methods for the EC2 AutoScaling Basics
/// scenario.
/// </summary>
public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Describe the steps in the EC2 AutoScaling Basics scenario.
    /// </summary>
    public void DisplayAutoScalingBasicsDescription()
    {
        Console.WriteLine("This code example performs the following
 operations:");
        Console.WriteLine(" 1. Creates an Amazon EC2 launch template.");
        Console.WriteLine(" 2. Creates an Auto Scaling group.");
        Console.WriteLine(" 3. Shows the details of the new Auto Scaling group");
        Console.WriteLine("    to show that only one instance was created.");
        Console.WriteLine(" 4. Enables metrics collection.");
        Console.WriteLine(" 5. Updates the Auto Scaling group to increase the");
        Console.WriteLine("    capacity to three.");
        Console.WriteLine(" 6. Describes Auto Scaling groups again to show the");
        Console.WriteLine("    current state of the group.");
        Console.WriteLine(" 7. Changes the desired capacity of the Auto
 Scaling");
        Console.WriteLine("    group to use an additional instance.");
        Console.WriteLine(" 8. Shows that there are now instances in the
 group.");
```

```
        Console.WriteLine(" 9. Lists the scaling activities that have occurred
 for the group.");
        Console.WriteLine("10. Displays the Amazon CloudWatch metrics that
 have");
        Console.WriteLine("    been collected.");
        Console.WriteLine("11. Disables metrics collection.");
        Console.WriteLine("12. Terminates all instances in the Auto Scaling
 group.");
        Console.WriteLine("13. Deletes the Auto Scaling group.");
        Console.WriteLine("14. Deletes the Amazon EC2 launch template.");
        PressEnter();
    }

    /// <summary>
    /// Display information about the Amazon Ec2 AutoScaling groups passed
    /// in the list of AutoScalingGroup objects.
    /// </summary>
    /// <param name="groups">A list of AutoScalingGroup objects.</param>
    public void DisplayGroupDetails(List<AutoScalingGroup> groups)
    {
        if (groups is null)
            return;

        groups.ForEach(group =>
        {
            Console.WriteLine($"Group name:\t{group.AutoScalingGroupName}");
            Console.WriteLine($"Group created:\t{group.CreatedTime}");
            Console.WriteLine($"Maximum number of instances:\t{group.MaxSize}");
            Console.WriteLine($"Desired number of instances:
\t{group.DesiredCapacity}");
        });
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
```

```
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

시작 템플릿과 지표를 관리하기 위해 시나리오가 호출하는 함수를 정의합니다. 이러한 함수는 Auto Scaling , Amazon EC2및 CloudWatch 작업을 래핑합니다.

```
namespace AutoScalingActions;

using Amazon.AutoScaling;
using Amazon.AutoScaling.Model;

/// <summary>
/// A class that includes methods to perform Amazon EC2 Auto Scaling
/// actions.
/// </summary>
public class AutoScalingWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

    /// <summary>
    /// Constructor for the AutoScalingWrapper class.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected Amazon EC2 Auto Scaling
 client.</param>
    public AutoScalingWrapper(IAmazonAutoScaling amazonAutoScaling)
    {
        _amazonAutoScaling = amazonAutoScaling;
    }


    /// <summary>
    /// Create a new Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name to use for the new Auto Scaling
    /// group.</param>
    /// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
    /// launch template to use to create instances in the group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        string availabilityZone)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
```

```
            LaunchTemplateName = launchTemplateName,
        };

        var zoneList = new List<string>
            {
                availabilityZone,
            };

        var request = new CreateAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = zoneList,
            LaunchTemplate = templateSpecification,
            MaxSize = 6,
            MinSize = 1
        };

        var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve information about Amazon EC2 Auto Scaling quotas to the
    /// active AWS account.
    /// </summary>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DescribeAccountLimitsAsync()
    {
        var response = await _amazonAutoScaling.DescribeAccountLimitsAsync();
        Console.WriteLine("The maximum number of Auto Scaling groups is " +
response.MaxNumberOfAutoScalingGroups);
        Console.WriteLine("The current number of Auto Scaling groups is " +
response.NumberOfAutoScalingGroups);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }



    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
```

```
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Amazon.AutoScaling.Model.Activity>>
 DescribeScalingActivitiesAsync(
        string groupName)
    {
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        return response.Activities;
    }



    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });
```

```
        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }



    /// <summary>
    /// Retrieve a list of information about Amazon EC2 Auto Scaling groups.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling groups.</returns>
    public async Task<List<AutoScalingGroup>?> DescribeAutoScalingGroupsAsync(
        string groupName)
    {
        var groupList = new List<string>
            {
                groupName,
            };

        var request = new DescribeAutoScalingGroupsRequest
        {
            AutoScalingGroupNames = groupList,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(request);
        var groups = response.AutoScalingGroups;

        return groups;
    }


    /// <summary>
```

```
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
 _amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }


    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
 _amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }


    /// <summary>
    /// Enable the collection of metric data for an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> EnableMetricsCollectionAsync(string groupName)
    {
        var listMetrics = new List<string>
            {
                "GroupMaxSize",
            };

        var collectionRequest = new EnableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
            Metrics = listMetrics,
            Granularity = "1Minute",
        };

        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };
```

```
        var response = await
 _amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
 {desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
 _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }


    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
```

```
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
 _amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
 {groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }

}


namespace AutoScalingActions;

using Amazon.EC2;
using Amazon.EC2.Model;
```

```
public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEc2;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonEc2">The injected Amazon EC2 client.</param>
    public EC2Wrapper(IAmazonEC2 amazonEc2)
    {
        _amazonEc2 = amazonEc2;
    }

    /// <summary>
    /// Create a new Amazon EC2 launch template.
    /// </summary>
    /// <param name="imageId">The image Id to use for instances launched
    /// using the Amazon EC2 launch template.</param>
    /// <param name="instanceType">The type of EC2 instances to create.</param>
    /// <param name="launchTemplateName">The name of the launch template.</param>
    /// <returns>Returns the TemplateID of the new launch template.</returns>
    public async Task<string> CreateLaunchTemplateAsync(
        string imageId,
        string instanceType,
        string launchTemplateName)
    {
        var request = new CreateLaunchTemplateRequest
        {
            LaunchTemplateData = new RequestLaunchTemplateData
            {
                ImageId = imageId,
                InstanceType = instanceType,
            },
            LaunchTemplateName = launchTemplateName,
        };

        var response = await _amazonEc2.CreateLaunchTemplateAsync(request);

        return response.LaunchTemplate.LaunchTemplateId;
    }

    /// <summary>
    /// Delete an Amazon EC2 launch template.
```

```
    /// </summary>
    /// <param name="launchTemplateId">The TemplateId of the launch template to
    /// delete.</param>
    /// <returns>The name of the EC2 launch template that was deleted.</returns>
    public async Task<string> DeleteLaunchTemplateAsync(string launchTemplateId)
    {
        var request = new DeleteLaunchTemplateRequest
        {
            LaunchTemplateId = launchTemplateId,
        };

        var response = await _amazonEc2.DeleteLaunchTemplateAsync(request);
        return response.LaunchTemplate.LaunchTemplateName;
    }

    /// <summary>
    /// Retrieve information about an EC2 launch template.
    /// </summary>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DescribeLaunchTemplateAsync(string
 launchTemplateName)
    {
        var request = new DescribeLaunchTemplatesRequest
        {
            LaunchTemplateNames = new List<string> { launchTemplateName, },
        };

        var response = await _amazonEc2.DescribeLaunchTemplatesAsync(request);

        if (response.LaunchTemplates is not null)
        {
            response.LaunchTemplates.ForEach(template =>
            {
                Console.Write($"{template.LaunchTemplateName}\t");
                Console.WriteLine(template.LaunchTemplateId);
            });

            return true;
        }

        return false;
```

```
    }

    /// <summary>
    /// Retrieve the availability zones for the current region.
    /// </summary>
    /// <returns>A collection of availability zones.</returns>
    public async Task<List<AvailabilityZone>> ListAvailabilityZonesAsync()
    {
        var response = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());

        return response.AvailabilityZones;
    }
}


namespace AutoScalingActions;

using Amazon.CloudWatch;
using Amazon.CloudWatch.Model;

/// <summary>
/// Contains methods to access Amazon CloudWatch metrics for the
/// Amazon EC2 Auto Scaling basics scenario.
/// </summary>
public class CloudWatchWrapper
{
    private readonly IAmazonCloudWatch _amazonCloudWatch;

    /// <summary>
    /// Constructor for the CloudWatchWrapper.
    /// </summary>
    /// <param name="amazonCloudWatch">The injected CloudWatch client.</param>
    public CloudWatchWrapper(IAmazonCloudWatch amazonCloudWatch)
    {
        _amazonCloudWatch = amazonCloudWatch;
    }

    /// <summary>
    /// Retrieve the metrics information collection for the Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A list of Metrics collected for the Auto Scaling group.</
returns>
```

```
    public async Task<List<Amazon.CloudWatch.Model.Metric>>
 GetCloudWatchMetricsAsync(string groupName)
    {
        var filter = new DimensionFilter
        {
            Name = "AutoScalingGroupName",
            Value = $"{groupName}",
        };

        var request = new ListMetricsRequest
        {
            MetricName = "AutoScalingGroupName",
            Dimensions = new List<DimensionFilter> { filter },
            Namespace = "AWS/AutoScaling",
        };

        var response = await _amazonCloudWatch.ListMetricsAsync(request);

        return response.Metrics;
    }

    /// <summary>
    /// Retrieve the metric data collected for an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of data points.</returns>
    public async Task<List<Datapoint>> GetMetricStatisticsAsync(string groupName)
    {
        var metricDimensions = new List<Dimension>
            {
                new Dimension
                {
                    Name = "AutoScalingGroupName",
                    Value = $"{groupName}",
                },
            };

        // The start time will be yesterday.
        var startTime = DateTime.UtcNow.AddDays(-1);

        var request = new GetMetricStatisticsRequest
        {
            MetricName = "AutoScalingGroupName",
```

```
            Dimensions = metricDimensions,
            Namespace = "AWS/AutoScaling",
            Period = 60, // 60 seconds.
            Statistics = new List<string>() { "Minimum" },
            StartTimeUtc = startTime,
            EndTimeUtc = DateTime.UtcNow,
        };

        var response = await _amazonCloudWatch.GetMetricStatisticsAsync(request);

        return response.Datapoints;
    }

}
```

- API 자세한 내용은 AWS SDK for .NET API 참조 의 다음 주제를 참조하세요.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

C++

SDK C++용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
  \sa groupsAndInstancesScenario()
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
              << std::endl;
    std::cout
            << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
 Scaling "
            << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
              << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
            "Would you like to use an existing EC2 launch template (y/n)?  ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
                "Enter the name of the existing EC2 launch template.  ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateNames(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
                ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                      << "' exists by calling DescribeLaunchTemplate." <<
 std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template.
 "
                      << outcome.GetError().GetMessage()
```

```
                      << std::endl;
        }
    }
    else { // 2.  Or create a new EC2 launch template.
        templateName = askQuestion("Enter the name for a new EC2 launch template:
");

        Aws::EC2::Model::CreateLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
        requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

        request.SetLaunchTemplateData(requestLaunchTemplateData);

        Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
                ec2Client.CreateLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "The EC2 launch template '" << templateName << " was
created."
                      << std::endl;
        }
        else if (outcome.GetError().GetExceptionName() ==
                 "InvalidLaunchTemplateName.AlreadyExistsException") {
            std::cout << "The EC2 template '" << templateName << "' already
exists"
                      << std::endl;
        }
        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
            "Enter a name for the Auto Scaling group:  ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
```

```
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
                ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "EC2 instances can be created in the following
Availability Zones:"
                    << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "   " << i + 1 << ".  "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    int availabilityZoneChoice = askQuestionForIntRange(
            "Choose an Availability Zone:  ", 1,
            static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {
        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);
```

```
        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        std::cout << "Here is the Auto Scaling group description." << std::endl;
        if (!autoScalingGroups.empty()) {
            logAutoScalingGroupInfo(autoScalingGroups);
        }
    }
    else {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    std::cout
            << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
            << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
```

```
   bool enableMetrics = askYesNoQuestion(
           "Do you want to collect metrics about the A"
           "Auto Scaling group during this demo (y/n)?  ");
   // 7. Optionally enable metrics collection for the Auto Scaling group.
   if (enableMetrics) {
       Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
       request.SetAutoScalingGroupName(groupName);

       request.AddMetrics("GroupMinSize");
       request.AddMetrics("GroupMaxSize");
       request.AddMetrics("GroupDesiredCapacity");
       request.AddMetrics("GroupInServiceInstances");
       request.AddMetrics("GroupTotalInstances");
       request.SetGranularity("1Minute");

       Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
               autoScalingClient.EnableMetricsCollection(request);
       if (outcome.IsSuccess()) {
           std::cout << "Auto Scaling metrics have been enabled."
                     << std::endl;
       }
       else {
           std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
           return false;
       }
   }

   std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
             "' from 1 to 3." << std::endl;
   askQuestion("Press enter to continue:  ", alwaysTrueTest);
   // 8. Update the Auto Scaling group, setting a new maximum size.
   {
       Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
       request.SetAutoScalingGroupName(groupName);
       request.SetMaxSize(3);

       Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
               autoScalingClient.UpdateAutoScalingGroup(request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                    << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                    << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's update the desired capacity in '" << groupName <<
              "' from 1 to 2." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    //  9. Update the Auto Scaling group, setting a new desired capacity.
    {
        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);
```

```
        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                           autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            std::cout
                    << "Here is the current state of the group." << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                    << "No EC2 launch groups were retrieved from DescribeGroup
request."
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;

    std::cout << "Let's terminate one of the EC2 instances in " << groupName <<
"."
              << std::endl;
    std::cout << "Because the desired capacity is 2, another EC2 instance will
start "
              << "to replace the terminated EC2 instance."
```

```
              << std::endl;
   std::cout << "The currently running EC2 instances are:" << std::endl;

   if (autoScalingGroups.empty()) {
       std::cerr << "Error describing groups. No groups returned." << std::endl;
       cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
       return false;
   }

   int instanceNumber = 1;
   Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
           autoScalingGroups[0].GetInstances());
   for (const Aws::String &instanceID: instanceIDs) {
       std::cout << "   " << instanceNumber << ". " << instanceID << std::endl;
       ++instanceNumber;
   }

   instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to
stop? ",
                                           1,

static_cast<int>(instanceIDs.size()));

   // 10. Terminate an EC2 instance in the Auto Scaling group.
   {
       Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
       request.SetInstanceId(instanceIDs[instanceNumber - 1]);
       request.SetShouldDecrementDesiredCapacity(false);

       Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
               autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

       if (outcome.IsSuccess()) {
           std::cout << "Waiting for EC2 instance with ID '"
                     << instanceIDs[instanceNumber - 1] << "' to terminate..."
                     << std::endl;
       }
       else {
           std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
```

```
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    waitForInstances(groupName, autoScalingGroups, autoScalingClient);

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) <<
"\n"
              << std::endl;
    std::cout << "Let's get a report of scaling activities for EC2 launch group
'"
              << groupName << "'."
              << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);
    // 11. Get a description of activities for the Auto Scaling group.
    {
        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
"
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
    }

    if (enableMetrics) {
        if (!logAutoScalingMetrics(groupName, clientConfig)) {
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    }

    std::cout << "Let's  clean up." << std::endl;
    askQuestion("Press enter to continue:  ", alwaysTrueTest);

    // 13. Disable metrics collection if enabled.
    if (enableMetrics) {
        Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
                autoScalingClient.DisableMetricsCollection(request);

        if (outcome.IsSuccess()) {
            std::cout << "Metrics collection has been disabled." << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
```

```
            return false;
        }
    }

    return cleanupResources(groupName, templateName, autoScalingClient,
 ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
                                            const
 Aws::AutoScaling::AutoScalingClient &client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
 instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
```

```
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
            else {
                if ((count % 5) == 0) {
                    std::cout << "No instance IDs returned for group." <<
std::endl;
                }

                continue;
            }
        }

        // 6.  Check lifecycle state of the instances using
DescribeAutoScalingInstances.
        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails
&details: instancesDetails) {
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
```

```
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const
 Aws::AutoScaling::AutoScalingClient &autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client)
 {
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
                Aws::String("Delete the Auto Scaling group '") + groupName +
                "'  (y/n)?"))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                    autoScalingClient.UpdateAutoScalingGroup(request);
```

```
            if (outcome.IsSuccess()) {
                std::cout
                        << "The minimum size and desired capacity of the Auto
Scaling group "
                        << "was set to zero before terminating the instances."
                        << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                          << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }

        Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
        if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                               autoScalingClient)) {
            if (!autoScalingGroups.empty()) {
                Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                        autoScalingGroups[0].GetInstances());
                for (const Aws::String &instanceID: instanceIDs) {

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                    request.SetInstanceId(instanceID);
                    request.SetShouldDecrementDesiredCapacity(true);


Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =

autoScalingClient.TerminateInstanceInAutoScalingGroup(
                                     request);

                    if (outcome.IsSuccess()) {
                        std::cout << "Initiating termination of EC2 instance '"
                                  << instanceID << "'." << std::endl;
                    }
                    else {
                        std::cerr
                                << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                                << outcome.GetError().GetMessage() << std::endl;
                        result = false;
                    }
                }
```

```
            }

            std::cout
                    << "Waiting for the EC2 instances to terminate before
deleting the "
                    << "Auto Scaling group..." << std::endl;
            waitForInstances(groupName, autoScalingGroups, autoScalingClient);
        }

        {
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was
deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    // 15. Delete the EC2 launch template.
    if (!templateName.empty() && (askYesNoQuestion(
            Aws::String("Delete the EC2 launch template '") + templateName +
            "' (y/n)?"))) {
        Aws::EC2::Model::DeleteLaunchTemplateRequest request;
        request.SetLaunchTemplateName(templateName);

        Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
                ec2Client.DeleteLaunchTemplate(request);

        if (outcome.IsSuccess()) {
            std::cout << "EC2 launch template '" << templateName << "' was
deleted."
                      << std::endl;
```

```
        }
        else {
            std::cerr << "Error with EC2::DeleteLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

 Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                         const Aws::AutoScaling::AutoScalingClient
 &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- API 자세한 내용은 AWS SDK for C++ API 참조 의 다음 주제를 참조하세요.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Java

SDK Java 2.x용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
```

```
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC)
 where instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
```

```
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise,
an empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred
for the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling
group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " +
activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
```

```
                    .autoScalingGroupName(groupName)
                    .desiredCapacity(2)
                    .build();

            autoScalingClient.setDesiredCapacity(capacityRequest);
            System.out.println("You have set the DesiredCapacity to 2");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createAutoScalingGroup(AutoScalingClient
autoScalingClient,
            String groupName,
            String launchTemplateName,
            String vpcZoneId) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupExists(groupsRequest);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
 autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest
 describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

 .describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
 response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
 instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .maxRecords(10)
                    .build();
```

```
            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks:
" + group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void describeAccountLimits(AutoScalingClient autoScalingClient)
{
       try {
           DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
           System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
           System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
           String launchTemplateName) {
       try {
           AutoScalingWaiter waiter = autoScalingClient.waiter();
           LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                   .launchTemplateName(launchTemplateName)
                   .build();

           UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                   .maxSize(3)
                   .autoScalingGroupName(groupName)
                   .launchTemplate(templateSpecification)
                   .build();

           autoScalingClient.updateAutoScalingGroup(groupRequest);
           DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                   .autoScalingGroupNames(groupName)
                   .build();

           WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                   .waitUntilGroupInService(groupsRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
```

```
            System.out.println("You successfully updated the auto scaling group
" + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- API 자세한 내용은 AWS SDK for Java 2.x API 참조 의 다음 주제를 참조하세요.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Kotlin

SDK Kotlin용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <groupName> <launchTemplateName> <serviceLinkedRoleARN> <vpcZoneId>

    Where:
        groupName - The name of the Auto Scaling group.
        launchTemplateName - The name of the launch template.
        serviceLinkedRoleARN - The Amazon Resource Name (ARN) of the service-
linked role that the Auto Scaling group uses.
```

```
        vpcZoneId - A subnet Id for a virtual private cloud (VPC) where instances
in the Auto Scaling group can be created.
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val groupName = args[0]
    val launchTemplateName = args[1]
    val serviceLinkedRoleARN = args[2]
    val vpcZoneId = args[3]

    println("**** Create an Auto Scaling group named $groupName")
    createAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN,
vpcZoneId)

    println("Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned")
    delay(60000)

    val instanceId = getSpecificAutoScaling(groupName)
    if (instanceId.compareTo("") == 0) {
        println("Error - no instance Id value")
        exitProcess(1)
    } else {
        println("The instance Id value is $instanceId")
    }

    println("**** Describe Auto Scaling with the Id value $instanceId")
    describeAutoScalingInstance(instanceId)

    println("**** Enable metrics collection $instanceId")
    enableMetricsCollection(groupName)

    println("**** Update an Auto Scaling group to maximum size of 3")
    updateAutoScalingGroup(groupName, launchTemplateName, serviceLinkedRoleARN)

    println("**** Describe all Auto Scaling groups to show the current state of
the groups")
    describeAutoScalingGroups(groupName)

    println("**** Describe account details")
```

```
    describeAccountLimits()

    println("Wait 1 min for the resources, including the instance. Otherwise, an
 empty instance Id is returned")
    delay(60000)

    println("**** Set desired capacity to 2")
    setDesiredCapacity(groupName)

    println("**** Get the two instance Id values and state")
    getAutoScalingGroups(groupName)

    println("**** List the scaling activities that have occurred for the group")
    describeScalingActivities(groupName)

    println("**** Terminate an instance in the Auto Scaling group")
    terminateInstanceInAutoScalingGroup(instanceId)

    println("**** Stop the metrics collection")
    disableMetricsCollection(groupName)

    println("**** Delete the Auto Scaling group")
    deleteSpecificAutoScalingGroup(groupName)
}

suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}

suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
```

```
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}

suspend fun describeScalingActivities(groupName: String?) {
    val scalingActivitiesRequest =
        DescribeScalingActivitiesRequest {
            autoScalingGroupName = groupName
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeScalingActivities(scalingActivitiesRequest)
        response.activities?.forEach { activity ->
            println("The activity Id is ${activity.activityId}")
            println("The activity details are ${activity.details}")
        }
    }
}

suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
                println("The lifecycle state is " + instance.lifecycleState)
            }
```

```
        }
    }
}

suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}

suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
```

```
    }
}

suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
    }
}

suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }
```

```
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}

suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}

suspend fun getSpecificAutoScaling(groupName: String): String {
    var instanceId = ""
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")

            group.instances?.forEach { instance ->
                instanceId = instance.instanceId.toString()
            }
        }
    }
    return instanceId
}
```

```
suspend fun describeAccountLimits() {
    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAccountLimits(DescribeAccountLimitsRequest {})
        println("The max number of Auto Scaling groups is
 ${response.maxNumberOfAutoScalingGroups}")
        println("The current number of Auto Scaling groups is
 ${response.numberOfAutoScalingGroups}")
    }
}

suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}

suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API 자세한 내용은 의 다음 주제에서 AWS SDK Kotlin API 참조 를 참조하세요.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup

- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

PHP

PHP용 SDK

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace AutoScaling;

use Aws\AutoScaling\AutoScalingClient;
use Aws\CloudWatch\CloudWatchClient;
use Aws\Ec2\Ec2Client;
use AwsUtilities\AWSServiceClass;
use AwsUtilities\RunnableExample;

class GettingStartedWithAutoScaling implements RunnableExample
{
    protected Ec2Client $ec2Client;
    protected AutoScalingClient $autoScalingClient;
    protected AutoScalingService $autoScalingService;
    protected CloudWatchClient $cloudWatchClient;
    protected string $templateName;
    protected string $autoScalingGroupName;
    protected array $role;

    public function runExample()
```

```
    {
        echo("\n");
        echo("--------------------------------------\n");
        print("Welcome to the Amazon EC2 Auto Scaling getting started demo using
 PHP!\n");
        echo("--------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $this->autoScalingClient = new AutoScalingClient($clientArgs);
        $this->autoScalingService = new AutoScalingService($this-
>autoScalingClient);
        $this->cloudWatchClient = new CloudWatchClient($clientArgs);

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        /**
         * Step 0: Create an EC2 launch template that you'll use to create an
 Auto Scaling group.
         */
        $this->ec2Client = new EC2Client($clientArgs);
        $this->templateName = "example_launch_template_$uniqid";
        $instanceType = "t1.micro";
        $amiId = "ami-0ca285d4c2cda3300";
        $launchTemplate = $this->ec2Client->createLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            'LaunchTemplateData' => [
                'InstanceType' => $instanceType,
                'ImageId' => $amiId,
            ]
            ]
        );

        /**
         * Step 1: CreateAutoScalingGroup: pass it the launch template you
 created in step 0.
         */
```

```
        $availabilityZones[] = $this->ec2Client->describeAvailabilityZones([])
['AvailabilityZones'][1]['ZoneName'];

        $this->autoScalingGroupName = "demoAutoScalingGroupName_$uniqid";
        $minSize = 1;
        $maxSize = 1;
        $launchTemplateId = $launchTemplate['LaunchTemplate']
['LaunchTemplateId'];
        $this->autoScalingService->createAutoScalingGroup(
            $this->autoScalingGroupName,
            $availabilityZones,
            $minSize,
            $maxSize,
            $launchTemplateId
        );

        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);

        /**
         * Step 2: DescribeAutoScalingInstances: show that one instance has
 launched.
         */
        $instanceIds = [$autoScalingGroup['AutoScalingGroups'][0]['Instances'][0]
['InstanceId']];
        $instances = $this->autoScalingService-
>describeAutoScalingInstances($instanceIds);
        echo "The Auto Scaling group {$this->autoScalingGroupName} was created
 successfully.\n";
        echo count($instances['AutoScalingInstances']) . " instances were created
 for the group.\n";
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'] . " is the max
 number of instances for the group.\n";

        /**
         * Step 3: EnableMetricsCollection: enable all metrics or a subset.
         */
        $this->autoScalingService->enableMetricsCollection($this-
>autoScalingGroupName, "1Minute");

        /**
         * Step 4: UpdateAutoScalingGroup: update max size to 3.
```

```
         */
        echo "Updating the max number of instances to 3.\n";
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MaxSize' => 3]);

        /**
         * Step 5: DescribeAutoScalingGroups: show the current state of the
 group.
         */
        $autoScalingGroup = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        echo $autoScalingGroup['AutoScalingGroups'][0]['MaxSize'];
        echo " is the updated max number of instances for the group.\n";

        $limits = $this->autoScalingService->describeAccountLimits();
        echo "Here are your account limits:\n";
        echo "MaxNumberOfAutoScalingGroups:
 {$limits['MaxNumberOfAutoScalingGroups']}\n";
        echo "MaxNumberOfLaunchConfigurations:
 {$limits['MaxNumberOfLaunchConfigurations']}\n";
        echo "NumberOfAutoScalingGroups:
 {$limits['NumberOfAutoScalingGroups']}\n";
        echo "NumberOfLaunchConfigurations:
 {$limits['NumberOfLaunchConfigurations']}\n";

        /**
         * Step 6: SetDesiredCapacity: set desired capacity to 2.
         */
        $this->autoScalingService->setDesiredCapacity($this-
>autoScalingGroupName, 2);
        sleep(10); // Wait for the group to start processing the request.
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);

        /**
         * Step 7: DescribeAutoScalingInstances: show that two instances are
 launched.
         */
        $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
```

```
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 8: TerminateInstanceInAutoScalingGroup: terminate one of the
 instances in the group.
         */
        $this->autoScalingService-
>terminateInstanceInAutoScalingGroup($instance['InstanceId'], false);
        do {
            sleep(10);
            $instances = $this->autoScalingService-
>describeAutoScalingInstances([$instance['InstanceId']]);
        } while (count($instances['AutoScalingInstances']) > 0);
        do {
            sleep(10);
            $autoScalingGroups = $this->autoScalingService-
>describeAutoScalingGroups([$this->autoScalingGroupName]);
            $instances = $autoScalingGroups['AutoScalingGroups'][0]['Instances'];
        } while (count($instances) < 2);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        foreach ($autoScalingGroups['AutoScalingGroups'] as $autoScalingGroup) {
            echo "There is a group named:
 {$autoScalingGroup['AutoScalingGroupName']}";
            echo "with an ARN of {$autoScalingGroup['AutoScalingGroupARN']}.\n";
            foreach ($autoScalingGroup['Instances'] as $instance) {
                echo "{$autoScalingGroup['AutoScalingGroupName']} has an instance
 with id of: ";
                echo "{$instance['InstanceId']} and a lifecycle state of:
 {$instance['LifecycleState']}.\n";
            }
        }

        /**
         * Step 9: DescribeScalingActivities: list the scaling activities that
 have occurred for the group so far.
         */
```

```
        $activities = $this->autoScalingService-
>describeScalingActivities($autoScalingGroup['AutoScalingGroupName']);
        echo "We found " . count($activities['Activities']) . " activities.\n";
        foreach ($activities['Activities'] as $activity) {
            echo "{$activity['ActivityId']} - {$activity['StartTime']} -
 {$activity['Description']}\n";
        }

        /**
         * Step 10: Use the Amazon CloudWatch API to get and show some metrics
 collected for the group.
         */
        $metricsNamespace = 'AWS/AutoScaling';
        $metricsDimensions = [
            [
                'Name' => 'AutoScalingGroupName',
                'Value' => $autoScalingGroup['AutoScalingGroupName'],
            ],
        ];
        $metrics = $this->cloudWatchClient->listMetrics(
            [
            'Dimensions' => $metricsDimensions,
            'Namespace' => $metricsNamespace,
            ]
        );
        foreach ($metrics['Metrics'] as $metric) {
            $timespan = 5;
            if ($metric['MetricName'] != 'GroupTotalCapacity' &&
 $metric['MetricName'] != 'GroupMaxSize') {
                continue;
            }
            echo "Over the last $timespan minutes, {$metric['MetricName']}
 recorded:\n";
            $stats = $this->cloudWatchClient->getMetricStatistics(
                [
                'Dimensions' => $metricsDimensions,
                'EndTime' => time(),
                'StartTime' => time() - (5 * 60),
                'MetricName' => $metric['MetricName'],
                'Namespace' => $metricsNamespace,
                'Period' => 60,
                'Statistics' => ['Sum'],
                ]
            );
```

```
            foreach ($stats['Datapoints'] as $stat) {
                echo "{$stat['Timestamp']}: {$stat['Sum']}\n";
            }
        }

        return $instances;
    }

    public function cleanUp()
    {
        /**
         * Step 11: DisableMetricsCollection: disable all metrics.
         */
        $this->autoScalingService->disableMetricsCollection($this-
>autoScalingGroupName);

        /**
         * Step 12: DeleteAutoScalingGroup: to delete the group you must stop all
 instances.
         * - UpdateAutoScalingGroup with MinSize=0
         * - TerminateInstanceInAutoScalingGroup for each instance,
         *     specify ShouldDecrementDesiredCapacity=True. Wait for instances to
 stop.
         * - Now you can delete the group.
         */
        $this->autoScalingService->updateAutoScalingGroup($this-
>autoScalingGroupName, ['MinSize' => 0]);
        $this->autoScalingService->terminateAllInstancesInAutoScalingGroup($this-
>autoScalingGroupName);
        $this->autoScalingService->waitUntilGroupInService([$this-
>autoScalingGroupName]);
        $this->autoScalingService->deleteAutoScalingGroup($this-
>autoScalingGroupName);

        /**
         * Step 13: Delete launch template.
         */
        $this->ec2Client->deleteLaunchTemplate(
            [
            'LaunchTemplateName' => $this->templateName,
            ]
        );
    }
```

```
    public function helloService()
    {
        $autoScalingClient = new AutoScalingClient([
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ]);

        $groups = $autoScalingClient->describeAutoScalingGroups([]);
        var_dump($groups);
    }
}
```

- API 자세한 내용은 AWS SDK for PHP API 참조 의 다음 주제를 참조하세요.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Python

SDK Python용(Boto3)

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
def run_scenario(as_wrapper: AutoScalingWrapper, svc_helper: ServiceHelper) ->
 None:
    """
    Runs the scenario demonstrating the management of Auto Scaling groups and
 instances.

    :param as_wrapper: An instance of the AutoScalingWrapper that manages Auto
 Scaling groups.
    :param svc_helper: An instance of the ServiceHelper that interacts with AWS
 services.
    :return: None
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    logger.info("Starting the Amazon EC2 Auto Scaling demo.")

    print("-" * 88)
    print(
        "Welcome to the Amazon EC2 Auto Scaling demo for managing groups and
 instances."
    )
    print("-" * 88)

    print(
        "This example requires a launch template that specifies how to create "
        "EC2 instances. You can use an existing template or create a new one."
    )
    template_name = q.ask(
        "Enter the name of an existing launch template or press Enter to create a
 new one: "
    )
    template = None
    if template_name:
        template = svc_helper.get_template(template_name)
    if template is None:
        inst_type = "t1.micro"
        ami_id = "ami-0ca285d4c2cda3300"
        print("Let's create a launch template with the following
 specifications:")
        print(f"\tInstanceType: {inst_type}")
        print(f"\tAMI ID: {ami_id}")
        template_name = q.ask("Enter a name for the template: ", q.non_empty)
        template = svc_helper.create_template(template_name, inst_type, ami_id)
```

```
    print("-" * 88)

    print("Let's create an Auto Scaling group.")
    group_name = q.ask("Enter a name for the group: ", q.non_empty)
    zones = svc_helper.get_availability_zones()
    print("EC2 instances can be created in the following Availability Zones:")
    for index, zone in enumerate(zones):
        print(f"\t{index+1}. {zone}")
    print(f"\t{len(zones)+1}. All zones")
    zone_sel = q.ask(
        "Which zone do you want to use? ", q.is_int, q.in_range(1, len(zones) +
1)
    )
    group_zones = [zones[zone_sel - 1]] if zone_sel <= len(zones) else zones
    print(f"Creating group {group_name}...")
    as_wrapper.create_autoscaling_group(group_name, group_zones, template_name,
1, 1)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Created Auto Scaling group %s.", group_name)
    print("Created group:")
    pp(group)
    print("Waiting for instance to start...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    use_metrics = q.ask(
        "Do you want to collect metrics about Amazon EC2 Auto Scaling during this
demo (y/n)? ",
        q.is_yesno,
    )
    if use_metrics:
        as_wrapper.enable_metrics(
            group_name,
            [
                "GroupMinSize",
                "GroupMaxSize",
                "GroupDesiredCapacity",
                "GroupInServiceInstances",
                "GroupTotalInstances",
            ],
        )
        logger.info("Enabled metrics for Auto Scaling group %s.", group_name)
        print(f"Metrics enabled for {group_name}.")
```

```
    print("-" * 88)

    print(f"Let's update the maximum number of instances in {group_name} from 1
to 3.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.update_group(group_name, MaxSize=3)
    group = as_wrapper.describe_group(group_name)
    logger.info("Updated maximum size for group %s to 3.", group_name)
    print("The group still has one running instance, but can have up to three:")
    print_simplified_group(group)
    print("-" * 88)

    print(f"Let's update the desired capacity of {group_name} from 1 to 2.")
    q.ask("Press Enter when you're ready.")
    as_wrapper.set_desired_capacity(group_name, 2)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info("Set desired capacity for group %s to 2.", group_name)
    print("Here's the current state of the group:")
    print_simplified_group(group)
    print("-" * 88)
    print("Waiting for the new instance to start...")
    instance_ids = wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's terminate one of the instances in {group_name}.")
    print("Because the desired capacity is 2, another instance will start.")
    print("The currently running instances are:")
    for index, inst_id in enumerate(instance_ids):
        print(f"\t{index+1}. {inst_id}")
    inst_sel = q.ask(
        "Which instance do you want to stop? ",
        q.is_int,
        q.in_range(1, len(instance_ids) + 1),
    )
    print(f"Stopping {instance_ids[inst_sel-1]}...")
    as_wrapper.terminate_instance(instance_ids[inst_sel - 1], False)
    wait(10)
    group = as_wrapper.describe_group(group_name)
    logger.info(
        "Terminated instance %s in group %s.", instance_ids[inst_sel - 1],
group_name
    )
    print(f"Here's the state of {group_name}:")
```

```
    print_simplified_group(group)
    print("Waiting for the scaling activities to complete...")
    wait_for_group(group_name, as_wrapper)
    print("-" * 88)

    print(f"Let's get a report of scaling activities for {group_name}.")
    q.ask("Press Enter when you're ready.")
    activities = as_wrapper.describe_scaling_activities(group_name)
    logger.info(
        "Retrieved %d scaling activities for group %s.", len(activities),
group_name
    )
    print(
        f"Found {len(activities)} activities.\n"
        f"Activities are ordered with the most recent one first:"
    )
    for act in activities:
        pp(act)
    print("-" * 88)

    if use_metrics:
        print("Let's look at CloudWatch metrics.")
        metric_namespace = "AWS/AutoScaling"
        metric_dimensions = [{"Name": "AutoScalingGroupName", "Value":
group_name}]
        print(f"The following metrics are enabled for {group_name}:")
        done = False
        while not done:
            metrics = svc_helper.get_metrics(metric_namespace, metric_dimensions)
            for index, metric in enumerate(metrics):
                print(f"\t{index+1}. {metric.name}")
            print(f"\t{len(metrics)+1}. None")
            metric_sel = q.ask(
                "Which metric do you want to see? ",
                q.is_int,
                q.in_range(1, len(metrics) + 1),
            )
            if metric_sel < len(metrics) + 1:
                span = 5
                metric = metrics[metric_sel - 1]
                print(f"Over the last {span} minutes, {metric.name} recorded:")
                # CloudWatch metric times are in the UTC+0 time zone.
                now = datetime.now(timezone.utc)
                metric_data = svc_helper.get_metric_statistics(
```

```
                    metric_dimensions, metric, now - timedelta(minutes=span), now
                )
                pp(metric_data)
                if not q.ask("Do you want to see another metric (y/n)? ",
q.is_yesno):
                    done = True
            else:
                done = True

    print(f"Let's clean up.")
    q.ask("Press Enter when you're ready.")
    if use_metrics:
        print(f"Stopping metrics collection for {group_name}.")
        as_wrapper.disable_metrics(group_name)
        logger.info("Disabled metrics collection for group %s.", group_name)

    print(
        "You must terminate all instances in the group before you can delete the
group."
    )
    print("Set minimum size to 0.")
    as_wrapper.update_group(group_name, MinSize=0)
    group = as_wrapper.describe_group(group_name)
    instance_ids = [inst["InstanceId"] for inst in group["Instances"]]
    for inst_id in instance_ids:
        print(f"Stopping {inst_id}.")
        as_wrapper.terminate_instance(inst_id, True)
        logger.info("Terminated instance %s in group %s.", inst_id, group_name)
    print("Waiting for instances to stop...")
    wait_for_instances(instance_ids, as_wrapper)
    print(f"Deleting {group_name}.")
    as_wrapper.delete_autoscaling_group(group_name)
    logger.info("Deleted Auto Scaling group %s.", group_name)
    print("-" * 88)

    if template is not None:
        if q.ask(
            f"Do you want to delete launch template {template_name} used in this
demo (y/n)? "
        ):
            svc_helper.delete_template(template_name)
            logger.info("Deleted launch template %s.", template_name)
            print("Template deleted.")
```

```
    print("\nThanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    try:
        wrapper = AutoScalingWrapper(boto3.client("autoscaling"))
        helper = ServiceHelper(boto3.client("ec2"), boto3.resource("cloudwatch"))
        run_scenario(wrapper, helper)
    except Exception:
        logger.exception("Something went wrong with the demo!")
```

시작 템플릿과 지표를 관리하기 위해 시나리오가 호출하는 함수를 정의합니다. 이러한 함수는 Amazon EC2 및 CloudWatch 작업을 래핑합니다.

```
class ServiceHelper:
    """Encapsulates Amazon EC2 and CloudWatch actions for the example."""

    def __init__(self, ec2_client, cloudwatch_resource):
        """
        :param ec2_client: A Boto3 Amazon EC2 client.
        :param cloudwatch_resource: A Boto3 CloudWatch resource.
        """
        self.ec2_client = ec2_client
        self.cloudwatch_resource = cloudwatch_resource

    def get_template(self, template_name: str) -> dict:
        """
        Gets a launch template. Launch templates specify configuration for
 instances
        that are launched by Amazon EC2 Auto Scaling.

        :param template_name: The name of the template to look up.
        :return: The template, if it exists.
        :raises ClientError: If there is an error retrieving the launch template.
        """
        try:
            response = self.ec2_client.describe_launch_templates(
                LaunchTemplateNames=[template_name]
            )
            template = response["LaunchTemplates"][0]
```

```
            logger.info("Launch template %s retrieved successfully.",
template_name)
            return template
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                logger.warning("Launch template %s does not exist.",
template_name)
            else:
                logger.error(
                    "Couldn't verify launch template %s. Error: %s: %s",
                    template_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise

    def create_template(self, template_name: str, inst_type: str, ami_id: str) ->
dict:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.

        :param template_name: The name to give to the template.
        :param inst_type: The type of the instance, such as t1.micro.
        :param ami_id: The ID of the Amazon Machine Image (AMI) to use when
creating
                       an instance.
        :return: Information about the newly created template.
        :raises ClientError: If there is an error creating the launch template.
        """
        try:
            response = self.ec2_client.create_launch_template(
                LaunchTemplateName=template_name,
                LaunchTemplateData={"InstanceType": inst_type, "ImageId":
ami_id},
            )
            template = response["LaunchTemplate"]
            logger.info(
                "Created launch template %s with instance type %s and AMI ID
%s.",
                template_name,
```

```
                inst_type,
                ami_id,
            )
            return template
        except ClientError as err:
            logger.error(
                "Couldn't create launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def delete_template(self, template_name: str) -> None:
        """
        Deletes a launch template.

        :param template_name: The name of the template to delete.
        :raises ClientError: If there is an error deleting the launch template.
        """
        try:

self.ec2_client.delete_launch_template(LaunchTemplateName=template_name)
            logger.info("Deleted launch template %s.", template_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete launch template %s. Error: %s: %s",
                template_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_availability_zones(self) -> list:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        :raises ClientError: If there is an error retrieving availability zones.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
```

```
            logger.info("Retrieved availability zones: %s.", ", ".join(zones))
            return zones
        except ClientError as err:
            logger.error(
                "Couldn't get availability zones. Error: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    def get_metrics(self, namespace: str, dimensions: list) -> list:
        """
        Gets a list of CloudWatch metrics filtered by namespace and dimensions.

        :param namespace: The namespace of the metrics to look up.
        :param dimensions: The dimensions of the metrics to look up.
        :return: The list of metrics.
        :raises ClientError: If there is an error retrieving CloudWatch metrics.
        """
        try:
            metrics = list(
                self.cloudwatch_resource.metrics.filter(
                    Namespace=namespace, Dimensions=dimensions
                )
            )
            logger.info(
                "Retrieved metrics for namespace %s with dimensions %s.",
                namespace,
                dimensions,
            )
            return metrics
        except ClientError as err:
            logger.error(
                "Couldn't get metrics for %s, %s. Error: %s: %s",
                namespace,
                dimensions,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

    @staticmethod
    def get_metric_statistics(
        dimensions: list, metric, start: datetime, end: datetime
```

```
    ) -> list:
        """
        Gets statistics for a CloudWatch metric within a specified time span.

        :param dimensions: The dimensions of the metric.
        :param metric: The metric to look up.
        :param start: The start of the time span for retrieved metrics.
        :param end: The end of the time span for retrieved metrics.
        :return: The list of data points found for the specified metric.
        :raises ClientError: If there is an error retrieving metric statistics.
        """
        try:
            response = metric.get_statistics(
                Dimensions=dimensions,
                StartTime=start,
                EndTime=end,
                Period=60,
                Statistics=["Sum"],
            )
            data = response["Datapoints"]
            logger.info("Retrieved statistics for metric %s.", metric.name)
            return data
        except ClientError as err:
            logger.error(
                "Couldn't get statistics for metric %s. Error: %s: %s",
                metric.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


def print_simplified_group(group: dict) -> None:
    """
    Prints a subset of data for an Auto Scaling group.

    :param group: The Auto Scaling group data to print.
    :return: None
    """
    print(group["AutoScalingGroupName"])
    print(f"\tLaunch template: {group['LaunchTemplate']['LaunchTemplateName']}")
    print(
        f"\tMin: {group['MinSize']}, Max: {group['MaxSize']}, Desired:
 {group['DesiredCapacity']}"
```

```
    )
    if group["Instances"]:
        print(f"\tInstances:")
        for inst in group["Instances"]:
            print(f"\t\t{inst['InstanceId']}: {inst['LifecycleState']}")


def wait_for_group(group_name: str, as_wrapper: AutoScalingWrapper) -> list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param group_name: The name of the Auto Scaling group.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs in the group.
    """
    group = as_wrapper.describe_group(group_name)
    instance_ids = [i["InstanceId"] for i in group["Instances"]]
    return wait_for_instances(instance_ids, as_wrapper)


def wait_for_instances(instance_ids: list, as_wrapper: AutoScalingWrapper) ->
 list:
    """
    Waits for instances to start or stop in an Auto Scaling group.
    Prints the data for each instance after scaling activities are complete.

    :param instance_ids: A list of instance IDs to wait for.
    :param as_wrapper: The AutoScalingWrapper that manages Auto Scaling groups.
    :return: A list of instance IDs that were waited on.
    """
    ready = False
    instances = []
    while not ready:
        instances = as_wrapper.describe_instances(instance_ids) if instance_ids
 else []
        if all([x["LifecycleState"] in ["Terminated", "InService"] for x in
 instances]):
            ready = True
        else:
            wait(10)
    if instances:
        print(
```

```
            f"Here are the details of the instance{'s' if len(instances) > 1 else
 ''}:"
        )
        for instance in instances:
            pp(instance)
    return instance_ids
```

- API 자세한 내용은 의 AWS SDK Python(Boto3) API 참조에 대한 다음 주제를 참조하세요.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Rust

SDK Rust용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
[package]
name = "autoscaling-code-examples"
version = "0.1.0"
authors = ["Doug Schwartz <dougsch@amazon.com>", "David Souther
 <dpsouth@amazon.com>"]
```

```
edition = "2021"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-sdk-autoscaling = { version = "1.3.0" }
aws-sdk-ec2 = { version = "1.3.0" }
aws-types = { version = "1.0.1" }
tokio = { version = "1.20.1", features = ["full"] }
clap = { version = "~4.4", features = ["derive"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
anyhow = "1.0.75"
tracing = "0.1.37"
tokio-stream = "0.1.14"


use std::{collections::BTreeSet, fmt::Display};

use anyhow::anyhow;
use autoscaling_code_examples::scenario::{AutoScalingScenario, ScenarioError};
use tracing::{info, warn};

async fn show_scenario_description(scenario: &AutoScalingScenario, event: &str) {
    let description = scenario.describe_scenario().await;
    info!("DescribeAutoScalingInstances: {event}\n{description}");
}

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    pub fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    pub fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}
```

```
impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();

    let shared_config = aws_config::from_env().load().await;

    let mut warnings = Warnings::default();

    // 1. Create an EC2 launch template that you'll use to create an auto scaling
 group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
    // 2. CreateAutoScalingGroup: pass it the launch template you created in step
 0. Give it min/max of 1 instance.
    // 4. EnableMetricsCollection: enable all metrics or a subset.
    let scenario = match
 AutoScalingScenario::prepare_scenario(&shared_config).await {
        Ok(scenario) => scenario,
        Err(errs) => {
            let err_str = errs
                .into_iter()
                .map(|e| e.to_string())
                .collect::<Vec<String>>()
                .join(", ");
            return Err(anyhow!("Failed to initialize scenario: {err_str}"));
        }
    };

    info!("Prepared autoscaling scenario:\n{scenario}");

    let stable = scenario.wait_for_stable(1).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
```

```
        );
    }

    // 3. DescribeAutoScalingInstances: show that one instance has launched.
    show_scenario_description(
        &scenario,
        "show that the group was created and one instance has launched",
    )
    .await;

    // 5. UpdateAutoScalingGroup: update max size to 3.
    let scale_max_size = scenario.scale_max_size(3).await;
    if let Err(err) = scale_max_size {
        warnings.push("There was a problem scaling max size", err);
    }

    // 6. DescribeAutoScalingGroups: the current state of the group
    show_scenario_description(
        &scenario,
        "show the current state of the group after setting max size",
    )
    .await;

    // 7. SetDesiredCapacity: set desired capacity to 2.
    let scale_desired_capacity = scenario.scale_desired_capacity(2).await;
    if let Err(err) = scale_desired_capacity {
        warnings.push("There was a problem setting desired capacity", err);
    }

    //   Wait for a second instance to launch.
    let stable = scenario.wait_for_stable(2).await;
    if let Err(err) = stable {
        warnings.push(
            "There was a problem while waiting for group to be stable",
            err,
        );
    }

    // 8. DescribeAutoScalingInstances: show that two instances are launched.
    show_scenario_description(
        &scenario,
        "show that two instances are launched after setting desired capacity",
    )
    .await;
```

```
    let ids_before = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    // 9. TerminateInstanceInAutoScalingGroup: terminate one of the instances in
the group.
    let terminate_some_instance = scenario.terminate_some_instance().await;
    if let Err(err) = terminate_some_instance {
        warnings.push("There was a problem replacing an instance", err);
    }

    let wait_after_terminate = scenario.wait_for_stable(1).await;
    if let Err(err) = wait_after_terminate {
        warnings.push(
            "There was a problem waiting after terminating an instance",
            err,
        );
    }

    let wait_scale_up_after_terminate = scenario.wait_for_stable(2).await;
    if let Err(err) = wait_scale_up_after_terminate {
        warnings.push(
            "There was a problem waiting for scale up after terminating an
instance",
            err,
        );
    }

    let ids_after = scenario
        .list_instances()
        .await
        .map(|v| v.into_iter().collect::<BTreeSet<_>>())
        .unwrap_or_default();

    let difference = ids_after.intersection(&ids_before).count();
    if !(difference == 1 && ids_before.len() == 2 && ids_after.len() == 2) {
        warnings.push(
            "Before and after set not different",
            ScenarioError::with(format!("{difference}")),
        );
    }
```

```
    // 10. DescribeScalingActivities: list the scaling activities that have
 occurred for the group so far.
    show_scenario_description(
        &scenario,
        "list the scaling activities that have occurred for the group so far",
    )
    .await;

    // 11. DisableMetricsCollection
    let scale_group = scenario.scale_group_to_zero().await;
    if let Err(err) = scale_group {
        warnings.push("There was a problem scaling the group to 0", err);
    }
    show_scenario_description(&scenario, "Scenario scaled to 0").await;

    // 12. DeleteAutoScalingGroup (to delete the group you must stop all
 instances):
    // 13. Delete LaunchTemplate.
    let clean_scenario = scenario.clean_scenario().await;
    if let Err(errs) = clean_scenario {
        for err in errs {
            warnings.push("There was a problem cleaning the scenario", err);
        }
    } else {
        info!("The scenario has been cleaned up!");
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        Err(anyhow!(
            "There were warnings during scenario execution:\n{warnings}"
        ))
    }
}

pub mod scenario;


use std::{
    error::Error,
    fmt::{Debug, Display},
    time::{Duration, SystemTime},
```

```
};

use anyhow::anyhow;
use aws_config::SdkConfig;
use aws_sdk_autoscaling::{
    error::{DisplayErrorContext, ProvideErrorMetadata},
    types::{Activity, AutoScalingGroup, LaunchTemplateSpecification},
};
use aws_sdk_ec2::types::RequestLaunchTemplateData;
use tracing::trace;

const LAUNCH_TEMPLATE_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_template_from_Rust_SDK";
const AUTOSCALING_GROUP_NAME: &str =
 "SDK_Code_Examples_EC2_Autoscaling_Group_from_Rust_SDK";
const MAX_WAIT: Duration = Duration::from_secs(5 * 60); // Wait at most 25
 seconds.
const WAIT_TIME: Duration = Duration::from_millis(500); // Wait half a second at
 a time.

struct Waiter {
    start: SystemTime,
    max: Duration,
}

impl Waiter {
    fn new() -> Self {
        Waiter {
            start: SystemTime::now(),
            max: MAX_WAIT,
        }
    }

    async fn sleep(&self) -> Result<(), ScenarioError> {
        if SystemTime::now()
            .duration_since(self.start)
            .unwrap_or(Duration::MAX)
            > self.max
        {
            Err(ScenarioError::with(
                "Exceeded maximum wait duration for stable group",
            ))
        } else {
            tokio::time::sleep(WAIT_TIME).await;
```

```
            Ok(())
        }
    }
}

pub struct AutoScalingScenario {
    ec2: aws_sdk_ec2::Client,
    autoscaling: aws_sdk_autoscaling::Client,
    launch_template_arn: String,
    auto_scaling_group_name: String,
}

impl Display for AutoScalingScenario {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        f.write_fmt(format_args!(
            "\tLaunch Template ID: {}\n",
            self.launch_template_arn
        ))?;
        f.write_fmt(format_args!(
            "\tScaling Group Name: {}\n",
            self.auto_scaling_group_name
        ))?;

        Ok(())
    }
}

pub struct AutoScalingScenarioDescription {
    group: Result<Vec<String>, ScenarioError>,
    instances: Result<Vec<String>, anyhow::Error>,
    activities: Result<Vec<Activity>, anyhow::Error>,
}

impl Display for AutoScalingScenarioDescription {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "\t      Group status:")?;
        match &self.group {
            Ok(groups) => {
                for status in groups {
                    writeln!(f, "\t\t- {status}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! - {e}")?,
        }
```

```
        writeln!(f, "\t         Instances:")?;
        match &self.instances {
            Ok(instances) => {
                for instance in instances {
                    writeln!(f, "\t\t- {instance}")?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        writeln!(f, "\t        Activities:")?;
        match &self.activities {
            Ok(activities) => {
                for activity in activities {
                    writeln!(
                        f,
                        "\t\t- {} Progress: {}% Status: {:?} End: {:?}",
                        activity.cause().unwrap_or("Unknown"),
                        activity.progress.unwrap_or(-1),
                        activity.status_code(),
                        // activity.status_message().unwrap_or_default()
                        activity.end_time(),
                    )?;
                }
            }
            Err(e) => writeln!(f, "\t\t! {e}")?,
        }

        Ok(())
    }
}

#[derive(Debug)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(|s| s.to_string()),
            code: err.code().map(|s| s.to_string()),
        }
```

```
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug)]
pub struct ScenarioError {
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl Error for ScenarioError {
    // While `Error` can capture `source` information about the underlying error,
 for this example
    // the ScenarioError captures the underlying information in MetadataError and
 treats it as a
```

```
    // single Error from this Crate. In other contexts, it may be appropriate to
 model the error
    // as including the SdkError as its source.
}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

impl AutoScalingScenario {
    pub async fn prepare_scenario(sdk_config: &SdkConfig) -> Result<Self,
 Vec<ScenarioError>> {
        let ec2 = aws_sdk_ec2::Client::new(sdk_config);
        let autoscaling = aws_sdk_autoscaling::Client::new(sdk_config);

        let auto_scaling_group_name = String::from(AUTOSCALING_GROUP_NAME);

        // Before creating any resources, prepare the list of AZs
        let availablity_zones = ec2.describe_availability_zones().send().await;
        if let Err(err) = availablity_zones {
            return Err(vec![ScenarioError::new("Failed to find AZs", &err)]);
        }

        let availability_zones: Vec<String> = availablity_zones
            .unwrap()
            .availability_zones
            .unwrap_or_default()
            .iter()
            .take(3)
            .map(|z| z.zone_name.clone().unwrap())
            .collect();

        // 1. Create an EC2 launch template that you'll use to create an auto
 scaling group. Bonus: use SDK with EC2.CreateLaunchTemplate to create the launch
 template.
        //   * Recommended: InstanceType='t1.micro',
 ImageId='ami-0ca285d4c2cda3300'
        let create_launch_template = ec2
            .create_launch_template()
            .launch_template_name(LAUNCH_TEMPLATE_NAME)
```

```
            .launch_template_data(
                RequestLaunchTemplateData::builder()
                    .instance_type(aws_sdk_ec2::types::InstanceType::T1Micro)
                    .image_id("ami-0ca285d4c2cda3300")
                    .build(),
            )
            .send()
            .await
            .map_err(|err| vec![ScenarioError::new("Failed to create launch
template", &err)])?;

        let launch_template_arn = match create_launch_template.launch_template {
            Some(launch_template) =>
launch_template.launch_template_id.unwrap_or_default(),
            None => {
                // Try to delete the launch template
                let _ = ec2
                    .delete_launch_template()
                    .launch_template_name(LAUNCH_TEMPLATE_NAME)
                    .send()
                    .await;
                return Err(vec![ScenarioError::with("Failed to load launch
template")]);
            }
        };

        // 2. CreateAutoScalingGroup: pass it the launch template you created in
step 0. Give it min/max of 1 instance.
        //   You can use EC2.describe_availability_zones() to get a list of AZs
(you have to specify an AZ when you create the group).
        //   Wait for instance to launch. Use a waiter if you have one, otherwise
DescribeAutoScalingInstances until LifecycleState='InService'
        if let Err(err) = autoscaling
            .create_auto_scaling_group()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .launch_template(
                LaunchTemplateSpecification::builder()
                    .launch_template_id(launch_template_arn.clone())
                    .version("$Latest")
                    .build(),
            )
            .max_size(1)
            .min_size(1)
            .set_availability_zones(Some(availability_zones))
```

```
            .send()
            .await
        {
            let mut errs = vec![ScenarioError::new(
                "Failed to create autoscaling group",
                &err,
            )];

            if let Err(err) = autoscaling
                .delete_auto_scaling_group()
                .auto_scaling_group_name(auto_scaling_group_name.as_str())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up autoscaling group",
                    &err,
                ));
            }

            if let Err(err) = ec2
                .delete_launch_template()
                .launch_template_id(launch_template_arn.clone())
                .send()
                .await
            {
                errs.push(ScenarioError::new(
                    "Failed to clean up launch template",
                    &err,
                ));
            }
            return Err(errs);
        }

        let scenario = AutoScalingScenario {
            ec2,
            autoscaling: autoscaling.clone(), // Clients are cheap so cloning
here to prevent a move is ok.
            auto_scaling_group_name: auto_scaling_group_name.clone(),
            launch_template_arn,
        };

        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
```

```
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;

        match enable_metrics_collection {
            Ok(_) => Ok(scenario),
            Err(err) => {
                scenario.clean_scenario().await?;
                Err(vec![ScenarioError::new(
                    "Failed to enable metrics collections for group",
                    &err,
                )])
            }
        }
    }

    pub async fn clean_scenario(self) -> Result<(), Vec<ScenarioError>> {
        let _ = self.wait_for_no_scaling().await;
        let delete_group = self
            .autoscaling
            .delete_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 14. Delete LaunchTemplate.
        let delete_launch_template = self
            .ec2
            .delete_launch_template()
            .launch_template_id(self.launch_template_arn.clone())
            .send()
            .await;

        let early_exit = match (delete_group, delete_launch_template) {
            (Ok(_), Ok(_)) => Ok(()),
            (Ok(_), Err(e)) => Err(vec![ScenarioError::new(
```

```
                "There was an error cleaning the launch template",
                &e,
            )]),
            (Err(e), Ok(_)) => Err(vec![ScenarioError::new(
                "There was an error cleaning the scale group",
                &e,
            )]),
            (Err(e1), Err(e2)) => Err(vec![
                ScenarioError::new("Multiple error cleaning the scenario Scale
Group", &e1),
                ScenarioError::new("Multiple error cleaning the scenario Launch
Template", &e2),
            ]),
        };

        if early_exit.is_err() {
            early_exit
        } else {
            // Wait for delete_group to finish
            let waiter = Waiter::new();
            let mut errors = Vec::<ScenarioError>::new();
            while errors.len() < 3 {
                if let Err(e) = waiter.sleep().await {
                    errors.push(e);
                    continue;
                }
                let describe_group = self
                    .autoscaling
                    .describe_auto_scaling_groups()

.auto_scaling_group_names(self.auto_scaling_group_name.clone())
                    .send()
                    .await;
                match describe_group {
                    Ok(group) => match group.auto_scaling_groups().first() {
                        Some(group) => {
                            if group.status() != Some("Delete in progress") {
                                errors.push(ScenarioError::with(format!(
                                    "Group in an unknown state while deleting:
{}",
                                    group.status().unwrap_or("unknown error")
                                )));
                                return Err(errors);
                            }
```

```
                        }
                        None => return Ok(()),
                    },
                    Err(err) => {
                        errors.push(ScenarioError::new("Failed to describe
autoscaling group during cleanup 3 times, last error", &err));
                    }
                }
                if errors.len() > 3 {
                    return Err(errors);
                }
            }
            Err(vec![ScenarioError::with(
                "Exited cleanup wait loop without retuning success or failing
after three rounds",
            )])
        }
    }

    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
```

```
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
```

```
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }

    pub async fn wait_for_no_scaling(&self) -> Result<(), ScenarioError> {
        let waiter = Waiter::new();
        let mut scaling = true;
        while scaling {
            waiter.sleep().await?;
            let describe_activities = self
                .autoscaling
                .describe_scaling_activities()
                .auto_scaling_group_name(self.auto_scaling_group_name.clone())
                .send()
                .await
                .map_err(|e| {
                    ScenarioError::new("Failed to get autoscaling activities for
group", &e)
                })?;
            let activities = describe_activities.activities();
            trace!(
                "Waiting for no scaling found {} activities",
```

```
                activities.len()
            );
            scaling = activities.iter().any(|a| a.progress() < Some(100));
        }
        Ok(())
    }

    pub async fn wait_for_stable(&self, size: usize) -> Result<(), ScenarioError>
 {
        self.wait_for_no_scaling().await?;

        let mut group = self.get_group().await?;
        let mut count = count_group_instances(&group);

        let waiter = Waiter::new();
        while count != size {
            trace!("Waiting for stable {size} (current: {count})");
            waiter.sleep().await?;
            group = self.get_group().await?;
            count = count_group_instances(&group);
        }

        Ok(())
    }

    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
```

```
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
instances", &err))
    }

    pub async fn scale_min_size(&self, size: i32) -> Result<(), ScenarioError> {
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failer to update group to min size ({size}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_max_size(&self, size: i32) -> Result<(), ScenarioError> {
        // 5. UpdateAutoScalingGroup: update max size to 3.
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .max_size(size)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to max size ({size})").as_str(),
                &err,
            ));
```

```
        }
        Ok(())
    }

    pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
        // 7. SetDesiredCapacity: set desired capacity to 2.
        //   Wait for a second instance to launch.
        let update_group = self
            .autoscaling
            .set_desired_capacity()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .desired_capacity(capacity)
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                format!("Failed to update group to desired capacity
({capacity}))").as_str(),
                &err,
            ));
        }
        Ok(())
    }

    pub async fn scale_group_to_zero(&self) -> Result<(), ScenarioError> {
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;

        // 12. DeleteAutoScalingGroup (to delete the group you must stop all
instances):
        //   UpdateAutoScalingGroup with MinSize=0
        let update_group = self
            .autoscaling
            .update_auto_scaling_group()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .min_size(0)
            .desired_capacity(0)
```

```
            .send()
            .await;
        if let Err(err) = update_group {
            return Err(ScenarioError::new(
                "Failed to update group for scaling down&",
                &err,
            ));
        }

        let stable = self.wait_for_stable(0).await;
        if let Err(err) = stable {
            return Err(ScenarioError::with(format!(
                "Error while waiting for group to be stable on scale down: {err}"
            )));
        }

        Ok(())
    }

    pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
        // Retrieve a list of instances in the auto scaling group.
        let auto_scaling_group = self.get_group().await?;
        let instances = auto_scaling_group.instances();
        // Or use other logic to find an instance to terminate.
        let instance = instances.first();
        if let Some(instance) = instance {
            let instance_id = if let Some(instance_id) = instance.instance_id() {
                instance_id
            } else {
                return Err(ScenarioError::with("Missing instance id"));
            };
            let termination = self
                .ec2
                .terminate_instances()
                .instance_ids(instance_id)
                .send()
                .await;
            if let Err(err) = termination {
                Err(ScenarioError::new(
                    "There was a problem terminating an instance",
                    &err,
                ))
            } else {
                Ok(())
```

```
            }
        } else {
            Err(ScenarioError::with("There was no instance to terminate"))
        }
    }
}

fn count_group_instances(group: &AutoScalingGroup) -> usize {
    group.instances.as_ref().map(|i| i.len()).unwrap_or(0)
}
```

- API 자세한 내용은 AWS SDK Rust API 참조 의 다음 주제를 참조하세요.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

를 사용한 Auto Scaling 작업 AWS SDKs

다음 코드 예제에서는 를 사용하여 개별 Auto Scaling 작업을 수행하는 방법을 보여줍니다 AWS SDKs. 각 예제에는 코드 설정 및 실행에 대한 지침을 찾을 수 GitHub있는 에 대한 링크가 포함되어 있습니다.

이러한 발췌문은 Auto Scaling을 호출API하며 컨텍스트에서 실행해야 하는 더 큰 프로그램의 코드 발췌문입니다. 를 사용한 Auto Scaling 시나리오 AWS SDKs 에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon EC2 Auto Scaling API 참조를 참조하세요.

예시

- 또는 와 AttachInstancesAWS SDK 함께 사용 CLI
- 또는 와 AttachLoadBalancerTargetGroupsAWS SDK 함께 사용 CLI
- 또는 와 AttachLoadBalancersAWS SDK 함께 사용 CLI
- 또는 와 CompleteLifecycleActionAWS SDK 함께 사용 CLI
- 또는 와 CreateAutoScalingGroupAWS SDK 함께 사용 CLI
- 또는 와 CreateLaunchConfigurationAWS SDK 함께 사용 CLI
- 또는 와 CreateOrUpdateTagsAWS SDK 함께 사용 CLI
- 또는 와 DeleteAutoScalingGroupAWS SDK 함께 사용 CLI
- 또는 와 DeleteLaunchConfigurationAWS SDK 함께 사용 CLI
- 또는 와 DeleteLifecycleHookAWS SDK 함께 사용 CLI
- 또는 와 DeleteNotificationConfigurationAWS SDK 함께 사용 CLI
- 또는 와 DeletePolicyAWS SDK 함께 사용 CLI
- 또는 와 DeleteScheduledActionAWS SDK 함께 사용 CLI
- 또는 와 DeleteTagsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAccountLimitsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAdjustmentTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingGroupsAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingInstancesAWS SDK 함께 사용 CLI
- 또는 와 DescribeAutoScalingNotificationTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeLaunchConfigurationsAWS SDK 함께 사용 CLI
- 또는 와 DescribeLifecycleHookTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeLifecycleHooksAWS SDK 함께 사용 CLI
- 또는 와 DescribeLoadBalancersAWS SDK 함께 사용 CLI
- 또는 와 DescribeMetricCollectionTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeNotificationConfigurationsAWS SDK 함께 사용 CLI
- 또는 와 DescribePoliciesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScalingActivitiesAWS SDK 함께 사용 CLI

- 또는 와 DescribeScalingProcessTypesAWS SDK 함께 사용 CLI
- 또는 와 DescribeScheduledActionsAWS SDK 함께 사용 CLI
- 또는 와 DescribeTagsAWS SDK 함께 사용 CLI
- 또는 와 DescribeTerminationPolicyTypesAWS SDK 함께 사용 CLI
- 또는 와 DetachInstancesAWS SDK 함께 사용 CLI
- 또는 와 DetachLoadBalancersAWS SDK 함께 사용 CLI
- 또는 와 DisableMetricsCollectionAWS SDK 함께 사용 CLI
- 또는 와 EnableMetricsCollectionAWS SDK 함께 사용 CLI
- 또는 와 EnterStandbyAWS SDK 함께 사용 CLI
- 또는 와 ExecutePolicyAWS SDK 함께 사용 CLI
- 또는 와 ExitStandbyAWS SDK 함께 사용 CLI
- 또는 와 PutLifecycleHookAWS SDK 함께 사용 CLI
- 또는 와 PutNotificationConfigurationAWS SDK 함께 사용 CLI
- 또는 와 PutScalingPolicyAWS SDK 함께 사용 CLI
- 또는 와 PutScheduledUpdateGroupActionAWS SDK 함께 사용 CLI
- 또는 와 RecordLifecycleActionHeartbeatAWS SDK 함께 사용 CLI
- 또는 와 ResumeProcessesAWS SDK 함께 사용 CLI
- 또는 와 SetDesiredCapacityAWS SDK 함께 사용 CLI
- 또는 와 SetInstanceHealthAWS SDK 함께 사용 CLI
- 또는 와 SetInstanceProtectionAWS SDK 함께 사용 CLI
- 또는 와 SuspendProcessesAWS SDK 함께 사용 CLI
- 또는 와 TerminateInstanceInAutoScalingGroupAWS SDK 함께 사용 CLI
- 또는 와 UpdateAutoScalingGroupAWS SDK 함께 사용 CLI

또는 와 **`AttachInstances`**AWS SDK 함께 사용 CLI

다음 코드 예제는 `AttachInstances`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹에 인스턴스를 연결하려면

이 예제에서는 지정된 인스턴스를 지정된 Auto Scaling 그룹에 연결합니다.

```
aws autoscaling attach-instances \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

이 명령은 출력을 생성하지 않습니다.

- 자세한 API 내용은 명령 참조AttachInstances의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 인스턴스를 지정된 Auto Scaling 그룹에 연결합니다. Auto Scaling은 Auto Scaling 그룹의 원하는 용량을 자동으로 증가시킵니다.

```
Mount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

- 자세한 API 내용은 Cmdlet 참조AttachInstances의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **`AttachLoadBalancerTargetGroups`**AWS SDK 함께 사용 CLI

다음 코드 예제는 `AttachLoadBalancerTargetGroups`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
```

- 자세한 API 내용은 참조AttachLoadBalancerTargetGroups의 섹션을 참조하세요. AWS SDK for .NET API

CLI

AWS CLI

Auto Scaling 그룹에 대상 그룹을 연결하는 방법

이 예시에서는 지정된 Auto Scaling 그룹에 지정된 대상 그룹을 연결합니다.

```
aws autoscaling attach-load-balancer-target-groups \
    --auto-scaling-group-name my-asg \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Elastic Load Balancing 및 Amazon Auto Scaling을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조AttachLoadBalancerTargetGroups의 섹션을 참조하세요. AWS CLI

JavaScript

SDK 용 JavaScript (v3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
const client = new AutoScalingClient({});
await client.send(
  new AttachLoadBalancerTargetGroupsCommand({
    AutoScalingGroupName: NAMES.autoScalingGroupName,
    TargetGroupARNs: [state.targetGroupArn],
  }),
);
```

- 자세한 API 내용은 참조AttachLoadBalancerTargetGroups의 섹션을 참조하세요. AWS SDK for JavaScript API

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
```

```
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")
```

- API 자세한 내용은 AttachLoadBalancerTargetGroups의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **AttachLoadBalancers**AWS SDK 함께 사용 CLI

다음 코드 예제는 `AttachLoadBalancers`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Classic Load Balancer를 Auto Scaling 그룹에 연결하려면

이 예제에서는 지정된 Classic Load Balancer를 지정된 Auto Scaling 그룹에 연결합니다.

```
aws autoscaling attach-load-balancers \
    --load-balancer-names my-load-balancer \
```

```
    --auto-scaling-group-name my-asg
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Elastic Load Balancing 및 Amazon Auto Scaling을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조AttachLoadBalancers의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 로드 밸런서를 지정된 Auto Scaling 그룹에 연결합니다.

```
Add-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- API 자세한 내용은 Cmdlet 참조AttachLoadBalancers의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **CompleteLifecycleAction**AWS SDK 함께 사용 CLI

다음 코드 예제는 `CompleteLifecycleAction`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

수명 주기 작업을 완료하려면

이 예제에서는 지정된 수명 주기 작업이 완료되어 인스턴스 시작 또는 종료를 완료할 수 있음을 Amazon EC2 Auto Scaling에 알립니다.

```
aws autoscaling complete-lifecycle-action \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-result CONTINUE \
```

```
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 수명 주기 후크를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조CompleteLifecycleAction의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 수명 주기 작업을 완료합니다.

```
Complete-ASLifecycleAction -LifecycleHookName myLifecycleHook -
AutoScalingGroupName my-asg -LifecycleActionResult CONTINUE -LifecycleActionToken
 bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- API 자세한 내용은 Cmdlet 참조CompleteLifecycleAction의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **`CreateAutoScalingGroup`**AWS SDK 함께 사용 CLI

다음 코드 예제는 `CreateAutoScalingGroup`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기
- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Create a new Amazon EC2 Auto Scaling group.
/// </summary>
/// <param name="groupName">The name to use for the new Auto Scaling
/// group.</param>
/// <param name="launchTemplateName">The name of the Amazon EC2 Auto Scaling
/// launch template to use to create instances in the group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> CreateAutoScalingGroupAsync(
    string groupName,
    string launchTemplateName,
    string availabilityZone)
{
    var templateSpecification = new LaunchTemplateSpecification
    {
        LaunchTemplateName = launchTemplateName,
    };

    var zoneList = new List<string>
        {
            availabilityZone,
        };

    var request = new CreateAutoScalingGroupRequest
    {
        AutoScalingGroupName = groupName,
        AvailabilityZones = zoneList,
        LaunchTemplate = templateSpecification,
        MaxSize = 6,
        MinSize = 1
    };
```

```
        var response = await
_amazonAutoScaling.CreateAutoScalingGroupAsync(request);
        Console.WriteLine($"{groupName} Auto Scaling Group created");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 자세한 API 내용은 참조CreateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        Aws::Vector<Aws::String> availabilityGroupZones;
        availabilityGroupZones.push_back(
                availabilityZones[availabilityZoneChoice - 1].GetZoneName());
        request.SetAvailabilityZones(availabilityGroupZones);
        request.SetMaxSize(1);
        request.SetMinSize(1);

        Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
        launchTemplateSpecification.SetLaunchTemplateName(templateName);
        request.SetLaunchTemplate(launchTemplateSpecification);
```

```
        Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
                autoScalingClient.CreateAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Created Auto Scaling group '" << groupName << "'..."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
            std::cout << "Auto Scaling group '" << groupName << "' already
 exists."
                      << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 자세한 API 내용은 참조CreateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

예 1: Auto Scaling 그룹을 생성하는 방법

다음 `create-auto-scaling-group` 예시에서는 리전 내 여러 가용 영역의 서브넷에 Auto Scaling 그룹을 생성합니다. 지정된 시작 템플릿의 기본 버전으로 인스턴스가 시작됩니다. 참고로 종료 정책, 상태 확인 구성 등 대부분의 다른 설정에는 기본값이 사용됩니다.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹을 참조하세요. EC2 Auto Scaling

예 2: Application Load Balancer, Network Load Balancer 또는 Gateway Load Balancer를 연결하는 방법

이 예제ARN에서는 예상 트래픽을 지원하는 로드 밸런서에 대한 대상 그룹의 를 지정합니다. 상태 확인 유형은 Elastic Load Balancing이 인스턴스를 비정상으로 보고하면 Auto Scaling 그룹이 인스턴스를 교체하도록 ELB를 지정합니다. 또한 이 명령은 상태 확인 유예 기간을 600초로 정의합니다. 유예 기간은 새로 시작된 인스턴스의 조기 종료를 방지하는 데 도움이 됩니다.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12 \
    --target-group-arns arn:aws:elasticloadbalancing:us-
west-2:123456789012:targetgroup/my-targets/943f017f100becff \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --min-size 1 \
    --max-size 5 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Elastic Load Balancing 및 Amazon Auto Scaling을 참조하세요. EC2 Auto Scaling

예 3: 배치 그룹을 지정하고 시작 템플릿의 최신 버전을 사용하는 방법

이 예시에서는 단일 가용 영역 내에 있는 배치 그룹에 인스턴스를 시작합니다. 이는 HPC 워크로드가 있는 지연 시간이 짧은 그룹에 유용할 수 있습니다. 또한 이 예시에서는 그룹의 최소 크기, 최대 크기, 원하는 용량을 지정합니다.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest' \
    --min-size 1 \
    --max-size 5 \
    --desired-capacity 3 \
    --placement-group my-placement-group \
```

```
    --vpc-zone-identifier "subnet-6194ea3b"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Linux 인스턴스용 Amazon 사용 설명서의 배치 그룹을 참조하세요. EC2

예 4: 단일 인스턴스 Auto Scaling 그룹을 지정하고 시작 템플릿의 특정 버전을 사용하는 방법

이 예시에서는 최소 및 최대 용량을 1로 설정한 Auto Scaling 그룹을 생성하여 하나의 인스턴스가 실행되도록 합니다. 명령은 또한 기존 의 ID가 ENI 지정된 시작 템플릿의 v1을 지정합니다. eth0ENI에 기존 를 지정하는 시작 템플릿을 사용하는 경우 요청에 서브넷 ID를 지정하지 않고 네트워크 인터페이스와 일치하는 Auto Scaling 그룹의 가용 영역을 지정해야 합니다.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='1'
 \
    --min-size 1 \
    --max-size 1 \
    --availability-zones us-west-2a
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹을 참조하세요. EC2 Auto Scaling

예 5: 다른 종료 정책을 지정하는 방법

이 예시에서는 시작 구성을 사용하여 Auto Scaling 그룹을 생성하고 가장 오래된 인스턴스부터 종료하도록 종료 정책을 설정합니다. 이 명령은 또한 키가 `Role`이고 값이 `WebServer`인 태그를 그룹과 해당 인스턴스에 적용합니다.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-configuration-name my-lc \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "OldestInstance" \
    --tags "ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 종료 정책 작업을 참조하세요. EC2 Auto Scaling

예 6: 시작 수명 주기 후크를 지정하는 방법

이 예시에서는 인스턴스 시작 시 사용자 지정 작업을 지원하는 수명 주기 후크가 있는 Auto Scaling 그룹을 생성합니다.

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

`config.json` 파일의 콘텐츠:

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-launch-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING",
        "NotificationTargetARN": "arn:aws:sqs:us-west-2:123456789012:my-sqs-
queue",
        "RoleARN": "arn:aws:iam::123456789012:role/my-notification-role",
        "NotificationMetadata": "SQS message metadata",
        "HeartbeatTimeout": 4800,
        "DefaultResult": "ABANDON"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
    "Tags": [{
        "ResourceType": "auto-scaling-group",
        "ResourceId": "my-asg",
        "PropagateAtLaunch": true,
        "Value": "test",
        "Key": "environment"
    }]
}
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 수명 주기 후크를 참조하세요. EC2 Auto Scaling

예 7: 종료 수명 주기 후크를 지정하는 방법

이 예시에서는 인스턴스 종료 시 사용자 지정 작업을 지원하는 수명 주기 후크가 있는 Auto Scaling 그룹을 생성합니다.

```
aws autoscaling create-auto-scaling-group \
    --cli-input-json file://~/config.json
```

`config.json`의 콘텐츠:

```
{
    "AutoScalingGroupName": "my-asg",
    "LaunchTemplate": {
        "LaunchTemplateId": "lt-1234567890abcde12"
    },
    "LifecycleHookSpecificationList": [{
        "LifecycleHookName": "my-termination-hook",
        "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING",
        "HeartbeatTimeout": 120,
        "DefaultResult": "CONTINUE"
    }],
    "MinSize": 1,
    "MaxSize": 5,
    "TargetGroupARNs": [
        "arn:aws:elasticloadbalancing:us-west-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067"
    ],
    "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
}
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 수명 주기 후크를 참조하세요. EC2 Auto Scaling

예 8: 사용자 지정 종료 정책을 지정하는 방법

이 예제에서는 규모에 따라 종료해도 안전한 인스턴스를 Amazon Auto Scaling에 알려주는 사용자 지정 Lambda 함수 종료 정책을 지정하는 EC2 Auto Scaling 그룹을 생성합니다.

```
aws autoscaling create-auto-scaling-group \
    --auto-scaling-group-name my-asg-single-instance \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling \
    --min-size 1 \
    --max-size 5 \
    --termination-policies "arn:aws:lambda:us-
west-2:123456789012:function:HelloFunction:prod" \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Lambda를 사용하여 사용자 지정 종료 정책 생성을 참조하세요.

- 자세한 API 내용은 명령 참조CreateAutoScalingGroup의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.CreateAutoScalingGroupRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.LaunchTemplateSpecification;
import software.amazon.awssdk.services.autoscaling.waiters.AutoScalingWaiter;

/**
```

```
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAutoScalingGroup {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName> <launchTemplateName> <serviceLinkedRoleARN>
 <vpcZoneId>

                Where:
                    groupName - The name of the Auto Scaling group.
                    launchTemplateName - The name of the launch template.\s
                    vpcZoneId - A subnet Id for a virtual private cloud (VPC)
 where instances in the Auto Scaling group can be created.
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        String launchTemplateName = args[1];
        String vpcZoneId = args[2];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
 vpcZoneId);
        autoScalingClient.close();
    }

    public static void createAutoScalingGroup(AutoScalingClient
 autoScalingClient,
            String groupName,
            String launchTemplateName,
```

```
            String vpcZoneId) {

        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
 LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            CreateAutoScalingGroupRequest request =
 CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .availabilityZones("us-east-1a")
                    .launchTemplate(templateSpecification)
                    .maxSize(1)
                    .minSize(1)
                    .vpcZoneIdentifier(vpcZoneId)
                    .build();

            autoScalingClient.createAutoScalingGroup(request);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupExists(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 자세한 API 내용은 참조CreateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
    vpcZoneIdVal: String,
) {
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val request =
        CreateAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            availabilityZones = listOf("us-east-1a")
            launchTemplate = templateSpecification
            maxSize = 1
            minSize = 1
            vpcZoneIdentifier = vpcZoneIdVal
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
        }

    // This object is required for the waiter call.
    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.createAutoScalingGroup(request)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("$groupName was created!")
```

```
    }
}
```

- API 자세한 내용은 CreateAutoScalingGroup의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function createAutoScalingGroup(
        $autoScalingGroupName,
        $availabilityZones,
        $minSize,
        $maxSize,
        $launchTemplateId
    ) {
        return $this->autoScalingClient->createAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'AvailabilityZones' => $availabilityZones,
            'MinSize' => $minSize,
            'MaxSize' => $maxSize,
            'LaunchTemplate' => [
                'LaunchTemplateId' => $launchTemplateId,
            ],
        ]);
    }
```

- 자세한 API 내용은 참조CreateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 이름과 속성을 사용하여 Auto Scaling 그룹을 생성합니다. 원하는 기본 용량은 최소 크기입니다. 따라서 이 Auto Scaling 그룹은 지정된 두 가용 영역에 하나씩 있는 두 개의 인스턴스를 시작합니다.

```
New-ASAutoScalingGroup -AutoScalingGroupName my-asg -LaunchConfigurationName my-
lc -MinSize 2 -MaxSize 6 -AvailabilityZone @("us-west-2a", "us-west-2b")
```

- API 자세한 내용은 Cmdlet 참조CreateAutoScalingGroup의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def create_group(
        self,
        group_name: str,
        group_zones: List[str],
        launch_template_name: str,
        min_size: int,
        max_size: int,
```

```
    ) -> None:
        """
        Creates an Auto Scaling group.

        :param group_name: The name to give to the group.
        :param group_zones: The Availability Zones in which instances can be
created.
        :param launch_template_name: The name of an existing Amazon EC2 launch
template.
                                     The launch template specifies the
configuration of
                                     instances that are created by auto scaling
activities.
        :param min_size: The minimum number of active instances in the group.
        :param max_size: The maximum number of active instances in the group.
        :return: None
        :raises ClientError: If there is an error creating the Auto Scaling
group.
        """
        try:
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=group_name,
                AvailabilityZones=group_zones,
                LaunchTemplate={
                    "LaunchTemplateName": launch_template_name,
                    "Version": "$Default",
                },
                MinSize=min_size,
                MaxSize=max_size,
            )

            # Wait for the group to exist.
            waiter = self.autoscaling_client.get_waiter("group_exists")
            waiter.wait(AutoScalingGroupNames=[group_name])

            logger.info(f"Successfully created Auto Scaling group {group_name}.")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to create Auto Scaling group {group_name}.")
            if error_code == "AlreadyExistsFault":
                logger.error(
                    f"An Auto Scaling group with the name '{group_name}' already
exists. "
```

```
                    "Please use a different name or update the existing group.",
                )
            elif error_code == "LimitExceededFault":
                logger.error(
                    "The request failed because you have reached the limit "
                    "on the number of Auto Scaling groups or launch
 configurations. "
                    "Consider deleting unused resources or request a limit
 increase. "
                    "\nSee Auto Scaling Service Quota documentation here:"
                    "\n\thttps://docs.aws.amazon.com/autoscaling/ec2/userguide/
ec2-auto-scaling-quotas.html"
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 자세한 내용은 CreateAutoScalingGroup의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn create_group(client: &Client, name: &str, id: &str) -> Result<(), Error>
 {
    client
        .create_auto_scaling_group()
        .auto_scaling_group_name(name)
        .instance_id(id)
        .min_size(1)
        .max_size(5)
        .send()
        .await?;
```

```
    println!("Created AutoScaling group");

    Ok(())
}
```

- API 자세한 내용은 Rust 참조 CreateAutoScalingGroup 의 섹션을 참조하세요. AWS SDK API

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **CreateLaunchConfiguration**AWS SDK 함께 사용 CLI

다음 코드 예제는 `CreateLaunchConfiguration`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 시작 구성을 생성하려면

이 예제에서는 간단한 시작 구성을 생성합니다.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 시작 구성 생성을 참조하세요.

예제 2: 보안 그룹, 키 페어 및 부트래핑 스크립트를 사용하여 시작 구성을 생성하려면

이 예제에서는 보안 그룹, 키 페어 및 사용자 데이터에 포함된 부트래핑 스크립트를 사용하여 시작 구성을 생성합니다.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
```

```
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --security-groups sg-eb2af88example \
    --key-name my-key-pair \
    --user-data file://myuserdata.txt
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 시작 구성 생성을 참조하세요.

예제 3: IAM 역할로 시작 구성을 생성하려면

이 예제에서는 IAM 역할의 인스턴스 프로파일 이름으로 시작 구성을 생성합니다.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --iam-instance-profile my-autoscaling-role
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 IAM Amazon Auto Scaling 사용 설명서의 Amazon EC2 인스턴스에서 실행되는 애플리케이션의 역할을 참조하세요. EC2 Auto Scaling

예제 4: 세부 모니터링이 활성화된 상태로 시작 구성을 생성하려면

이 예제에서는 EC2 세부 모니터링이 활성화된 시작 구성을 생성하여 1분 동안 EC2 지표를 CloudWatch 에 전송합니다.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --instance-monitoring Enabled=true
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 인스턴스에 대한 모니터링 구성을 참조하세요. EC2 Auto Scaling

예제 5: 스팟 인스턴스를 시작하는 시작 구성을 생성하려면

이 예제에서는 스팟 인스턴스를 유일한 구매 옵션으로 사용하는 시작 구성을 생성합니다.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --spot-price "0.50"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 스팟 인스턴스 요청을 참조하세요.

예제 6: EC2 인스턴스를 사용하여 시작 구성을 생성하려면

이 예제에서는 기존 인스턴스의 속성을 기반으로 시작 구성을 생성합니다. 배치 테넌시와 `--placement-tenancy` 및 `--no-associate-public-ip-address` 옵션을 포함하여 퍼블릭 IP 주소를 설정할지 여부를 재정의합니다.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc-from-instance \
    --instance-id i-0123a456700123456 \
    --instance-type m5.large \
    --no-associate-public-ip-address \
    --placement-tenancy dedicated
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 EC2 인스턴스를 사용하여 시작 구성 생성을 참조하세요.

예제 7: Amazon EBS 볼륨에 대한 블록 디바이스 매핑을 사용하여 시작 구성을 생성하려면

이 예제에서는 디바이스 이름과 EBS gp3 볼륨 크기가 20인 Amazon `/dev/sdh` 볼륨에 대한 블록 디바이스 매핑을 사용하여 시작 구성을 생성합니다.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdh","Ebs":
{"VolumeSize":20,"VolumeType":"gp3"}}]'
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 참조EBS의 섹션을 참조하세요. EC2 Auto Scaling API

형식이 JSON지정된 파라미터 값을 인용하는 구문에 대한 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 에서 문자열로 인용 부호 사용을 참조하세요 AWS CLI.

예제 8: 인스턴스 스토어 볼륨에 대한 블록 디바이스 매핑을 사용하여 시작 구성을 생성하려면

이 예제에서는 를 인스턴스 스토어 볼륨`ephemeral1`으로 사용하여 디바이스 이름이 인 시작 구성을 생성합니다`/dev/sdc`.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/
sdc","VirtualName":"ephemeral1"}]'
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 참조BlockDeviceMapping의 섹션을 참조하세요. EC2 Auto Scaling API

따옴표 JSON형식의 파라미터 값을 위한 구문에 대한 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 에서 문자열로 따옴표 사용을 참조하세요 AWS CLI.

예제 9: 시작 구성을 생성하고 시작 시 블록 디바이스가 연결되지 않도록 하려면

이 예제에서는 의 블록 디바이스 매핑AMI(예:)에 의해 지정된 블록 디바이스를 억제하는 시작 구성을 생성합니다`/dev/sdf`.

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name my-lc \
    --image-id ami-04d5cc9b88example \
    --instance-type m5.large \
    --block-device-mappings '[{"DeviceName":"/dev/sdf","NoDevice":""}]'
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 참조BlockDeviceMapping의 섹션을 참조하세요. EC2 Auto Scaling API

따옴표 JSON형식의 파라미터 값을 위한 구문에 대한 자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 에서 문자열로 따옴표 사용을 참조하세요 AWS CLI.

- 자세한 API 내용은 명령 참조CreateLaunchConfiguration의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 'my-lc'라는 시작 구성을 생성합니다. 이 시작 구성을 사용하는 Auto Scaling 그룹에서 시작한 EC2 인스턴스는 지정된 인스턴스 유형, AMI, 보안 그룹 및 IAM 역할을 사용합니다.

```
New-ASLaunchConfiguration -LaunchConfigurationName my-lc -InstanceType
 "m3.medium" -ImageId "ami-12345678" -SecurityGroup "sg-12345678" -
IamInstanceProfile "myIamRole"
```

- API 자세한 내용은 Cmdlet 참조CreateLaunchConfiguration의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **`CreateOrUpdateTags`**AWS SDK 함께 사용 CLI

다음 코드 예제는 `CreateOrUpdateTags`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹에 대한 태그를 생성하거나 업데이트하려면

이 예제에서는 지정된 Auto Scaling 그룹에 두 개의 태그를 추가합니다.

```
aws autoscaling create-or-update-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Role,Value=WebServer,PropagateAtLaunch=true ResourceId=my-
asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research,PropagateAtLaunch=true
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스 태그를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조CreateOrUpdateTags의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 단일 태그를 추가합니다. 태그 키는 'myTag'이고 태그 값은 'myTagValue'입니다. Auto Scaling은 이 태그를 Auto Scaling 그룹에서 시작한 후속 EC2 인스턴스로 전파합니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Set-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag"; Value="myTagValue"; PropagateAtLaunch=$true} )
```

예제 2: PowerShell 버전 2에서는 New-Object를 사용하여 태그 파라미터에 대한 태그를 생성해야 합니다.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
$tag.Value = "myTagValue"
$tag.PropagateAtLaunch = $true
Set-ASTag -Tag $tag
```

- API 자세한 내용은 Cmdlet 참조CreateOrUpdateTags의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DeleteAutoScalingGroup**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DeleteAutoScalingGroup`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기
- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Auto Scaling 그룹의 최소 크기를 0으로 업데이트하고 그룹의 모든 인스턴스를 해지한 다음 그룹을 삭제합니다.

```
    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
```

```
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
```

```
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
 groupName)
    {
        var describeGroupsResponse = await
 _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }
```

```
    /// <summary>
    /// Delete an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAutoScalingGroupAsync(
```

```
        string groupName)
    {
        var deleteAutoScalingGroupRequest = new DeleteAutoScalingGroupRequest
        {
            AutoScalingGroupName = groupName,
            ForceDelete = true,
        };

        var response = await
_amazonAutoScaling.DeleteAutoScalingGroupAsync(deleteAutoScalingGroupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully deleted {groupName}");
            return true;
        }

        Console.WriteLine($"Couldn't delete {groupName}.");
        return false;
    }
```

- 자세한 API 내용은 참조DeleteAutoScalingGroup의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
```

```
            Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);

            Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
                    autoScalingClient.DeleteAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout << "Auto Scaling group '" << groupName << "' was
 deleted."
                          << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
```

- 자세한 API 내용은 참조DeleteAutoScalingGroup의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

예 1: 지정된 Auto Scaling 그룹을 삭제하는 방법

이 예시에서는 지정된 Auto Scaling 그룹을 삭제합니다.

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 인프라 삭제를 참조하세요. EC2 Auto Scaling

예 2: 지정된 Auto Scaling 그룹을 강제로 삭제하는 방법

그룹의 인스턴스가 종료될 때까지 기다리지 않고 Auto Scaling 그룹을 삭제하려면 `--force-delete` 옵션을 사용하세요.

```
aws autoscaling delete-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --force-delete
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 인프라 삭제를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DeleteAutoScalingGroup의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import
 software.amazon.awssdk.services.autoscaling.model.DeleteAutoScalingGroupRequest;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAutoScalingGroup {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        deleteAutoScalingGroup(autoScalingClient, groupName);
        autoScalingClient.close();
    }

    public static void deleteAutoScalingGroup(AutoScalingClient
 autoScalingClient, String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
 DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();


 autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 자세한 API 내용은 참조DeleteAutoScalingGroup의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteSpecificAutoScalingGroup(groupName: String) {
    val deleteAutoScalingGroupRequest =
        DeleteAutoScalingGroupRequest {
            autoScalingGroupName = groupName
            forceDelete = true
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest)
        println("You successfully deleted $groupName")
    }
}
```

- API 자세한 내용은 DeleteAutoScalingGroup의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function deleteAutoScalingGroup($autoScalingGroupName)
    {
        return $this->autoScalingClient->deleteAutoScalingGroup([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'ForceDelete' => true,
        ]);
    }
```

- 자세한 API 내용은 참조DeleteAutoScalingGroup의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 실행 중인 인스턴스가 없는 경우 지정된 Auto Scaling 그룹을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASAutoScalingGroup (DeleteAutoScalingGroup)" on
 Target "my-asg".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -Force
```

예제 3: 이 예제는 지정된 Auto Scaling 그룹을 삭제하고 포함된 실행 중인 인스턴스를 모두 종료합니다.

```
Remove-ASAutoScalingGroup -AutoScalingGroupName my-asg -ForceDelete $true -Force
```

- API 자세한 내용은 Cmdlet 참조DeleteAutoScalingGroup의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

Auto Scaling 그룹의 최소 크기를 0으로 업데이트하고 그룹의 모든 인스턴스를 해지한 다음 그룹을 삭제합니다.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
```

```
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
```

```
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")
```

- API 자세한 내용은 DeleteAutoScalingGroup의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn delete_group(client: &Client, name: &str, force: bool) -> Result<(),
 Error> {
    client
        .delete_auto_scaling_group()
        .auto_scaling_group_name(name)
        .set_force_delete(if force { Some(true) } else { None })
        .send()
        .await?;

    println!("Deleted Auto Scaling group");

    Ok(())
}
```

- API 자세한 내용은 Rust 참조 DeleteAutoScalingGroup 의 섹션을 참조하세요. AWS SDK API

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **`DeleteLaunchConfiguration`**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DeleteLaunchConfiguration`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

시작 구성을 삭제하려면

이 예제에서는 지정된 시작 구성을 삭제합니다.

```
aws autoscaling delete-launch-configuration \
    --launch-configuration-name my-launch-config
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 인프라 삭제를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DeleteLaunchConfiguration의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 Auto Scaling 그룹에 연결되지 않은 경우 지정된 시작 구성을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLaunchConfiguration (DeleteLaunchConfiguration)"
 on Target "my-lc".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-ASLaunchConfiguration -LaunchConfigurationName my-lc -Force
```

- API 자세한 내용은 Cmdlet 참조DeleteLaunchConfiguration의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DeleteLifecycleHook**AWS SDK 함께 사용 CLI

다음 코드 예제는 DeleteLifecycleHook의 사용 방법을 보여 줍니다.

CLI

AWS CLI

수명 주기 후크를 삭제하려면

이 예제에서는 지정된 수명 주기 후크를 삭제합니다.

```
aws autoscaling delete-lifecycle-hook \
    --lifecycle-hook-name my-lifecycle-hook \
    --auto-scaling-group-name my-asg
```

이 명령은 출력을 생성하지 않습니다.

- 자세한 API 내용은 명령 참조DeleteLifecycleHook의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹에 대해 지정된 수명 주기 후크를 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASLifecycleHook (DeleteLifecycleHook)" on Target
 "myLifecycleHook".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

예 2: 강제 파라미터를 지정하는 경우 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -Force
```

- API 자세한 내용은 Cmdlet 참조DeleteLifecycleHook의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DeleteNotificationConfiguration**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DeleteNotificationConfiguration`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 알림을 삭제하려면

이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 알림을 삭제합니다.

```
aws autoscaling delete-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 알림 구성 삭제를 참조하세요.

- 자세한 API 내용은 명령 참조DeleteNotificationConfiguration의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 알림 작업을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic"
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASNotificationConfiguration
 (DeleteNotificationConfiguration)" on Target
"arn:aws:sns:us-west-2:123456789012:my-topic".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-ASNotificationConfiguration -AutoScalingGroupName my-asg -TopicARN
 "arn:aws:sns:us-west-2:123456789012:my-topic" -Force
```

- API 자세한 내용은 Cmdlet 참조DeleteNotificationConfiguration의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DeletePolicy**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DeletePolicy`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

조정 정책을 삭제하려면

이 예제에서는 지정된 조정 정책을 삭제합니다.

```
aws autoscaling delete-policy \
    --auto-scaling-group-name my-asg \
    --policy-name alb1000-target-tracking-scaling-policy
```

이 명령은 출력을 생성하지 않습니다.

- 자세한 API 내용은 명령 참조DeletePolicy의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 정책을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASPolicy (DeletePolicy)" on Target
 "myScaleInPolicy".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-ASPolicy -AutoScalingGroupName my-asg -PolicyName myScaleInPolicy -Force
```

- API 자세한 내용은 Cmdlet 참조DeletePolicy의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DeleteScheduledAction**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DeleteScheduledAction`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹에서 예약된 작업을 삭제하려면

이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 예약된 작업을 삭제합니다.

```
aws autoscaling delete-scheduled-action \
```

```
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action
```

이 명령은 출력을 생성하지 않습니다.

- 자세한 API 내용은 명령 참조DeleteScheduledAction의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹에 대해 지정된 예약된 작업을 삭제합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction"
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-ASScheduledAction (DeleteScheduledAction)" on Target
 "myScheduledAction".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-ASScheduledAction -AutoScalingGroupName my-asg -ScheduledAction
 "myScheduledAction" -Force
```

- 자세한 API 내용은 Cmdlet 참조DeleteScheduledAction의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DeleteTags**AWS SDK 함께 사용 CLI

다음 코드 예제는 DeleteTags의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹에서 태그를 삭제하려면

이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 태그를 삭제합니다.

```
aws autoscaling delete-tags \
    --tags ResourceId=my-asg,ResourceType=auto-scaling-
group,Key=Dept,Value=Research
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스 태그를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DeleteTags의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹에서 지정된 태그를 제거합니다. 작업이 진행되기 전에 확인 메시지가 표시됩니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } )
```

출력:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-ASTag (DeleteTags)" on target
 "Amazon.AutoScaling.Model.Tag".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
 "Y"):
```

예제 2: 강제 파라미터를 지정하면 작업이 진행되기 전에 확인 메시지가 표시되지 않습니다.

```
Remove-ASTag -Tag @( @{ResourceType="auto-scaling-group"; ResourceId="my-asg";
 Key="myTag" } ) -Force
```

예제 3: Powershell 버전 2에서는 New-Object를 사용하여 태그 파라미터에 대한 태그를 생성해야 합니다.

```
$tag = New-Object Amazon.AutoScaling.Model.Tag
$tag.ResourceType = "auto-scaling-group"
$tag.ResourceId = "my-asg"
$tag.Key = "myTag"
Remove-ASTag -Tag $tag -Force
```

- API 자세한 내용은 Cmdlet 참조DeleteTags의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **`DescribeAccountLimits`**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DescribeAccountLimits`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Amazon EC2 Auto Scaling 계정 제한을 설명하려면

이 예제에서는 AWS 계정의 Amazon EC2 Auto Scaling 제한에 대해 설명합니다.

```
aws autoscaling describe-account-limits
```

출력:

```
{
    "NumberOfLaunchConfigurations": 5,
    "MaxNumberOfLaunchConfigurations": 100,
    "NumberOfAutoScalingGroups": 3,
    "MaxNumberOfAutoScalingGroups": 20
}
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 서비스 할당량을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeAccountLimits의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 AWS 계정의 Auto Scaling 리소스 제한에 대해 설명합니다.

```
Get-ASAccountLimit
```

출력:

```
MaxNumberOfAutoScalingGroups    : 20
MaxNumberOfLaunchConfigurations : 100
```

- API 자세한 내용은 Cmdlet 참조DescribeAccountLimits의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeAdjustmentTypes**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DescribeAdjustmentTypes`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용 가능한 조정 유형을 설명하려면

이 예제에서는 사용 가능한 조정 유형을 설명합니다.

```
aws autoscaling describe-adjustment-types
```

출력:

```
{
    "AdjustmentTypes": [
        {
            "AdjustmentType": "ChangeInCapacity"
        },
        {
            "AdjustmentType": "ExactCapacity"
        },
        {
            "AdjustmentType": "PercentChangeInCapacity"
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 조정 유형 조정을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeAdjustmentTypes의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 조정 유형을 설명합니다.

```
Get-ASAdjustmentType
```

출력:

```
Type
----
ChangeInCapacity
ExactCapacity
PercentChangeInCapacity
```

- API 자세한 내용은 Cmdlet 참조DescribeAdjustmentTypes의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeAutoScalingGroups**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeAutoScalingGroups의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기
- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
```

```
            }
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
_amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);
```

```
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
            client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

예 1: 지정된 Auto Scaling 그룹을 설명하는 방법

이 예시에서는 지정된 Auto Scaling 그룹을 설명합니다.

```
aws autoscaling describe-auto-scaling-groups \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "my-asg",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
            "LaunchTemplate": {
                "LaunchTemplateName": "my-launch-template",
                "Version": "1",
                "LaunchTemplateId": "lt-1234567890abcde12"
            },
```

```
            "MinSize": 0,
            "MaxSize": 1,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-west-2a",
                "us-west-2b",
                "us-west-2c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-06905f55584de02da",
                    "InstanceType": "t2.micro",
                    "AvailabilityZone": "us-west-2a",
                    "HealthStatus": "Healthy",
                    "LifecycleState": "InService",
                    "ProtectedFromScaleIn": false,
                    "LaunchTemplate": {
                        "LaunchTemplateName": "my-launch-template",
                        "Version": "1",
                        "LaunchTemplateId": "lt-1234567890abcde12"
                    }
                }
            ],
            "CreatedTime": "2023-10-28T02:39:22.152Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-
c934b782",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN":"arn",
            "TrafficSources": []
        }
    ]
}
```

예 2: 처음 100개의 지정된 Auto Scaling 그룹을 설명하는 방법

이 예시에서는 지정된 Auto Scaling 그룹을 설명합니다. 최대 100개의 그룹 이름을 지정할 수 있습니다.

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 100 \
    --auto-scaling-group-name "group1" "group2" "group3" "group4"
```

샘플 출력은 예 1을 참조하세요.

예 3: 지정된 리전에서 Auto Scaling 그룹을 설명하는 방법

이 예시에서는 지정된 리전의 Auto Scaling 그룹을 최대 75개까지 설명합니다.

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 75 \
    --region us-east-1
```

샘플 출력은 예 1을 참조하세요.

예 4: 지정된 개수의 Auto Scaling 그룹을 설명하는 방법

특정 개수의 Auto Scaling 그룹을 반환하려면 `--max-items` 옵션을 사용하세요.

```
aws autoscaling describe-auto-scaling-groups \
    --max-items 1
```

샘플 출력은 예 1을 참조하세요.

출력에 `NextToken` 필드가 포함된 경우 그룹이 더 많습니다. 추가 그룹을 가져오려면 다음과 같이 후속 직접 호출에서 이 필드의 값을 `--starting-token` 옵션과 함께 사용하세요.

```
aws autoscaling describe-auto-scaling-groups \
    --starting-token Z3M3LMPEXAMPLE
```

샘플 출력은 예 1을 참조하세요.

예제 5: 시작 구성을 사용하는 Auto Scaling 그룹 설명

이 예제에서는 `--query` 옵션을 사용하여 시작 구성을 사용하는 Auto Scaling 그룹을 설명합니다.

```
aws autoscaling describe-auto-scaling-groups \
    --query 'AutoScalingGroups[?LaunchConfigurationName!=`null`]'
```

출력:

```
[
    {
        "AutoScalingGroupName": "my-asg",
        "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-
a11a-7b0acd480f03:autoScalingGroupName/my-asg",
        "LaunchConfigurationName": "my-lc",
        "MinSize": 0,
        "MaxSize": 1,
        "DesiredCapacity": 1,
        "DefaultCooldown": 300,
        "AvailabilityZones": [
            "us-west-2a",
            "us-west-2b",
            "us-west-2c"
        ],
        "LoadBalancerNames": [],
        "TargetGroupARNs": [],
        "HealthCheckType": "EC2",
        "HealthCheckGracePeriod": 0,
        "Instances": [
            {
                "InstanceId": "i-088c57934a6449037",
                "InstanceType": "t2.micro",
                "AvailabilityZone": "us-west-2c",
                "HealthStatus": "Healthy",
                "LifecycleState": "InService",
                "LaunchConfigurationName": "my-lc",
                "ProtectedFromScaleIn": false
            }
        ],
        "CreatedTime": "2023-10-28T02:39:22.152Z",
        "SuspendedProcesses": [],
        "VPCZoneIdentifier": "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782",
        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
```

```
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN":"arn",
        "TrafficSources": []
    }
]
```

자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 출력 필터링 AWS CLI을 참조하세요.

- 자세한 API 내용은 명령 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.autoscaling.AutoScalingClient;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingException;
import software.amazon.awssdk.services.autoscaling.model.AutoScalingGroup;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsResponse;
import
 software.amazon.awssdk.services.autoscaling.model.DescribeAutoScalingGroupsRequest;
import software.amazon.awssdk.services.autoscaling.model.Instance;
import java.util.List;

/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeAutoScalingInstances {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String instanceId = getAutoScaling(autoScalingClient, groupName);
        System.out.println(instanceId);
        autoScalingClient.close();
    }

    public static String getAutoScaling(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());

                List<Instance> instances = group.instances();
```

```
                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                }
            }
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getAutoScalingGroups(groupName: String) {
    val scalingGroupsRequest =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingGroups(scalingGroupsRequest)
        response.autoScalingGroups?.forEach { group ->
            println("The group name is ${group.autoScalingGroupName}")
            println("The group ARN is ${group.autoScalingGroupArn}")
            group.instances?.forEach { instance ->
                println("The instance id is ${instance.instanceId}")
```

```
                    println("The lifecycle state is " + instance.lifecycleState)
                }
            }
        }
    }
```

- API 자세한 내용은 DescribeAutoScalingGroups의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function describeAutoScalingGroups($autoScalingGroupNames)
    {
        return $this->autoScalingClient->describeAutoScalingGroups([
            'AutoScalingGroupNames' => $autoScalingGroupNames
        ]);
    }
```

- 자세한 API 내용은 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling 그룹의 이름을 나열합니다.

```
Get-ASAutoScalingGroup | format-table -property AutoScalingGroupName
```

출력:

```
AutoScalingGroupName
--------------------
my-asg-1
my-asg-2
my-asg-3
my-asg-4
my-asg-5
my-asg-6
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹을 설명합니다.

```
Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1
```

출력:

```
AutoScalingGroupARN     : arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:930d940e-891e-4781-a11a-7b0acd480
                          f03:autoScalingGroupName/my-asg-1
AutoScalingGroupName    : my-asg-1
AvailabilityZones       : {us-west-2b, us-west-2a}
CreatedTime             : 3/1/2015 9:05:31 AM
DefaultCooldown         : 300
DesiredCapacity         : 2
EnabledMetrics          : {}
HealthCheckGracePeriod  : 300
HealthCheckType         : EC2
Instances               : {my-lc}
LaunchConfigurationName : my-lc
LoadBalancerNames       : {}
MaxSize                 : 0
MinSize                 : 0
PlacementGroup          :
Status                  :
SuspendedProcesses      : {}
Tags                    : {}
TerminationPolicies     : {Default}
VPCZoneIdentifier       : subnet-e4f33493,subnet-5264e837
```

예제 3: 이 예제에서는 지정된 두 Auto Scaling 그룹을 설명합니다.

```
Get-ASAutoScalingGroup -AutoScalingGroupName @("my-asg-1", "my-asg-2")
```

예제 4: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 Auto Scaling 인스턴스를 설명합니다.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg-1).Instances
```

예제 5: 이 예제에서는 모든 Auto Scaling 그룹을 설명합니다.

```
Get-ASAutoScalingGroup
```

예제 6: 이 예제에서는 모든 Auto Scaling 그룹을 10개의 배치로 설명합니다.

```
$nextToken = $null
do {
  Get-ASAutoScalingGroup -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

예제 7: 이 예제에서는 지정된 Auto Scaling 그룹에 LaunchTemplate 대해 설명합니다. 이 예제에서는 '인스턴스 구매 옵션'이 '템플릿을 시작하려면 애더리'로 설정되어 있다고 가정합니다. 이 옵션이 “구매 옵션 및 인스턴스 유형 결합”으로 설정된 경우 'MixedInstancesPolicy.' LaunchTemplate속성을 사용하여 액세스할 수 LaunchTemplate 있습니다.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-ag-1).LaunchTemplate
```

출력:

```
LaunchTemplateId     LaunchTemplateName   Version
----------------     ------------------   -------
lt-06095fd619cb40371 test-launch-template $Default
```

- 자세한 API 내용은 Cmdlet 참조DescribeAutoScalingGroups의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_group(self, group_name: str) -> Optional[Dict[str, Any]]:
        """
        Gets information about an Auto Scaling group.

        :param group_name: The name of the group to look up.
        :return: A dictionary with information about the group if found,
 otherwise None.
        :raises ClientError: If there is an error describing the Auto Scaling
 group.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_groups"
            )
            response_iterator =
 paginator.paginate(AutoScalingGroupNames=[group_name])
            groups = []
            for response in response_iterator:
                groups.extend(response.get("AutoScalingGroups", []))

            logger.info(
```

```
                f"Successfully retrieved information for Auto Scaling group
{group_name}."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to describe Auto Scaling group {group_name}.")
            if error_code == "ResourceContentionFault":
                logger.error(
                    "There is a conflict with another operation that is modifying
the "
                    f"Auto Scaling group '{group_name}' Please try again later."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
        else:
            return groups[0] if len(groups) > 0 else None
```

- API 자세한 내용은 DescribeAutoScalingGroups의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn list_groups(client: &Client) -> Result<(), Error> {
    let resp = client.describe_auto_scaling_groups().send().await?;

    println!("Groups:");

    let groups = resp.auto_scaling_groups();

    for group in groups {
```

```
        println!(
            "Name:  {}",
            group.auto_scaling_group_name().unwrap_or("Unknown")
        );
        println!(
            "Arn:   {}",
            group.auto_scaling_group_arn().unwrap_or("unknown"),
        );
        println!("Zones: {:?}", group.availability_zones(),);
        println!();
    }

    println!("Found {} group(s)", groups.len());

    Ok(())
}
```

- API 자세한 내용은 Rust 참조 DescribeAutoScalingGroups 의 섹션을 참조하세요. AWS SDK API

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **`DescribeAutoScalingInstances`**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DescribeAutoScalingInstances`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Get data about the instances in an Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling details.</returns>
    public async Task<List<AutoScalingInstanceDetails>>
 DescribeAutoScalingInstancesAsync(
        string groupName)
    {
        var groups = await DescribeAutoScalingGroupsAsync(groupName);
        var instanceIds = new List<string>();
        groups!.ForEach(group =>
        {
            if (group.AutoScalingGroupName == groupName)
            {
                group.Instances.ForEach(instance =>
                {
                    instanceIds.Add(instance.InstanceId);
                });
            }
        });

        var scalingGroupsRequest = new DescribeAutoScalingInstancesRequest
        {
            MaxRecords = 10,
            InstanceIds = instanceIds,
        };

        var response = await
 _amazonAutoScaling.DescribeAutoScalingInstancesAsync(scalingGroupsRequest);
```

```
        var instanceDetails = response.AutoScalingInstances;

        return instanceDetails;
    }
```

- 자세한 API 내용은 참조DescribeAutoScalingInstances의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
        request.SetInstanceIds(instanceIDs);

        Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
                client.DescribeAutoScalingInstances(request);

        if (outcome.IsSuccess()) {
            const
Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                    outcome.GetResult().GetAutoScalingInstances();

        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
```

```
                        << outcome.GetError().GetMessage()
                        << std::endl;
            return false;
        }
```

- 자세한 API 내용은 참조DescribeAutoScalingInstances의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

예 1: 하나 이상의 인스턴스를 설명하는 방법

이 예시에서는 지정된 인스턴스를 설명합니다.

```
aws autoscaling describe-auto-scaling-instances \
    --instance-ids i-06905f55584de02da
```

출력:

```
{
    "AutoScalingInstances": [
        {
            "InstanceId": "i-06905f55584de02da",
            "InstanceType": "t2.micro",
            "AutoScalingGroupName": "my-asg",
            "AvailabilityZone": "us-west-2b",
            "LifecycleState": "InService",
            "HealthStatus": "HEALTHY",
            "ProtectedFromScaleIn": false,
            "LaunchTemplate": {
                "LaunchTemplateId": "lt-1234567890abcde12",
                "LaunchTemplateName": "my-launch-template",
                "Version": "1"
            }
        }
    ]
}
```

예 2: 하나 이상의 인스턴스를 설명하는 방법

이 예시에서는 `--max-items` 옵션을 사용하여 이 직접 호출과 함께 반환할 인스턴스 수를 지정합니다.

```
aws autoscaling describe-auto-scaling-instances \
    --max-items 1
```

출력에 `NextToken` 필드가 포함된 경우 인스턴스가 더 많습니다. 추가 인스턴스를 가져오려면 다음과 같이 후속 직접 호출에서 이 필드의 값을 `--starting-token` 옵션과 함께 사용하세요.

```
aws autoscaling describe-auto-scaling-instances \
    --starting-token Z3M3LMPEXAMPLE
```

샘플 출력은 예 1을 참조하세요.

예제 3: 시작 구성을 사용하는 인스턴스 설명

이 예제에서는 `--query` 옵션을 사용하여 시작 구성을 사용하는 인스턴스를 설명합니다.

```
aws autoscaling describe-auto-scaling-instances \
    --query 'AutoScalingInstances[?LaunchConfigurationName!=`null`]'
```

출력:

```
[
    {
        "InstanceId": "i-088c57934a6449037",
        "InstanceType": "t2.micro",
        "AutoScalingGroupName": "my-asg",
        "AvailabilityZone": "us-west-2c",
        "LifecycleState": "InService",
        "HealthStatus": "HEALTHY",
        "LaunchConfigurationName": "my-lc",
        "ProtectedFromScaleIn": false
    }
]
```

자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 출력 필터링 AWS CLI을 참조하세요.

- 자세한 API 내용은 명령 참조DescribeAutoScalingInstances의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
       try {
           DescribeAutoScalingInstancesRequest
describeAutoScalingInstancesRequest = DescribeAutoScalingInstancesRequest
                   .builder()
                   .instanceIds(id)
                   .build();

           DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
           List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
           for (AutoScalingInstanceDetails instance : instances) {
               System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
           }

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 자세한 API 내용은 참조DescribeAutoScalingInstances의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAutoScalingInstance(id: String) {
    val describeAutoScalingInstancesRequest =
        DescribeAutoScalingInstancesRequest {
            instanceIds = listOf(id)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response =
 autoScalingClient.describeAutoScalingInstances(describeAutoScalingInstancesRequest)
        response.autoScalingInstances?.forEach { group ->
            println("The instance lifecycle state is: ${group.lifecycleState}")
        }
    }
}
```

- API 자세한 내용은 DescribeAutoScalingInstances의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function describeAutoScalingInstances($instanceIds)
```

```
    {
        return $this->autoScalingClient->describeAutoScalingInstances([
            'InstanceIds' => $instanceIds
        ]);
    }
```

- 자세한 API 내용은 참조DescribeAutoScalingInstances의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling 인스턴스IDs의 를 나열합니다.

```
Get-ASAutoScalingInstance | format-table -property InstanceId
```

출력:

```
InstanceId
----------
i-12345678
i-87654321
i-abcd1234
```

예제 2: 이 예제에서는 지정된 Auto Scaling 인스턴스를 설명합니다.

```
Get-ASAutoScalingInstance -InstanceId i-12345678
```

출력:

```
AutoScalingGroupName    : my-asg
AvailabilityZone        : us-west-2b
HealthStatus            : HEALTHY
InstanceId              : i-12345678
LaunchConfigurationName : my-lc
LifecycleState          : InService
```

예제 3: 이 예제에서는 지정된 두 Auto Scaling 인스턴스를 설명합니다.

```
Get-ASAutoScalingInstance -InstanceId @("i-12345678", "i-87654321")
```

예제 4: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 Auto Scaling 인스턴스를 설명합니다.

```
(Get-ASAutoScalingGroup -AutoScalingGroupName my-asg).Instances | Get-
ASAutoScalingInstance
```

예제 5: 이 예제에서는 모든 Auto Scaling 인스턴스를 설명합니다.

```
Get-ASAutoScalingInstance
```

예제 6: 이 예제에서는 모든 Auto Scaling 인스턴스를 10개의 배치로 설명합니다.

```
$nextToken = $null
do {
  Get-ASAutoScalingInstance -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- API 자세한 내용은 Cmdlet 참조DescribeAutoScalingInstances의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
```

```
        """
        self.autoscaling_client = autoscaling_client


    def describe_instances(self, instance_ids: List[str]) -> List[Dict[str,
Any]]:
        """
        Gets information about instances.

        :param instance_ids: A list of instance IDs to look up.
        :return: A list of dictionaries with information about each instance,
                 or an empty list if none are found.
        :raises ClientError: If there is an error describing the instances.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_auto_scaling_instances"
            )
            response_iterator = paginator.paginate(InstanceIds=instance_ids)

            instances = []
            for response in response_iterator:
                instances.extend(response.get("AutoScalingInstances", []))

            logger.info(f"Successfully described instances: {instance_ids}")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe instances {instance_ids}. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )
            raise
        else:
            return instances
```

- API 자세한 내용은 DescribeAutoScalingInstances의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    pub async fn list_instances(&self) -> Result<Vec<String>, ScenarioError> {
        // The direct way to list instances is by using
 DescribeAutoScalingGroup's instances property. However, this returns a
 Vec<Instance>, as opposed to a Vec<AutoScalingInstanceDetails>.
        // Ok(self.get_group().await?.instances.unwrap_or_default().map(|
i| i.instance_id.clone().unwrap_or_default()).filter(|id| !
id.is_empty()).collect())

        // Alternatively, and for the sake of example,
 DescribeAutoScalingInstances returns a list that can be filtered by the client.
        self.autoscaling
            .describe_auto_scaling_instances()
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
            .map(|items| {
                items
                    .into_iter()
                    .filter(|i| {
                        i.auto_scaling_group_name.as_deref()
                            == Some(self.auto_scaling_group_name.as_str())
                    })
                    .map(|i| i.instance_id.unwrap_or_default())
                    .filter(|id| !id.is_empty())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Failed to get list of auto scaling
 instances", &err))
    }
```

- API 자세한 내용은 Rust 참조 DescribeAutoScalingInstances 의 섹션을 참조하세요. AWS SDK API

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeAutoScalingNotificationTypes**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DescribeAutoScalingNotificationTypes`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용 가능한 알림 유형을 설명하려면

이 예제에서는 사용 가능한 알림 유형을 설명합니다.

```
aws autoscaling describe-auto-scaling-notification-types
```

출력:

```
{
    "AutoScalingNotificationTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCH",
        "autoscaling:EC2_INSTANCE_LAUNCH_ERROR",
        "autoscaling:EC2_INSTANCE_TERMINATE",
        "autoscaling:EC2_INSTANCE_TERMINATE_ERROR",
        "autoscaling:TEST_NOTIFICATION"
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹이 확장될 때 Amazon SNS 알림 받기를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeAutoScalingNotificationTypes의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 알림 유형을 나열합니다.

```
Get-ASAutoScalingNotificationType
```

출력:

```
autoscaling:EC2_INSTANCE_LAUNCH
autoscaling:EC2_INSTANCE_LAUNCH_ERROR
autoscaling:EC2_INSTANCE_TERMINATE
autoscaling:EC2_INSTANCE_TERMINATE_ERROR
autoscaling:TEST_NOTIFICATION
```

- 자세한 API 내용은 Cmdlet 참조DescribeAutoScalingNotificationTypes의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeLaunchConfigurations**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DescribeLaunchConfigurations`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 지정된 시작 구성을 설명하려면

이 예제에서는 지정된 시작 구성을 설명합니다.

```
aws autoscaling describe-launch-configurations \
    --launch-configuration-names my-launch-config
```

출력:

```
{
    "LaunchConfigurations": [
        {
```

```
            "LaunchConfigurationName": "my-launch-config",
            "LaunchConfigurationARN": "arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:98d3b196-4cf9-4e88-8ca1-8547c24ced8b:launchConfig
my-launch-config",
            "ImageId": "ami-0528a5175983e7f28",
            "KeyName": "my-key-pair-uswest2",
            "SecurityGroups": [
                "sg-05eaec502fcdadc2e"
            ],
            "ClassicLinkVPCSecurityGroups": [],
            "UserData": "",
            "InstanceType": "t2.micro",
            "KernelId": "",
            "RamdiskId": "",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                        "SnapshotId": "snap-06c1606ba5ca274b1",
                        "VolumeSize": 8,
                        "VolumeType": "gp2",
                        "DeleteOnTermination": true,
                        "Encrypted": false
                    }
                }
            ],
            "InstanceMonitoring": {
                "Enabled": true
            },
            "CreatedTime": "2020-10-28T02:39:22.321Z",
            "EbsOptimized": false,
            "AssociatePublicIpAddress": true,
            "MetadataOptions": {
                "HttpTokens": "required",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "disabled"
            }
        }
    ]
}
```

예제 2: 지정된 수의 시작 구성을 설명하려면

특정 수의 시작 구성을 반환하려면 `--max-items` 옵션을 사용합니다.

```
aws autoscaling describe-launch-configurations \
    --max-items 1
```

출력에 NextToken 필드가 포함된 경우 시작 구성이 더 많습니다. 추가 시작 구성을 가져오려면 다음과 같이 후속 통화에서 --starting-token 옵션과 함께 이 필드의 값을 사용합니다.

```
aws autoscaling describe-launch-configurations \
    --starting-token Z3M3LMPEXAMPLE
```

- 자세한 API 내용은 명령 참조DescribeLaunchConfigurations의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 시작 구성의 이름을 나열합니다.

```
Get-ASLaunchConfiguration | format-table -property LaunchConfigurationName
```

출력:

```
LaunchConfigurationName
-----------------------
my-lc-1
my-lc-2
my-lc-3
my-lc-4
my-lc-5
```

예제 2: 이 예제에서는 지정된 시작 구성을 설명합니다.

```
Get-ASLaunchConfiguration -LaunchConfigurationName my-lc-1
```

출력:

```
AssociatePublicIpAddress     : True
BlockDeviceMappings          : {/dev/xvda}
ClassicLinkVPCId             :
```

```
ClassicLinkVPCSecurityGroups : {}
CreatedTime                  : 12/12/2014 3:22:08 PM
EbsOptimized                 : False
IamInstanceProfile           :
ImageId                      : ami-043a5034
InstanceMonitoring           : Amazon.AutoScaling.Model.InstanceMonitoring
InstanceType                 : t2.micro
KernelId                     :
KeyName                      :
LaunchConfigurationARN       : arn:aws:autoscaling:us-
west-2:123456789012:launchConfiguration:7e5f31e4-693b-4604-9322-
                               e6f68d7fafad:launchConfigurationName/my-lc-1
LaunchConfigurationName      : my-lc-1
PlacementTenancy             :
RamdiskId                    :
SecurityGroups               : {sg-67ef0308}
SpotPrice                    :
UserData                     :
```

예제 3: 이 예제에서는 지정된 두 가지 시작 구성을 설명합니다.

```
Get-ASLaunchConfiguration -LaunchConfigurationName @("my-lc-1", "my-lc-2")
```

예제 4: 이 예제에서는 모든 시작 구성을 설명합니다.

```
Get-ASLaunchConfiguration
```

예제 5: 이 예제에서는 모든 시작 구성을 10개 배치로 설명합니다.

```
$nextToken = $null
do {
  Get-ASLaunchConfiguration -NextToken $nextToken -MaxRecord 10
  $nextToken = $AWSHistory.LastServiceResponse.NextToken
} while ($nextToken -ne $null)
```

- 자세한 API 내용은 Cmdlet 참조DescribeLaunchConfigurations의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeLifecycleHookTypes**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeLifecycleHookTypes의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용 가능한 수명 주기 후크 유형을 설명하려면

이 예제에서는 사용 가능한 수명 주기 후크 유형을 설명합니다.

```
aws autoscaling describe-lifecycle-hook-types
```

출력:

```
{
    "LifecycleHookTypes": [
        "autoscaling:EC2_INSTANCE_LAUNCHING",
        "autoscaling:EC2_INSTANCE_TERMINATING"
    ]
}
```

- 자세한 API 내용은 명령 참조DescribeLifecycleHookTypes의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 수명 주기 후크 유형을 나열합니다.

```
Get-ASLifecycleHookType
```

출력:

```
autoscaling:EC2_INSTANCE_LAUNCHING
auto-scaling:EC2_INSTANCE_TERMINATING
```

- API 자세한 내용은 Cmdlet 참조DescribeLifecycleHookTypes의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeLifecycleHooks**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeLifecycleHooks의 사용 방법을 보여 줍니다.

CLI

AWS CLI

수명 주기 후크를 설명하려면

이 예제에서는 지정된 Auto Scaling 그룹의 수명 주기 후크를 설명합니다.

```
aws autoscaling describe-lifecycle-hooks \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "LifecycleHooks": [
        {
            "GlobalTimeout": 3000,
            "HeartbeatTimeout": 30,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-launch-hook",
            "DefaultResult": "ABANDON",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_LAUNCHING"
        },
        {
            "GlobalTimeout": 6000,
            "HeartbeatTimeout": 60,
            "AutoScalingGroupName": "my-asg",
            "LifecycleHookName": "my-termination-hook",
            "DefaultResult": "CONTINUE",
            "LifecycleTransition": "autoscaling:EC2_INSTANCE_TERMINATING"
        }
    ]
}
```

- 자세한 API 내용은 명령 참조DescribeLifecycleHooks의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 수명 주기 후크를 설명합니다.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook
```

출력:

```
AutoScalingGroupName  : my-asg
DefaultResult         : ABANDON
GlobalTimeout         : 172800
HeartbeatTimeout      : 3600
LifecycleHookName     : myLifecycleHook
LifecycleTransition   : auto-scaling:EC2_INSTANCE_LAUNCHING
NotificationMetadata  :
NotificationTargetARN : arn:aws:sns:us-west-2:123456789012:my-topic
RoleARN               : arn:aws:iam::123456789012:role/my-iam-role
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹의 모든 수명 주기 후크를 설명합니다.

```
Get-ASLifecycleHook -AutoScalingGroupName my-asg
```

예제 3: 이 예제에서는 모든 Auto Scaling 그룹의 모든 수명 주기 후크를 설명합니다.

```
Get-ASLifecycleHook
```

- API 자세한 내용은 Cmdlet 참조DescribeLifecycleHooks의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeLoadBalancers**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeLoadBalancers의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹의 Classic Load Balancer를 설명하려면

이 예제에서는 지정된 Auto Scaling 그룹의 Classic Load Balancer에 대해 설명합니다.

```
aws autoscaling describe-load-balancers \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "LoadBalancers": [
        {
            "State": "Added",
            "LoadBalancerName": "my-load-balancer"
        }
    ]
}
```

- 자세한 API 내용은 명령 참조DescribeLoadBalancers의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹의 로드 밸런서를 설명합니다.

```
Get-ASLoadBalancer -AutoScalingGroupName my-asg
```

출력:

```
LoadBalancerName    State
----------------    -----
my-lb               Added
```

- API 자세한 내용은 Cmdlet 참조DescribeLoadBalancers의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeMetricCollectionTypes**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeMetricCollectionTypes의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용 가능한 지표 컬렉션 유형을 설명하려면

이 예제에서는 사용 가능한 지표 컬렉션 유형을 설명합니다.

```
aws autoscaling describe-metric-collection-types
```

출력:

```
{
    "Metrics": [
        {
            "Metric": "GroupMinSize"
        },
        {
            "Metric": "GroupMaxSize"
        },
        {
            "Metric": "GroupDesiredCapacity"
        },
        {
            "Metric": "GroupInServiceInstances"
        },
        {
            "Metric": "GroupInServiceCapacity"
        },
        {
            "Metric": "GroupPendingInstances"
        },
        {
            "Metric": "GroupPendingCapacity"
        },
        {
```

```
            "Metric": "GroupTerminatingInstances"
        },
        {
            "Metric": "GroupTerminatingCapacity"
        },
        {
            "Metric": "GroupStandbyInstances"
        },
        {
            "Metric": "GroupStandbyCapacity"
        },
        {
            "Metric": "GroupTotalInstances"
        },
        {
            "Metric": "GroupTotalCapacity"
        }
    ],
    "Granularities": [
        {
            "Granularity": "1Minute"
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 지표를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeMetricCollectionTypes의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 지표 컬렉션 유형을 나열합니다.

```
(Get-ASMetricCollectionType).Metrics
```

출력:

```
Metric
------
```

```
GroupMinSize
GroupMaxSize
GroupDesiredCapacity
GroupInServiceInstances
GroupPendingInstances
GroupTerminatingInstances
GroupStandbyInstances
GroupTotalInstances
```

예제 2: 이 예제에서는 해당 세분화를 나열합니다.

```
(Get-ASMetricCollectionType).Granularities
```

출력:

```
Granularity
-----------
1Minute
```

- API 자세한 내용은 Cmdlet 참조DescribeMetricCollectionTypes의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeNotificationConfigurations**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DescribeNotificationConfigurations`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 지정된 그룹의 알림 구성을 설명하려면

이 예제에서는 지정된 Auto Scaling 그룹에 대한 알림 구성을 설명합니다.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹이 확장될 때 Amazon SNS 알림 받기를 참조하세요. EC2 Auto Scaling

예제 1: 지정된 수의 알림 구성을 설명하려면

특정 수의 알림 구성을 반환하려면 `max-items` 파라미터를 사용합니다.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-auto-scaling-group \
    --max-items 1
```

출력:

```
{
    "NotificationConfigurations": [
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic-2"
        },
        {
            "AutoScalingGroupName": "my-asg",
            "NotificationType": "autoscaling:TEST_NOTIFICATION",
            "TopicARN": "arn:aws:sns:us-west-2:123456789012:my-sns-topic"
        }
    ]
}
```

출력에 NextToken 필드가 포함된 경우 알림 구성이 더 많습니다. 추가 알림 구성을 가져오려면 다음과 같이 후속 호출에서 starting-token 파라미터와 함께 이 필드의 값을 사용합니다.

```
aws autoscaling describe-notification-configurations \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹이 조정될 때 Amazon SNS 알림 받기를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeNotificationConfigurations의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹과 연결된 알림 작업에 대해 설명합니다.

```
Get-ASNotificationConfiguration -AutoScalingGroupName my-asg | format-list
```

출력:

```
AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_LAUNCH
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic

AutoScalingGroupName : my-asg
NotificationType     : auto-scaling:EC2_INSTANCE_TERMINATE
TopicARN             : arn:aws:sns:us-west-2:123456789012:my-topic
```

예제 2: 이 예제에서는 모든 Auto Scaling 그룹과 연결된 알림 작업을 설명합니다.

```
Get-ASNotificationConfiguration
```

- API 자세한 내용은 Cmdlet 참조DescribeNotificationConfigurations의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribePolicies**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribePolicies의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 지정된 그룹의 조정 정책을 설명하려면

이 예제에서는 지정된 Auto Scaling 그룹에 대한 조정 정책을 설명합니다.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "ScalingPolicies": [
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "alb1000-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:3065d9c8-9969-4bec-
bb6a-3fbe5550fde6:autoScalingGroupName/my-asg:policyName/alb1000-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-
AlarmHigh-924887a9-12d7-4e01-8686-6f844d13a196"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-f96f899d-
b8e7-4d09-a010-c1aaa35da296",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-f96f899d-b8e7-4d09-a010-
c1aaa35da296"
                }
```

```
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ALBRequestCountPerTarget",
                    "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-
alb-target-group/943f017f100becff"
                },
                "TargetValue": 1000.0,
                "DisableScaleIn": false
            },
            "Enabled": true
        },
        {
            "AutoScalingGroupName": "my-asg",
            "PolicyName": "cpu40-target-tracking-scaling-policy",
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:123456789012:scalingPolicy:5fd26f71-39d4-4690-82a9-
b8515c45cdde:autoScalingGroupName/my-asg:policyName/cpu40-target-tracking-
scaling-policy",
            "PolicyType": "TargetTrackingScaling",
            "StepAdjustments": [],
            "Alarms": [
                {
                    "AlarmName": "TargetTracking-my-asg-
AlarmHigh-139f9789-37b9-42ad-bea5-b5b147d7f473",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmHigh-139f9789-37b9-42ad-
bea5-b5b147d7f473"
                },
                {
                    "AlarmName": "TargetTracking-my-asg-AlarmLow-bd681c67-
fc18-4c56-8468-fb8e413009c9",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:123456789012:alarm:TargetTracking-my-asg-AlarmLow-bd681c67-fc18-4c56-8468-
fb8e413009c9"
                }
            ],
            "TargetTrackingConfiguration": {
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "ASGAverageCPUUtilization"
                },
                "TargetValue": 40.0,
                "DisableScaleIn": false
            },
```

```
            "Enabled": true
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 동적 조정을 참조하세요. EC2 Auto Scaling

예제 2: 지정된 이름의 조정 정책을 설명하는 방법

특정 조정 정책을 반환하려면 `--policy-names` 옵션을 사용합니다.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --policy-names cpu40-target-tracking-scaling-policy
```

샘플 출력은 예 1을 참조하세요.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 동적 조정을 참조하세요. EC2 Auto Scaling

예제 3: 여러 조정 정책 설명

특정 수의 정책을 반환하려면 `--max-items` 옵션을 사용합니다.

```
aws autoscaling describe-policies \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

샘플 출력은 예 1을 참조하세요.

출력에 `NextToken` 필드가 포함된 경우 이 필드의 값을 후속 호출의 `--starting-token` 옵션과 함께 사용하여 추가 정책을 가져옵니다.

```
aws autoscaling describe-policies --auto-scaling-group-name my-asg --starting-
token Z3M3LMPEXAMPLE
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 동적 조정을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribePolicies의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 정책을 설명합니다.

```
Get-ASPolicy -AutoScalingGroupName my-asg
```

출력:

```
AdjustmentType          : ChangeInCapacity
Alarms                  : {}
AutoScalingGroupName    : my-asg
Cooldown                : 0
EstimatedInstanceWarmup : 0
MetricAggregationType   :
MinAdjustmentMagnitude  : 0
MinAdjustmentStep       : 0
PolicyARN               : arn:aws:auto-scaling:us-
west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-e1d769fc24ef
                          :autoScalingGroupName/my-asg:policyName/myScaleInPolicy
PolicyName              : myScaleInPolicy
PolicyType              : SimpleScaling
ScalingAdjustment       : -1
StepAdjustments         : {}
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 정책을 설명합니다.

```
Get-ASPolicy -AutoScalingGroupName my-asg -PolicyName @("myScaleOutPolicy",
 "myScaleInPolicy")
```

예제 3: 이 예제에서는 모든 Auto Scaling 그룹에 대한 모든 정책을 설명합니다.

```
Get-ASPolicy
```

- 자세한 API 내용은 Cmdlet 참조DescribePolicies의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeScalingActivities**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeScalingActivities의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

AWS SDK for .NET

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Retrieve a list of the Amazon EC2 Auto Scaling activities for an
    /// Amazon EC2 Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Amazon EC2 Auto Scaling group.</
param>
    /// <returns>A list of Amazon EC2 Auto Scaling activities.</returns>
    public async Task<List<Amazon.AutoScaling.Model.Activity>>
 DescribeScalingActivitiesAsync(
        string groupName)
    {
        var scalingActivitiesRequest = new DescribeScalingActivitiesRequest
        {
            AutoScalingGroupName = groupName,
            MaxRecords = 10,
        };

        var response = await
 _amazonAutoScaling.DescribeScalingActivitiesAsync(scalingActivitiesRequest);
        return response.Activities;
    }
```

- 자세한 API 내용은 참조DescribeScalingActivities의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
        Aws::String nextToken; // Used for pagination;
        do {
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
            Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
                    autoScalingClient.DescribeScalingActivities(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities
=
                        outcome.GetResult().GetActivities();
                allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
                nextToken  = outcome.GetResult().GetNextToken();
```

```
            }
            else {
                std::cerr << "Error with AutoScaling::DescribeScalingActivities.
 "
                          << outcome.GetError().GetMessage()
                          << std::endl;

            }
        } while (!nextToken.empty());

        std::cout << "Found " << allActivities.size() << " activities."
                  << std::endl;
        std::cout << "Activities are ordered with the most recent first."
                  << std::endl;
        for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
            std::cout << activity.GetDescription() << std::endl;
            std::cout << activity.GetDetails() << std::endl;
        }
```

- 자세한 API 내용은 참조DescribeScalingActivities의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

예 1: 지정된 그룹에 대한 크기 조정 활동을 설명하는 방법

이 예시에서는 지정된 Auto Scaling 그룹에 대한 크기 조정 활동을 설명합니다.

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
```

```
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹에 대한 조정 활동 확인을 참조하세요. EC2 Auto Scaling

예 2: 삭제된 그룹에 대한 크기 조정 활동을 설명하는 방법

Auto Scaling 그룹이 삭제된 후 크기 조정 활동을 설명하려면 `--include-deleted-groups` 옵션을 추가하세요.

```
aws autoscaling describe-scaling-activities \
    --auto-scaling-group-name my-asg \
    --include-deleted-groups
```

출력:

```
{
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "Description": "Launching a new EC2 instance.  Status Reason: Your
 Spot request price of 0.001 is lower than the minimum required Spot request
 fulfillment price of 0.0031. Launching EC2 instance failed.",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of
 AutoScalingGroup constraints to min: 1, max: 5, desired: 3 changing the desired
 capacity from 0 to 3.  At 2021-01-13T20:47:27Z an instance was started in
 response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 3.",
```

```
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than
 the minimum required Spot request fulfillment price of 0.0031. Launching EC2
 instance failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        }
    ]
}
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 문제 해결을 참조하세요. EC2 Auto Scaling

예 3: 지정된 개수의 크기 조정 활동을 설명하는 방법

특정 개수의 활동을 반환하려면 `--max-items` 옵션을 사용하세요.

```
aws autoscaling describe-scaling-activities \
    --max-items 1
```

출력:

```
{
    "Activities": [
        {
            "ActivityId": "f9f2d65b-f1f2-43e7-b46d-d86756459699",
            "Description": "Launching a new EC2 instance: i-0d44425630326060f",
            "AutoScalingGroupName": "my-asg",
            "Cause": "At 2020-10-30T19:35:51Z a user request update of
 AutoScalingGroup constraints to min: 0, max: 16, desired: 16 changing the
 desired capacity from 0 to 16.  At 2020-10-30T19:36:07Z an instance was started
 in response to a difference between desired and actual capacity, increasing the
 capacity from 0 to 16.",
            "StartTime": "2020-10-30T19:36:09.766Z",
            "EndTime": "2020-10-30T19:36:41Z",
            "StatusCode": "Successful",
```

```
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":
\"us-west-2b\"}"
        }
    ]
}
```

출력에 `NextToken` 필드가 포함된 경우 활동이 더 많습니다. 추가 활동을 가져오려면 다음과 같이 후속 직접 호출에서 이 필드의 값을 `--starting-token` 옵션과 함께 사용하세요.

```
aws autoscaling describe-scaling-activities \
    --starting-token Z3M3LMPEXAMPLE
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹에 대한 조정 활동 확인을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeScalingActivities의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                    .autoScalingGroupName(groupName)
                    .maxRecords(10)
                    .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                    .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
```

```
                System.out.println("The activity Id is " +
 activity.activityId());
                System.out.println("The activity details are " +
 activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 자세한 API 내용은 참조DescribeScalingActivities의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun describeAutoScalingGroups(groupName: String) {
    val groupsReques =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
            maxRecords = 10
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        val response = autoScalingClient.describeAutoScalingGroups(groupsReques)
        response.autoScalingGroups?.forEach { group ->
            println("The service to use for the health checks:
 ${group.healthCheckType}")
        }
    }
}
```

- API 자세한 내용은 DescribeScalingActivities의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public function describeScalingActivities($autoScalingGroupName)
{
    return $this->autoScalingClient->describeScalingActivities([
        'AutoScalingGroupName' => $autoScalingGroupName,
    ]);
}
```

- 자세한 API 내용은 참조DescribeScalingActivities의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 지난 6주 동안의 조정 활동을 설명합니다.

```
Get-ASScalingActivity -AutoScalingGroupName my-asg
```

출력:

```
ActivityId           : 063308ae-aa22-4a9b-94f4-9fae4EXAMPLE
AutoScalingGroupName : my-asg
```

```
Cause                : At 2015-11-22T15:45:16Z a user request explicitly set
 group desired capacity changing the desired
                       capacity from 1 to 2.  At 2015-11-22T15:45:34Z an instance
 was started in response to a difference
                       between desired and actual capacity, increasing the
 capacity from 1 to 2.
Description          : Launching a new EC2 instance: i-26e715fc
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/22/2015 7:46:09 AM
Progress             : 100
StartTime            : 11/22/2015 7:45:35 AM
StatusCode           : Successful
StatusMessage        :

ActivityId           : ce719997-086d-4c73-a2f1-ab703EXAMPLE
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:57:53Z a user request created an
 AutoScalingGroup changing the desired capacity
                        from 0 to 1.  At 2015-11-20T22:57:58Z an instance was
 started in response to a difference betwe
                       en desired and actual capacity, increasing the capacity
 from 0 to 1.
Description          : Launching a new EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              : 11/20/2015 2:58:32 PM
Progress             : 100
StartTime            : 11/20/2015 2:57:59 PM
StatusCode           : Successful
StatusMessage        :
```

예제 2: 이 예제에서는 지정된 조정 활동을 설명합니다.

```
Get-ASScalingActivity -ActivityId "063308ae-aa22-4a9b-94f4-9fae4EXAMPLE"
```

예제 3: 이 예제에서는 모든 Auto Scaling 그룹에 대한 지난 6주 동안의 조정 활동을 설명합니다.

```
Get-ASScalingActivity
```

- API 자세한 내용은 Cmdlet 참조DescribeScalingActivities의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def describe_scaling_activities(self, group_name: str) -> List[Dict[str,
 Any]]:
        """
        Gets information about scaling activities for the group. Scaling
 activities
        are things like instances stopping or starting in response to user
 requests
        or capacity changes.

        :param group_name: The name of the group to look up.
        :return: A list of dictionaries representing the scaling activities for
 the
                 group, ordered with the most recent activity first.
        :raises ClientError: If there is an error describing the scaling
 activities.
        """
        try:
            paginator = self.autoscaling_client.get_paginator(
                "describe_scaling_activities"
            )
            response_iterator =
 paginator.paginate(AutoScalingGroupName=group_name)
```

```
            activities = []
            for response in response_iterator:
                activities.extend(response.get("Activities", []))

            logger.info(
                f"Successfully described scaling activities for group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't describe scaling activities for group '{group_name}'.
Error code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
        else:
            return activities
```

- API 자세한 내용은 DescribeScalingActivities의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에는 더 많은 기능이 있습니다 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    pub async fn describe_scenario(&self) -> AutoScalingScenarioDescription {
        let group = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await
            .map(|s| {
                s.auto_scaling_groups()
                    .iter()
                    .map(|s| {
                        format!(
                            "{}: {}",
                            s.auto_scaling_group_name().unwrap_or("Unknown"),
                            s.status().unwrap_or("Unknown")
                        )
                    })
                    .collect::<Vec<String>>()
            })
            .map_err(|e| {
                ScenarioError::new("Failed to describe auto scaling groups for
scenario", &e)
            });

        let instances = self
            .list_instances()
            .await
            .map_err(|e| anyhow!("There was an error listing instances: {e}",));

        // 10. DescribeScalingActivities: list the scaling activities that have
occurred for the group so far.
        //   Bonus: use CloudWatch API to get and show some metrics collected for
the group.
        //   CW.ListMetrics with Namespace='AWS/AutoScaling' and
Dimensions=[{'Name': 'AutoScalingGroupName', 'Value': }]
        //   CW.GetMetricStatistics with Statistics='Sum'. Start and End times
must be in UTC!
        let activities = self
            .autoscaling
            .describe_scaling_activities()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .into_paginator()
            .items()
```

```
            .send()
            .collect::<Result<Vec<_>, _>>()
            .await
            .map_err(|e| {
                anyhow!(
                    "There was an error retrieving scaling activities: {}",
                    DisplayErrorContext(&e)
                )
            });

        AutoScalingScenarioDescription {
            group,
            instances,
            activities,
        }
    }
```

- API 자세한 내용은 Rust 참조 DescribeScalingActivities 의 섹션을 참조하세요. AWS SDK API

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeScalingProcessTypes**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeScalingProcessTypes의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용 가능한 프로세스 유형을 설명하려면

이 예제에서는 사용 가능한 프로세스 유형을 설명합니다.

```
aws autoscaling describe-scaling-process-types
```

출력:

```
{
```

```
    "Processes": [
        {
            "ProcessName": "AZRebalance"
        },
        {
            "ProcessName": "AddToLoadBalancer"
        },
        {
            "ProcessName": "AlarmNotification"
        },
        {
            "ProcessName": "HealthCheck"
        },
        {
            "ProcessName": "InstanceRefresh"
        },
        {
            "ProcessName": "Launch"
        },
        {
            "ProcessName": "ReplaceUnhealthy"
        },
        {
            "ProcessName": "ScheduledActions"
        },
        {
            "ProcessName": "Terminate"
        }
    ]
}
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 조정 프로세스 일시 중지 및 재개를 참조하세요.

- 자세한 API 내용은 명령 참조DescribeScalingProcessTypes의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 프로세스 유형을 나열합니다.

```
Get-ASScalingProcessType
```

출력:

```
ProcessName
-----------
AZRebalance
AddToLoadBalancer
AlarmNotification
HealthCheck
Launch
ReplaceUnhealthy
ScheduledActions
Terminate
```

- API 자세한 내용은 Cmdlet 참조DescribeScalingProcessTypes의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeScheduledActions**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeScheduledActions의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 예약된 모든 작업을 설명하려면

이 예제에서는 예약된 모든 작업에 대해 설명합니다.

```
aws autoscaling describe-scheduled-actions
```

출력:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
```

```
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 예약된 조정을 참조하세요.

예제 2: 지정된 그룹에 대해 예약된 작업을 설명하려면

특정 Auto Scaling 그룹에 대해 예약된 작업을 설명하려면 `--auto-scaling-group-name` 옵션을 사용합니다.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
```

```
        }
    ]
}
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 예약된 조정을 참조하세요.

예제 3: 지정된 예약된 작업을 설명하려면

특정 예약된 작업을 설명하려면 `--scheduled-action-names` 옵션을 사용합니다.

```
aws autoscaling describe-scheduled-actions \
    --scheduled-action-names my-recurring-action
```

출력:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 예약된 조정을 참조하세요.

예제 4: 지정된 시작 시간으로 예약된 작업을 설명하려면

특정 시간에 시작하는 예약된 작업을 설명하려면 `--start-time` 옵션을 사용합니다.

```
aws autoscaling describe-scheduled-actions \
```

```
    --start-time "2023-12-01T04:00:00Z"
```

출력:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 예약된 조정을 참조하세요. EC2 Auto Scaling

예제 5: 지정된 시간에 종료되는 예약된 작업을 설명하려면

특정 시간에 종료되는 예약된 작업을 설명하려면 `--end-time` 옵션을 사용합니다.

```
aws autoscaling describe-scheduled-actions \
    --end-time "2023-12-01T04:00:00Z"
```

출력:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
```

```
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 예약된 조정을 참조하세요. EC2 Auto Scaling

예제 6: 지정된 수의 예약된 작업을 설명하려면

특정 수의 예약된 작업을 반환하려면 `--max-items` 옵션을 사용합니다.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --max-items 1
```

출력:

```
{
    "ScheduledUpdateGroupActions": [
        {
            "AutoScalingGroupName": "my-asg",
            "ScheduledActionName": "my-recurring-action",
            "Recurrence": "30 0 1 1,6,12 *",
            "ScheduledActionARN": "arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8e86b655-b2e6-4410-8f29-
b4f094d6871c:autoScalingGroupName/my-asg:scheduledActionName/my-recurring-
action",
            "StartTime": "2023-12-01T04:00:00Z",
            "Time": "2023-12-01T04:00:00Z",
            "MinSize": 1,
            "MaxSize": 6,
            "DesiredCapacity": 4,
            "TimeZone": "America/New_York"
```

```
        }
    ]
}
```

출력에 NextToken 필드가 포함된 경우 더 많은 예약된 작업이 있습니다. 추가 예약된 작업을 가져오려면 다음과 같이 후속 통화에서 --starting-token 옵션과 함께 이 필드의 값을 사용합니다.

```
aws autoscaling describe-scheduled-actions \
    --auto-scaling-group-name my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 예약된 조정을 참조하세요.

- 자세한 API 내용은 명령 참조DescribeScheduledActions의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 예약된 조정 작업을 설명합니다.

```
Get-ASScheduledAction -AutoScalingGroupName my-asg
```

출력:

```
AutoScalingGroupName : my-asg
DesiredCapacity      : 10
EndTime              :
MaxSize              :
MinSize              :
Recurrence           :
ScheduledActionARN   : arn:aws:autoscaling:us-
west-2:123456789012:scheduledUpdateGroupAction:8a4c5f24-6ec6-4306-a2dd-f7
                       2c3af3a4d6:autoScalingGroupName/my-
asg:scheduledActionName/myScheduledAction
ScheduledActionName  : myScheduledAction
StartTime            : 11/30/2015 8:00:00 AM
Time                 : 11/30/2015 8:00:00 AM
```

예제 2: 이 예제에서는 지정된 예약된 조정 작업을 설명합니다.

```
Get-ASScheduledAction -ScheduledActionName @("myScheduledScaleOut",
 "myScheduledScaleIn")
```

예제 3: 이 예제에서는 지정된 시간으로 시작하는 예약된 조정 작업에 대해 설명합니다.

```
Get-ASScheduledAction -StartTime "2015-12-01T08:00:00Z"
```

예제 4: 이 예제에서는 지정된 시간만큼 종료되는 예약된 조정 작업을 설명합니다.

```
Get-ASScheduledAction -EndTime "2015-12-30T08:00:00Z"
```

예제 5: 이 예제에서는 모든 Auto Scaling 그룹에 대해 예약된 조정 작업을 설명합니다.

```
Get-ASScheduledAction
```

- API 자세한 내용은 Cmdlet 참조DescribeScheduledActions의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeTags**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeTags의 사용 방법을 보여 줍니다.

CLI

AWS CLI

모든 태그를 설명하려면

이 예제에서는 모든 태그를 설명합니다.

```
aws autoscaling describe-tags
```

출력:

```
{
    "Tags": [
        {
```

```
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "Research",
            "Key": "Dept"
        },
        {
            "ResourceType": "auto-scaling-group",
            "ResourceId": "my-asg",
            "PropagateAtLaunch": true,
            "Value": "WebServer",
            "Key": "Role"
        }
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스 태그를 참조하세요. EC2 Auto Scaling

예제 2: 지정된 그룹에 대한 태그를 설명하려면

특정 Auto Scaling 그룹에 대한 태그를 설명하려면 `--filters` 옵션을 사용합니다.

```
aws autoscaling describe-tags --filters Name=auto-scaling-group,Values=my-asg
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스 태그 지정을 참조하세요. EC2 Auto Scaling

예제 3: 지정된 태그 수를 설명하려면

특정 수의 태그를 반환하려면 `--max-items` 옵션을 사용합니다.

```
aws autoscaling describe-tags \
    --max-items 1
```

출력에 `NextToken` 필드가 포함된 경우 태그가 더 많습니다. 추가 태그를 가져오려면 다음과 같이 후속 호출에서 `--starting-token` 옵션과 함께 이 필드의 값을 사용합니다.

```
aws autoscaling describe-tags \
    --filters Name=auto-scaling-group,Values=my-asg \
    --starting-token Z3M3LMPEXAMPLE
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스 태그를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeTags의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 키 값이 'myTag' 또는 'myTag2'인 태그를 설명합니다. 필터 이름의 가능한 값은 'auto-scaling-group', '키', '값' 및 'propagate-at-launch'입니다. 이 예제에서 사용하는 구문에는 PowerShell 버전 3 이상이 필요합니다.

```
Get-ASTag -Filter @( @{ Name="key"; Values=@("myTag", "myTag2") } )
```

출력:

```
Key               : myTag2
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue2

Key               : myTag
PropagateAtLaunch : True
ResourceId        : my-asg
ResourceType      : auto-scaling-group
Value             : myTagValue
```

예제 2: PowerShell 버전 2에서는 New-Object를 사용하여 필터 파라미터에 대한 필터를 생성해야 합니다.

```
$keys = New-Object string[] 2
$keys[0] = "myTag"
$keys[1] = "myTag2"
$filter = New-Object Amazon.AutoScaling.Model.Filter
$filter.Name = "key"
$filter.Values = $keys
Get-ASTag -Filter @( $filter )
```

예제 3: 이 예제에서는 모든 Auto Scaling 그룹의 모든 태그를 설명합니다.

```
Get-ASTag
```

- API 자세한 내용은 Cmdlet 참조DescribeTags의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DescribeTerminationPolicyTypes**AWS SDK 함께 사용 CLI

다음 코드 예제는 DescribeTerminationPolicyTypes의 사용 방법을 보여 줍니다.

CLI

AWS CLI

사용 가능한 종료 정책 유형을 설명하려면

이 예제에서는 사용 가능한 종료 정책 유형을 설명합니다.

```
aws autoscaling describe-termination-policy-types
```

출력:

```
{
    "TerminationPolicyTypes": [
        "AllocationStrategy",
        "ClosestToNextInstanceHour",
        "Default",
        "NewestInstance",
        "OldestInstance",
        "OldestLaunchConfiguration",
        "OldestLaunchTemplate"
    ]
}
```

자세한 내용은 Amazon Auto Scaling 사용 설명서의 에서 스케일링 중에 종료되는 Auto Scaling 인스턴스 제어를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DescribeTerminationPolicyTypes의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 Auto Scaling에서 지원하는 종료 정책을 나열합니다.

```
Get-ASTerminationPolicyType
```

출력:

```
ClosestToNextInstanceHour
Default
NewestInstance
OldestInstance
OldestLaunchConfiguration
```

- API 자세한 내용은 Cmdlet 참조DescribeTerminationPolicyTypes의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DetachInstances**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DetachInstances`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹에서 인스턴스를 분리하려면

이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 분리합니다.

```
aws autoscaling detach-instances \
    --instance-ids i-030017cfa84b20135 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

출력:

```
{
    "Activities": [
        {
            "ActivityId": "5091cb52-547a-47ce-a236-c9ccbc2cb2c9",
            "AutoScalingGroupName": "my-asg",
            "Description": "Detaching EC2 instance: i-030017cfa84b20135",
            "Cause": "At 2020-10-31T17:35:04Z instance i-030017cfa84b20135 was
 detached in response to a user request, shrinking the capacity from 2 to 1.",
            "StartTime": "2020-04-12T15:02:16.179Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- 자세한 API 내용은 명령 참조DetachInstances의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 분리하고 Auto Scaling이 대체 인스턴스를 시작하지 않도록 원하는 용량을 줄입니다.

```
Dismount-ASInstance -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

출력:

```
ActivityId           : 06733445-ce94-4039-be1b-b9f1866e276e
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-93633f9b was detached
 in response to a user request, shrinking
                       the capacity from 2 to 1.
Description          : Detaching EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
```

```
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

예제 2: 이 예제는 원하는 용량을 줄이지 않고 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 분리합니다. Auto Scaling은 대체 인스턴스를 시작합니다.

```
Dismount-ASInstance -InstanceId i-7bf746a2 -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

출력:

```
ActivityId           : f43a3cd4-d38c-4af7-9fe0-d76ec2307b6d
AutoScalingGroupName : my-asg
Cause                : At 2015-11-20T22:34:59Z instance i-7bf746a2 was detached
 in response to a user request.
Description          : Detaching EC2 instance: i-7bf746a2
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/20/2015 2:34:59 PM
StatusCode           : InProgress
StatusMessage        :
```

- API 자세한 내용은 Cmdlet 참조DetachInstances의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DetachLoadBalancers**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DetachLoadBalancers`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹에서 Classic Load Balancer를 분리하려면

이 예제에서는 지정된 Auto Scaling 그룹에서 지정된 Classic Load Balancer를 분리합니다.

```
aws autoscaling detach-load-balancers \
    --load-balancer-names my-load-balancer \
    --auto-scaling-group-name my-asg
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹에 로드 밸런서 연결을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DetachLoadBalancers의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹에서 지정된 로드 밸런서를 분리합니다.

```
Dismount-ASLoadBalancer -LoadBalancerName my-lb -AutoScalingGroupName my-asg
```

- API 자세한 내용은 Cmdlet 참조DetachLoadBalancers의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **DisableMetricsCollection**AWS SDK 함께 사용 CLI

다음 코드 예제는 `DisableMetricsCollection`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Disable the collection of metric data for an Amazon EC2 Auto Scaling
    /// group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <returns>A Boolean value that indicates the success or failure of
    /// the operation.</returns>
    public async Task<bool> DisableMetricsCollectionAsync(string groupName)
    {
        var request = new DisableMetricsCollectionRequest
        {
            AutoScalingGroupName = groupName,
        };

        var response = await
_amazonAutoScaling.DisableMetricsCollectionAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 자세한 API 내용은 참조DisableMetricsCollection의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
            autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

    }
```

- 자세한 API 내용은 참조DisableMetricsCollection의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

Auto Scaling 그룹 지표 수집을 비활성화는 방법

이 예시에서는 지정된 Auto Scaling 그룹에 대한 `GroupDesiredCapacity` 지표 수집을 비활성화합니다.

```
aws autoscaling disable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스에 대한 CloudWatch 지표 모니터링을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조DisableMetricsCollection의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void disableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
           System.out.println("The disable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
```

```
        }
    }
```

- 자세한 API 내용은 참조DisableMetricsCollection의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun disableMetricsCollection(groupName: String) {
    val disableMetricsCollectionRequest =
        DisableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->

 autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest)
        println("The disable metrics collection operation was successful")
    }
}
```

- API 자세한 내용은 DisableMetricsCollection의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function disableMetricsCollection($autoScalingGroupName)
    {
        return $this->autoScalingClient->disableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
        ]);
    }
```

- 자세한 API 내용은 참조DisableMetricsCollection의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 지표의 모니터링을 비활성화합니다.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg -Metric
 @("GroupMinSize", "GroupMaxSize")
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 지표의 모니터링을 비활성화합니다.

```
Disable-ASMetricsCollection -AutoScalingGroupName my-asg
```

- 자세한 API 내용은 Cmdlet 참조DisableMetricsCollection의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def disable_metrics(self, group_name: str) -> Dict[str, Any]:
        """
        Stops CloudWatch metric collection for the Auto Scaling group.

        :param group_name: The name of the group.
        :return: A dictionary with the response from disabling the metrics
 collection.
        :raises ClientError: If there is an error disabling metrics collection.
        """
        try:
            response = self.autoscaling_client.disable_metrics_collection(
                AutoScalingGroupName=group_name
            )
            logger.info(
                f"Successfully disabled metrics collection for group
 '{group_name}'."
            )
            return response
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
```

```
                f"Couldn't disable metrics for group '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            raise
```

- API 자세한 내용은 DisableMetricsCollection의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        // If this fails it's fine, just means there are extra cloudwatch metrics
events for the scale-down.
        let _ = self
            .autoscaling
            .disable_metrics_collection()
            .auto_scaling_group_name(self.auto_scaling_group_name.clone())
            .send()
            .await;
```

- API 자세한 내용은 Rust 참조 DisableMetricsCollection 의 섹션을 참조하세요. AWS SDK API

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **EnableMetricsCollection**AWS SDK 함께 사용 CLI

다음 코드 예제는 `EnableMetricsCollection`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/// <summary>
/// Enable the collection of metric data for an Auto Scaling group.
/// </summary>
/// <param name="groupName">The name of the Auto Scaling group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> EnableMetricsCollectionAsync(string groupName)
{
    var listMetrics = new List<string>
        {
            "GroupMaxSize",
        };

    var collectionRequest = new EnableMetricsCollectionRequest
    {
        AutoScalingGroupName = groupName,
        Metrics = listMetrics,
        Granularity = "1Minute",
    };
```

```
        var response = await
_amazonAutoScaling.EnableMetricsCollectionAsync(collectionRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 자세한 API 내용은 참조EnableMetricsCollection의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
                autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                      << std::endl;
```

```
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 자세한 API 내용은 참조EnableMetricsCollection의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

예 1: Auto Scaling 그룹 지표 수집을 활성화하는 방법

이 예시에서는 지정된 Auto Scaling 그룹에 대한 데이터 수집을 활성화합니다.

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --granularity "1Minute"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스에 대한 CloudWatch 지표 모니터링을 참조하세요. EC2 Auto Scaling

예 2: Auto Scaling 그룹의 지정된 지표에 대한 데이터를 수집하는 방법

특정 지표에 대한 데이터를 수집하려면 `--metrics` 옵션을 사용하세요.

```
aws autoscaling enable-metrics-collection \
    --auto-scaling-group-name my-asg \
    --metrics GroupDesiredCapacity --granularity "1Minute"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹 및 인스턴스에 대한 CloudWatch 지표 모니터링을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조EnableMetricsCollection의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void enableMetricsCollection(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                   .autoScalingGroupName(groupName)
                   .metrics("GroupMaxSize")
                   .granularity("1Minute")
                   .build();

           autoScalingClient.enableMetricsCollection(collectionRequest);
           System.out.println("The enable metrics collection operation was
successful");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 자세한 API 내용은 참조EnableMetricsCollection의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun enableMetricsCollection(groupName: String?) {
    val collectionRequest =
        EnableMetricsCollectionRequest {
            autoScalingGroupName = groupName
            metrics = listOf("GroupMaxSize")
            granularity = "1Minute"
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.enableMetricsCollection(collectionRequest)
        println("The enable metrics collection operation was successful")
    }
}
```

- API 자세한 내용은 EnableMetricsCollection의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function enableMetricsCollection($autoScalingGroupName, $granularity)
    {
```

```
        return $this->autoScalingClient->enableMetricsCollection([
            'AutoScalingGroupName' => $autoScalingGroupName,
            'Granularity' => $granularity,
        ]);
    }
```

- 자세한 API 내용은 참조EnableMetricsCollection의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 지표를 모니터링할 수 있습니다.

```
Enable-ASMetricsCollection  -Metric @("GroupMinSize", "GroupMaxSize") -
AutoScalingGroupName my-asg -Granularity 1Minute
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 지표를 모니터링할 수 있습니다.

```
Enable-ASMetricsCollection -AutoScalingGroupName my-asg -Granularity 1Minute
```

- API 자세한 내용은 Cmdlet 참조EnableMetricsCollection의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""
```

```
    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def enable_metrics(self, group_name: str, metrics: List[str]) -> Dict[str,
Any]:
        """
        Enables CloudWatch metric collection for Amazon EC2 Auto Scaling
activities.

        :param group_name: The name of the group to enable.
        :param metrics: A list of metrics to collect.
        :return: A dictionary with the response from enabling the metrics
collection.
        :raises ClientError: If there is an error enabling metrics collection.
        """
        try:
            response = self.autoscaling_client.enable_metrics_collection(
                AutoScalingGroupName=group_name, Metrics=metrics,
Granularity="1Minute"
            )
            logger.info(
                f"Successfully enabled metrics for Auto Scaling group
'{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Couldn't enable metrics on '{group_name}'. Error code:
{error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "ResourceContentionFault":
                logger.error(
                    f"There is a conflict with another operation that is
modifying the Auto Scaling group '{group_name}'. "
                    "Please try again later."
                )
            elif error_code == "InvalidParameterCombination":
```

```
                logger.error(
                    f"The combination of parameters provided for enabling metrics
on '{group_name}' is not valid. "
                    "Please check the parameters and try again."
                )
            raise
        else:
            return response
```

- API 자세한 내용은 EnableMetricsCollection의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        let enable_metrics_collection = autoscaling
            .enable_metrics_collection()
            .auto_scaling_group_name(auto_scaling_group_name.as_str())
            .granularity("1Minute")
            .set_metrics(Some(vec![
                String::from("GroupMinSize"),
                String::from("GroupMaxSize"),
                String::from("GroupDesiredCapacity"),
                String::from("GroupInServiceInstances"),
                String::from("GroupTotalInstances"),
            ]))
            .send()
            .await;
```

- API 자세한 내용은 Rust 참조 EnableMetricsCollection 의 섹션을 참조하세요. AWS SDK API

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **EnterStandby**AWS SDK 함께 사용 CLI

다음 코드 예제는 EnterStandby의 사용 방법을 보여 줍니다.

CLI

AWS CLI

인스턴스를 대기 모드로 이동하려면

이 예제에서는 지정된 인스턴스를 대기 모드로 전환합니다. 이는 현재 사용 중인 인스턴스를 업데이트하거나 문제를 해결하는 데 유용합니다.

```
aws autoscaling enter-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --should-decrement-desired-capacity
```

출력:

```
{
    "Activities": [
        {
            "ActivityId": "ffa056b4-6ed3-41ba-ae7c-249dfae6eba1",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance to Standby: i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:31:00Z instance i-061c63c5eb45f0416 was
 moved to standby in response to a user request, shrinking the capacity from 1 to
 0.",
            "StartTime": "2020-10-31T20:31:00.949Z",
            "StatusCode": "InProgress",
            "Progress": 50,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 인스턴스 수명 주기를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조EnterStandby의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 인스턴스를 대기 모드로 전환하고 Auto Scaling이 대체 인스턴스를 시작하지 않도록 원하는 용량을 줄입니다.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $true
```

출력:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request,
                       shrinking the capacity from 2 to 1.
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

예제 2: 이 예제에서는 원하는 용량을 줄이지 않고 지정된 인스턴스를 대기 모드로 전환합니다. Auto Scaling은 대체 인스턴스를 시작합니다.

```
Enter-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg -
ShouldDecrementDesiredCapacity $false
```

출력:

```
ActivityId           : e36a5a54-ced6-4df8-bd19-708e2a59a649
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:48:06Z instance i-95b8484f was moved to
 standby in response to a user request.
```

```
Description          : Moving EC2 instance to Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 50
StartTime            : 11/22/2015 7:48:06 AM
StatusCode           : InProgress
StatusMessage        :
```

- 자세한 API 내용은 Cmdlet 참조EnterStandby의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **ExecutePolicy**AWS SDK 함께 사용 CLI

다음 코드 예제는 `ExecutePolicy`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

조정 정책을 실행하려면

이 예제에서는 지정된 Auto Scaling 그룹에 `my-step-scale-out-policy` 대해 라는 조정 정책을 실행합니다.

```
aws autoscaling execute-policy \
    --auto-scaling-group-name my-asg \
    --policy-name  my-step-scale-out-policy \
    --metric-value 95 \
    --breach-threshold 80
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 단계 및 간단한 조정 정책을 참조하세요.

- 자세한 API 내용은 명령 참조ExecutePolicy의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹에 대해 지정된 정책을 실행합니다.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy"
```

예제 2: 이 예제는 휴지 기간이 완료될 때까지 기다린 후 지정된 Auto Scaling 그룹에 대해 지정된 정책을 실행합니다.

```
Start-ASPolicy -AutoScalingGroupName my-asg -PolicyName "myScaleInPolicy" -
HonorCooldown $true
```

- API 자세한 내용은 Cmdlet 참조ExecutePolicy의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **ExitStandby**AWS SDK 함께 사용 CLI

다음 코드 예제는 `ExitStandby`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

대기 모드에서 인스턴스를 이동하려면

이 예제에서는 지정된 인스턴스를 대기 모드에서 벗어납니다.

```
aws autoscaling exit-standby \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg
```

출력:

```
{
    "Activities": [
        {
```

```
            "ActivityId": "142928e1-a2dc-453a-9b24-b85ad6735928",
            "AutoScalingGroupName": "my-asg",
            "Description": "Moving EC2 instance out of Standby:
 i-061c63c5eb45f0416",
            "Cause": "At 2020-10-31T20:32:50Z instance i-061c63c5eb45f0416 was
 moved out of standby in response to a user request, increasing the capacity from
 0 to 1.",
            "StartTime": "2020-10-31T20:32:50.222Z",
            "StatusCode": "PreInService",
            "Progress": 30,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

자세한 내용은 Amazon [Auto Scaling 사용 설명서의 Auto Scaling 그룹에서 인스턴스를 일시적으로 제거하는](#) 단원을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조ExitStandby의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 인스턴스를 대기 모드에서 벗어납니다.

```
Exit-ASStandby -InstanceId i-93633f9b -AutoScalingGroupName my-asg
```

출력:

```
ActivityId           : 1833d3e8-e32f-454e-b731-0670ad4c6934
AutoScalingGroupName : my-asg
Cause                : At 2015-11-22T15:51:21Z instance i-95b8484f was moved out
 of standby in response to a user
                       request, increasing the capacity from 1 to 2.
Description          : Moving EC2 instance out of Standby: i-95b8484f
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 30
StartTime            : 11/22/2015 7:51:21 AM
StatusCode           : PreInService
```

```
StatusMessage        :
```

- 자세한 API 내용은 Cmdlet 참조ExitStandby의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **PutLifecycleHook**AWS SDK 함께 사용 CLI

다음 코드 예제는 `PutLifecycleHook`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 수명 주기 후크 생성

이 예제에서는 4,800초의 제한 시간으로 새로 시작된 인스턴스에서 호출할 수명 주기 후크를 생성합니다. 이는 사용자 데이터 스크립트가 완료될 때까지 인스턴스를 대기 상태로 유지하거나를 사용하여 AWS Lambda 함수를 호출하는 데 유용합니다 EventBridge.

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --heartbeat-timeout 4800
```

이 명령은 출력을 생성하지 않습니다. 동일한 이름의 수명 주기 후크가 이미 있는 경우 새 수명 주기 후크로 덮어씁니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 수명 주기 후크를 참조하세요. EC2 Auto Scaling

예제 2: 인스턴스 상태 전환을 알리기 위해 Amazon SNS 이메일 메시지를 보내려면

이 예제에서는 인스턴스 시작 시 알림을 수신하는 데 사용할 Amazon SNS 주제 및 IAM 역할과 함께 수명 주기 후크를 생성합니다.

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
```

```
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --role-arn arn:aws:iam::123456789012:role/my-auto-scaling-role
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 수명 주기 후크를 참조하세요. EC2 Auto Scaling

예제 3: Amazon SQS 대기열에 메시지를 게시하는 방법

이 예제에서는 메타데이터가 포함된 메시지를 지정된 Amazon SQS 대기열에 게시하는 수명 주기 후크를 생성합니다.

```
aws autoscaling put-lifecycle-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-hook-name my-launch-hook \
    --lifecycle-transition autoscaling:EC2_INSTANCE_LAUNCHING \
    --notification-target-arn arn:aws:sqs:us-west-2:123456789012:my-sqs-queue \
    --role-arn arn:aws:iam::123456789012:role/my-notification-role \
    --notification-metadata "SQS message metadata"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 수명 주기 후크를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조PutLifecycleHook의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 지정된 수명 주기 후크를 추가합니다.

```
Write-ASLifecycleHook -AutoScalingGroupName my-asg -LifecycleHookName
 "myLifecycleHook" -LifecycleTransition "autoscaling:EC2_INSTANCE_LAUNCHING" -
NotificationTargetARN "arn:aws:sns:us-west-2:123456789012:my-sns-topic" -RoleARN
 "arn:aws:iam::123456789012:role/my-iam-role"
```

- API 자세한 내용은 Cmdlet 참조PutLifecycleHook의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **PutNotificationConfiguration**AWS SDK 함께 사용 CLI

다음 코드 예제는 `PutNotificationConfiguration`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

알림을 추가하려면

이 예제에서는 지정된 Auto Scaling 그룹에 지정된 알림을 추가합니다.

```
aws autoscaling put-notification-configuration \
    --auto-scaling-group-name my-asg \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-sns-topic \
    --notification-type autoscaling:TEST_NOTIFICATION
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹이 확장될 때 Amazon SNS 알림 받기를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조PutNotificationConfiguration의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹이 EC2 인스턴스를 시작할 때 지정된 SNS 주제에 알림을 보내도록 구성합니다.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -
NotificationType "autoscaling:EC2_INSTANCE_LAUNCH" -TopicARN "arn:aws:sns:us-
west-2:123456789012:my-topic"
```

예제 2: 이 예제는 지정된 Auto Scaling 그룹이 EC2 인스턴스를 시작하거나 종료할 때 지정된 SNS 주제에 알림을 보내도록 구성합니다.

```
Write-ASNotificationConfiguration -AutoScalingGroupName my-asg -NotificationType
 @("autoscaling:EC2_INSTANCE_LAUNCH", "autoscaling:EC2_INSTANCE_TERMINATE") -
TopicARN "arn:aws:sns:us-west-2:123456789012:my-topic"
```

- API 자세한 내용은 Cmdlet 참조PutNotificationConfiguration의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **PutScalingPolicy**AWS SDK 함께 사용 CLI

다음 코드 예제는 `PutScalingPolicy`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 그룹에 대상 추적 조정 정책을 추가하려면

다음 `put-scaling-policy` 예제에서는 지정된 Auto Scaling 그룹에 대상 추적 조정 정책을 적용합니다. 출력에는 사용자를 대신하여 생성된 두 CloudWatch 경보의 ARNs 및 이름이 포함됩니다. 이름이 같은 조정 정책이 이미 있는 경우 새 조정 정책으로 덮어씁니다.

```
aws autoscaling put-scaling-policy --auto-scaling-group-name my-asg \
  --policy-name alb1000-target-tracking-scaling-policy \
  --policy-type TargetTrackingScaling \
  --target-tracking-configuration file://config.json
```

`config.json`의 콘텐츠:

```
{
     "TargetValue": 1000.0,
     "PredefinedMetricSpecification": {
          "PredefinedMetricType": "ALBRequestCountPerTarget",
          "ResourceLabel": "app/my-alb/778d41231b141a0f/targetgroup/my-alb-
target-group/943f017f100becff"
     }
}
```

출력:

```
{
     "PolicyARN": "arn:aws:autoscaling:region:account-id:scalingPolicy:228f02c2-
c665-4bfd-aaac-8b04080bea3c:autoScalingGroupName/my-asg:policyName/alb1000-
target-tracking-scaling-policy",
     "Alarms": [
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmHigh-fc0e4183-23ac-497e-9992-691c9980c38e",
             "AlarmName": "TargetTracking-my-asg-AlarmHigh-
fc0e4183-23ac-497e-9992-691c9980c38e"
         },
         {
             "AlarmARN": "arn:aws:cloudwatch:region:account-
id:alarm:TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-bd9e-471a352ee1a2",
             "AlarmName": "TargetTracking-my-asg-AlarmLow-61a39305-ed0c-47af-
bd9e-471a352ee1a2"
         }
     ]
 }
```

자세한 예는 Amazon EC2 Auto Scaling 사용 설명서의 AWS 명령줄 인터페이스(AWS CLI)에 대한 조정 정책 예제를 참조하세요.

- 자세한 API 내용은 명령 참조PutScalingPolicy의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 지정된 정책을 추가합니다. 지정된 조정 유형에 따라 ScalingAdjustment 파라미터를 해석하는 방법이 결정됩니다. 'ChangeInCapacity'를 사용하면 양수 값이 지정된 인스턴스 수만큼 용량을 늘리고 음수 값은 지정된 인스턴스 수만큼 용량을 줄입니다.

```
Write-ASScalingPolicy -AutoScalingGroupName my-asg -AdjustmentType
 "ChangeInCapacity" -PolicyName "myScaleInPolicy" -ScalingAdjustment -1
```

출력:

```
arn:aws:autoscaling:us-west-2:123456789012:scalingPolicy:aa3836ab-5462-42c7-adab-
e1d769fc24ef:autoScalingGroupName/my-asg
```

```
:policyName/myScaleInPolicy
```

- API 자세한 내용은 Cmdlet 참조PutScalingPolicy의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **PutScheduledUpdateGroupAction**AWS SDK 함께 사용 CLI

다음 코드 예제는 `PutScheduledUpdateGroupAction`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: Auto Scaling 그룹에 예약된 작업을 추가하려면

이 예제에서는 지정된 Auto Scaling 그룹에 지정된 예약된 작업을 추가합니다.

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-scheduled-action \
    --start-time "2023-05-12T08:00:00Z" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

이 명령은 출력을 생성하지 않습니다. 동일한 이름의 예약된 작업이 이미 있는 경우 새 예약된 작업으로 덮어씁니다.

자세한 예는 Amazon Auto Scaling 사용 설명서의 예약된 조정을 참조하세요. EC2 Auto Scaling

예제 2: 반복 일정을 지정하려면

이 예제에서는 매년 1월, 6월, 12월 1일에 00:30시에 실행되도록 예약된 작업을 생성하여 반복 일정에 따라 확장합니다.

```
aws autoscaling put-scheduled-update-group-action \
    --auto-scaling-group-name my-asg \
    --scheduled-action-name my-recurring-action \
```

```
    --recurrence "30 0 1 1,6,12 *" \
    --min-size 2 \
    --max-size 6 \
    --desired-capacity 4
```

이 명령은 출력을 생성하지 않습니다. 동일한 이름의 예약된 작업이 이미 있는 경우 새 예약된 작업으로 덮어씁니다.

자세한 예는 Amazon EC2 Auto Scaling 사용 설명서의 예약된 조정을 참조하세요.

- 자세한 API 내용은 명령 참조PutScheduledUpdateGroupAction의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 일회성 예약 작업을 생성하거나 업데이트하여 지정된 시작 시간에 원하는 용량을 변경합니다.

```
Write-ASScheduledUpdateGroupAction -AutoScalingGroupName my-asg -
ScheduledActionName "myScheduledAction" -StartTime "2015-12-01T00:00:00Z" -
DesiredCapacity 10
```

- API 자세한 내용은 Cmdlet 참조PutScheduledUpdateGroupAction의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **RecordLifecycleActionHeartbeat**AWS SDK 함께 사용 CLI

다음 코드 예제는 `RecordLifecycleActionHeartbeat`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

수명 주기 작업 하트비트를 기록하려면

이 예제에서는 인스턴스를 보류 상태로 유지하기 위해 수명 주기 작업 하트비트를 기록합니다.

```
aws autoscaling record-lifecycle-action-heartbeat \
    --lifecycle-hook-name my-launch-hook \
    --auto-scaling-group-name my-asg \
    --lifecycle-action-token bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon Auto Scaling 수명 주기 후크를 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조RecordLifecycleActionHeartbeat의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 수명 주기 작업에 대한 하트비트를 기록합니다. 이렇게 하면 사용자 지정 작업을 완료할 때까지 인스턴스가 보류 상태로 유지됩니다.

```
Write-ASLifecycleActionHeartbeat -AutoScalingGroupName my-asg -LifecycleHookName
 myLifecycleHook -LifecycleActionToken bcd2f1b8-9a78-44d3-8a7a-4dd07d7cf635
```

- 자세한 API 내용은 Cmdlet 참조RecordLifecycleActionHeartbeat의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **ResumeProcesses**AWS SDK 함께 사용 CLI

다음 코드 예제는 `ResumeProcesses`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

일시 중지된 프로세스를 재개하려면

이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 일시 중지된 조정 프로세스를 재개합니다.

```
aws autoscaling resume-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 조정 프로세스 일시 중지 및 재개를 참조하세요.

- 자세한 API 내용은 명령 참조ResumeProcesses의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹에 대해 지정된 Auto Scaling 프로세스를 재개합니다.

```
Resume-ASProcess -AutoScalingGroupName my-asg -ScalingProcess "AlarmNotification"
```

예제 2: 이 예제는 지정된 Auto Scaling 그룹에 대해 일시 중지된 모든 Auto Scaling 프로세스를 재개합니다.

```
Resume-ASProcess -AutoScalingGroupName my-asg
```

- API 자세한 내용은 Cmdlet 참조ResumeProcesses의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 안내서 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **SetDesiredCapacity**AWS SDK 함께 사용 CLI

다음 코드 예제는 `SetDesiredCapacity`의 사용 방법을 보여 줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Set the desired capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="desiredCapacity">The desired capacity for the Auto
    /// Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> SetDesiredCapacityAsync(
        string groupName,
        int desiredCapacity)
    {
        var capacityRequest = new SetDesiredCapacityRequest
        {
            AutoScalingGroupName = groupName,
            DesiredCapacity = desiredCapacity,
        };

        var response = await
_amazonAutoScaling.SetDesiredCapacityAsync(capacityRequest);
        Console.WriteLine($"You have set the DesiredCapacity to
{desiredCapacity}.");

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 자세한 API 내용은 참조SetDesiredCapacity의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetDesiredCapacity(2);

        Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
                autoScalingClient.SetDesiredCapacity(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 자세한 API 내용은 참조SetDesiredCapacity의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

Auto Scaling 그룹에 원하는 용량을 설정하는 방법

이 예시에서는 지정된 Auto Scaling 그룹에 원하는 용량을 설정합니다.

```
aws autoscaling set-desired-capacity \
```

```
    --auto-scaling-group-name my-asg \
    --desired-capacity 2 \
    --honor-cooldown
```

이 명령은 성공하면 프롬프트로 돌아갑니다.

- 자세한 API 내용은 명령 참조SetDesiredCapacity의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                   .autoScalingGroupName(groupName)
                   .desiredCapacity(2)
                   .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 자세한 API 내용은 참조SetDesiredCapacity의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun setDesiredCapacity(groupName: String) {
    val capacityRequest =
        SetDesiredCapacityRequest {
            autoScalingGroupName = groupName
            desiredCapacity = 2
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.setDesiredCapacity(capacityRequest)
        println("You set the DesiredCapacity to 2")
    }
}
```

- API 자세한 내용은 SetDesiredCapacity의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function setDesiredCapacity($autoScalingGroupName, $desiredCapacity)
    {
        return $this->autoScalingClient->setDesiredCapacity([
            'AutoScalingGroupName' => $autoScalingGroupName,
```

```
            'DesiredCapacity' => $desiredCapacity,
        ]);
    }
```

- 자세한 API 내용은 참조SetDesiredCapacity의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹의 크기를 설정합니다.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2
```

예제 2: 이 예제는 지정된 Auto Scaling 그룹의 크기를 설정하고 새 크기로 크기 조정하기 전에 휴지 기간이 완료될 때까지 기다립니다.

```
Set-ASDesiredCapacity -AutoScalingGroupName my-asg -DesiredCapacity 2 -
HonorCooldown $true
```

- API 자세한 내용은 Cmdlet 참조SetDesiredCapacity의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
```

```
        """
        self.autoscaling_client = autoscaling_client


    def set_desired_capacity(self, group_name: str, capacity: int) -> None:
        """
        Sets the desired capacity of the group. Amazon EC2 Auto Scaling tries to
keep the
        number of running instances equal to the desired capacity.

        :param group_name: The name of the group to update.
        :param capacity: The desired number of running instances.
        :return: None
        :raises ClientError: If there is an error setting the desired capacity.
        """
        try:
            self.autoscaling_client.set_desired_capacity(
                AutoScalingGroupName=group_name,
                DesiredCapacity=capacity,
                HonorCooldown=False,
            )
            logger.info(
                f"Successfully set desired capacity of {capacity} for Auto
Scaling group '{group_name}'."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(
                f"Failed to set desired capacity for Auto Scaling group
'{group_name}'."
            )
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
Scaling group '{group_name}'. "
                    "Please wait for the activity to complete before attempting
to set the desired capacity."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 자세한 내용은 SetDesiredCapacity의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   pub async fn scale_desired_capacity(&self, capacity: i32) -> Result<(),
ScenarioError> {
       // 7. SetDesiredCapacity: set desired capacity to 2.
       //   Wait for a second instance to launch.
       let update_group = self
           .autoscaling
           .set_desired_capacity()
           .auto_scaling_group_name(self.auto_scaling_group_name.clone())
           .desired_capacity(capacity)
           .send()
           .await;
       if let Err(err) = update_group {
           return Err(ScenarioError::new(
               format!("Failed to update group to desired capacity
({capacity}))").as_str(),
               &err,
           ));
       }
       Ok(())
   }
```

- API 자세한 내용은 Rust 참조 SetDesiredCapacity 의 섹션을 참조하세요. AWS SDK API

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **SetInstanceHealth**AWS SDK 함께 사용 CLI

다음 코드 예제는 SetInstanceHealth의 사용 방법을 보여 줍니다.

CLI

AWS CLI

인스턴스의 상태를 설정하려면

이 예제에서는 지정된 인스턴스의 상태를 로 설정합니다Unhealthy.

```
aws autoscaling set-instance-health \
    --instance-id i-061c63c5eb45f0416 \
    --health-status Unhealthy
```

이 명령은 출력을 생성하지 않습니다.

- 자세한 API 내용은 명령 참조SetInstanceHealth의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 인스턴스의 상태를 'Unhealthy'로 설정하여 서비스를 중단합니다. Auto Scaling은 인스턴스를 종료하고 교체합니다.

```
Set-ASInstanceHealth -HealthStatus Unhealthy -InstanceId i-93633f9b
```

예제 2: 이 예제는 지정된 인스턴스의 상태를 '정상'으로 설정하여 서비스를 유지합니다. Auto Scaling 그룹의 상태 확인 유예 기간은 적용되지 않습니다.

```
Set-ASInstanceHealth -HealthStatus Healthy -InstanceId i-93633f9b -
ShouldRespectGracePeriod $false
```

- API 자세한 내용은 Cmdlet 참조SetInstanceHealth의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **SetInstanceProtection**AWS SDK 함께 사용 CLI

다음 코드 예제는 SetInstanceProtection의 사용 방법을 보여 줍니다.

CLI

AWS CLI

예제 1: 인스턴스에 대한 인스턴스 보호 설정을 활성화하려면

이 예제에서는 지정된 인스턴스에 대한 인스턴스 보호를 활성화합니다.

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg --protected-from-scale-in
```

이 명령은 출력을 생성하지 않습니다.

예제 2: 인스턴스에 대한 인스턴스 보호 설정을 비활성화하려면

이 예제에서는 지정된 인스턴스에 대한 인스턴스 보호를 비활성화합니다.

```
aws autoscaling set-instance-protection \
    --instance-ids i-061c63c5eb45f0416 \
    --auto-scaling-group-name my-asg \
    --no-protected-from-scale-in
```

이 명령은 출력을 생성하지 않습니다.

- 자세한 API 내용은 명령 참조SetInstanceProtection의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 인스턴스에 대한 인스턴스 보호를 활성화합니다.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $true
```

예제 2: 이 예제는 지정된 인스턴스에 대한 인스턴스 보호를 비활성화합니다.

```
Set-ASInstanceProtection -AutoScalingGroupName my-asg -InstanceId i-12345678 -
ProtectedFromScaleIn $false
```

- API 자세한 내용은 Cmdlet 참조SetInstanceProtection의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **SuspendProcesses**AWS SDK 함께 사용 CLI

다음 코드 예제는 SuspendProcesses의 사용 방법을 보여 줍니다.

CLI

AWS CLI

Auto Scaling 프로세스를 일시 중지하려면

이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 조정 프로세스를 일시 중지합니다.

```
aws autoscaling suspend-processes \
    --auto-scaling-group-name my-asg \
    --scaling-processes AlarmNotification
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 조정 프로세스 일시 중지 및 재개를 참조하세요.

- 자세한 API 내용은 명령 참조SuspendProcesses의 섹션을 참조하세요. AWS CLI

PowerShell

용 도구 PowerShell

예제 1: 이 예제에서는 지정된 Auto Scaling 그룹에 대해 지정된 Auto Scaling 프로세스를 일시 중지합니다.

```
Suspend-ASProcess -AutoScalingGroupName my-asg -ScalingProcess
 "AlarmNotification"
```

예제 2: 이 예제에서는 지정된 Auto Scaling 그룹에 대한 모든 Auto Scaling 프로세스를 일시 중지합니다.

```
Suspend-ASProcess -AutoScalingGroupName my-asg
```

- API 자세한 내용은 Cmdlet 참조SuspendProcesses의 섹션을 참조하세요. AWS Tools for PowerShell

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **TerminateInstanceInAutoScalingGroup**AWS SDK 함께 사용 CLI

다음 코드 예제는 `TerminateInstanceInAutoScalingGroup`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기
- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Terminate all instances in the Auto Scaling group in preparation for
    /// deleting the group.
    /// </summary>
    /// <param name="instanceId">The instance Id of the instance to terminate.</
param>
    /// <returns>A Boolean value that indicates the success or failure of
```

```
    /// the operation.</returns>
    public async Task<bool> TerminateInstanceInAutoScalingGroupAsync(
        string instanceId)
    {
        var request = new TerminateInstanceInAutoScalingGroupRequest
        {
            InstanceId = instanceId,
            ShouldDecrementDesiredCapacity = false,
        };

        var response = await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You have terminated the instance: {instanceId}");
            return true;
        }

        Console.WriteLine($"Could not terminate {instanceId}");
        return false;
    }
```

- 자세한 API 내용은 참조TerminateInstanceInAutoScalingGroup의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest
request;
        request.SetInstanceId(instanceIDs[instanceNumber - 1]);
        request.SetShouldDecrementDesiredCapacity(false);

        Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome
outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Waiting for EC2 instance with ID '"
                      << instanceIDs[instanceNumber - 1] << "' to terminate..."
                      << std::endl;
        }
        else {
            std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 자세한 API 내용은 참조TerminateInstanceInAutoScalingGroup의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

Auto Scaling 그룹에서 인스턴스를 종료하는 방법

이 예시에서는 그룹 크기를 업데이트하지 않고 지정된 Auto Scaling 그룹에서 지정된 인스턴스를 종료합니다. Amazon EC2 Auto Scaling은 지정된 인스턴스가 종료된 후 대체 인스턴스를 시작합니다.

```
aws autoscaling terminate-instance-in-auto-scaling-group \
    --instance-id i-061c63c5eb45f0416 \
    --no-should-decrement-desired-capacity
```

출력:

```
{
    "Activities": [
        {
            "ActivityId": "8c35d601-793c-400c-fcd0-f64a27530df7",
            "AutoScalingGroupName": "my-asg",
            "Description": "Terminating EC2 instance: i-061c63c5eb45f0416",
            "Cause": "",
            "StartTime": "2020-10-31T20:34:25.680Z",
            "StatusCode": "InProgress",
            "Progress": 0,
            "Details": "{\"Subnet ID\":\"subnet-6194ea3b\",\"Availability Zone\":
\"us-west-2c\"}"
        }
    ]
}
```

- 자세한 API 내용은 명령 참조TerminateInstanceInAutoScalingGroup의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
 autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
 TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
```

```
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 자세한 API 내용은 참조TerminateInstanceInAutoScalingGroup의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun terminateInstanceInAutoScalingGroup(instanceIdVal: String) {
    val request =
        TerminateInstanceInAutoScalingGroupRequest {
            instanceId = instanceIdVal
            shouldDecrementDesiredCapacity = false
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.terminateInstanceInAutoScalingGroup(request)
        println("You have terminated instance $instanceIdVal")
    }
}
```

- API 자세한 내용은 TerminateInstanceInAutoScalingGroup의 에서 AWS SDK Kotlin API 참조를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function terminateInstanceInAutoScalingGroup(
        $instanceId,
        $shouldDecrementDesiredCapacity = true,
        $attempts = 0
    ) {
        try {
            return $this->autoScalingClient-
>terminateInstanceInAutoScalingGroup([
                'InstanceId' => $instanceId,
                'ShouldDecrementDesiredCapacity' =>
 $shouldDecrementDesiredCapacity,
            ]);
        } catch (AutoScalingException $exception) {
            if ($exception->getAwsErrorCode() == "ScalingActivityInProgress" &&
 $attempts < 5) {
                error_log("Cannot terminate an instance while it is still
 pending. Waiting then trying again.");
                sleep(5 * (1 + $attempts));
                return $this->terminateInstanceInAutoScalingGroup(
                    $instanceId,
                    $shouldDecrementDesiredCapacity,
                    ++$attempts
                );
            } else {
                throw $exception;
            }
        }
    }
```

- 자세한 API 내용은 참조TerminateInstanceInAutoScalingGroup의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 인스턴스를 종료하고 Auto Scaling이 대체 인스턴스를 시작하지 않도록 Auto Scaling 그룹의 원하는 용량을 줄입니다.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $true
```

출력:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request, shrinking the capacity from 2 to 1.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

예제 2: 이 예제는 Auto Scaling 그룹의 원하는 용량을 줄이지 않고 지정된 인스턴스를 종료합니다. Auto Scaling은 대체 인스턴스를 시작합니다.

```
Stop-ASInstanceInAutoScalingGroup -InstanceId i-93633f9b -
ShouldDecrementDesiredCapacity $false
```

출력:

```
ActivityId           : 2e40d9bd-1902-444c-abf3-6ea0002efdc5
AutoScalingGroupName :
Cause                : At 2015-11-22T16:09:03Z instance i-93633f9b was taken out
 of service in response to a user
                       request.
Description          : Terminating EC2 instance: i-93633f9b
Details              : {"Availability Zone":"us-west-2b","Subnet
 ID":"subnet-5264e837"}
```

```
EndTime              :
Progress             : 0
StartTime            : 11/22/2015 8:09:03 AM
StatusCode           : InProgress
StatusMessage        :
```

- API 자세한 내용은 Cmdlet 참조TerminateInstanceInAutoScalingGroup의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def terminate_instance(
        self, instance_id: str, decrease_capacity: bool
    ) -> Dict[str, Any]:
        """
        Stops an instance.

        :param instance_id: The ID of the instance to stop.
        :param decrease_capacity: Specifies whether to decrease the desired
 capacity
                                  of the group. When passing True for this
 parameter,
```

```
                                you can stop an instance without having a
replacement
                                instance start when the desired capacity
threshold is
                                crossed.
        :return: A dictionary containing details of the scaling activity that
occurs
                 in response to this action.
        :raises ClientError: If there is an error terminating the instance.
        """
        try:
            response =
self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
ShouldDecrementDesiredCapacity=decrease_capacity
            )
            logger.info(f"Successfully terminated instance {instance_id}.")
            return response["Activity"]

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to terminate instance {instance_id}.")
            if error_code == "ScalingActivityInProgress":
                logger.error(
                    "A scaling activity is currently in progress for the Auto
Scaling group "
                    f"associated with instance '{instance_id}'. "
                    "Please wait for the activity to complete before attempting
to terminate the instance."
                )
            elif error_code == "ResourceInUse":
                logger.error(
                    f"The instance '{instance_id}' or an associated resource is
currently in use "
                    "and cannot be terminated. "
                    "Ensure the instance is not involved in any ongoing processes
and try again."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 자세한 내용은 TerminateInstanceInAutoScalingGroup의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
pub async fn terminate_some_instance(&self) -> Result<(), ScenarioError> {
    // Retrieve a list of instances in the auto scaling group.
    let auto_scaling_group = self.get_group().await?;
    let instances = auto_scaling_group.instances();
    // Or use other logic to find an instance to terminate.
    let instance = instances.first();
    if let Some(instance) = instance {
        let instance_id = if let Some(instance_id) = instance.instance_id() {
            instance_id
        } else {
            return Err(ScenarioError::with("Missing instance id"));
        };
        let termination = self
            .ec2
            .terminate_instances()
            .instance_ids(instance_id)
            .send()
            .await;
        if let Err(err) = termination {
            Err(ScenarioError::new(
                "There was a problem terminating an instance",
                &err,
            ))
        } else {
            Ok(())
        }
    } else {
        Err(ScenarioError::with("There was no instance to terminate"))
```

```
        }
    }

    async fn get_group(&self) -> Result<AutoScalingGroup, ScenarioError> {
        let describe_auto_scaling_groups = self
            .autoscaling
            .describe_auto_scaling_groups()
            .auto_scaling_group_names(self.auto_scaling_group_name.clone())
            .send()
            .await;

        if let Err(err) = describe_auto_scaling_groups {
            return Err(ScenarioError::new(
                format!(
                    "Failed to get status of autoscaling group {}",
                    self.auto_scaling_group_name.clone()
                )
                .as_str(),
                &err,
            ));
        }

        let describe_auto_scaling_groups_output =
describe_auto_scaling_groups.unwrap();
        let auto_scaling_groups =
describe_auto_scaling_groups_output.auto_scaling_groups();
        let auto_scaling_group = auto_scaling_groups.first();

        if auto_scaling_group.is_none() {
            return Err(ScenarioError::with(format!(
                "Could not find autoscaling group {}",
                self.auto_scaling_group_name.clone()
            )));
        }

        Ok(auto_scaling_group.unwrap().clone())
    }
```

- API 자세한 내용은 Rust 참조 TerminateInstanceInAutoScalingGroup 의 섹션을 참조하세요. AWS SDK API

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

또는 와 **UpdateAutoScalingGroup**AWS SDK 함께 사용 CLI

다음 코드 예제는 `UpdateAutoScalingGroup`의 사용 방법을 보여 줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 기본 사항 알아보기
- 복원력이 뛰어난 서비스 구축 및 관리

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /// <summary>
    /// Update the capacity of an Auto Scaling group.
    /// </summary>
    /// <param name="groupName">The name of the Auto Scaling group.</param>
    /// <param name="launchTemplateName">The name of the EC2 launch template.</
param>
    /// <param name="maxSize">The maximum number of instances that can be
    /// created for the Auto Scaling group.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> UpdateAutoScalingGroupAsync(
        string groupName,
        string launchTemplateName,
        int maxSize)
    {
        var templateSpecification = new LaunchTemplateSpecification
        {
            LaunchTemplateName = launchTemplateName,
```

```
        };

        var groupRequest = new UpdateAutoScalingGroupRequest
        {
            MaxSize = maxSize,
            AutoScalingGroupName = groupName,
            LaunchTemplate = templateSpecification,
        };

        var response = await
_amazonAutoScaling.UpdateAutoScalingGroupAsync(groupRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"You successfully updated the Auto Scaling group
{groupName}.");
            return true;
        }
        else
        {
            return false;
        }
    }
```

- 자세한 API 내용은 참조UpdateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for .NET API

C++

SDK C++용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

        }
```

- 자세한 API 내용은 참조UpdateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for C++ API

CLI

AWS CLI

예 1: Auto Scaling 그룹의 크기 한도를 업데이트하는 방법

이 예시에서는 지정된 Auto Scaling 그룹을 최소 크기가 2, 최대 크기가 10으로 업데이트합니다.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --min-size 2 \
    --max-size 10
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹의 용량 제한 설정을 참조하세요. EC2 Auto Scaling

예 2: Elastic Load Balancing 상태 확인을 추가하고 사용할 가용 영역 및 서브넷을 지정하는 방법

이 예시에서는 Elastic Load Balancing 상태 확인을 추가하도록 지정된 Auto Scaling 그룹을 업데이트합니다. 또한 이 명령은 의 값을 IDs 여러 가용 영역의 서브넷 목록`--vpc-zone-identifier`으로 업데이트합니다.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --health-check-type ELB \
    --health-check-grace-period 600 \
    --vpc-zone-identifier "subnet-5ea0c127,subnet-6194ea3b,subnet-c934b782"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Elastic Load Balancing 및 Amazon Auto Scaling을 참조하세요. EC2 Auto Scaling

예 3: 배치 그룹 및 종료 정책을 업데이트하는 방법

이 예시에서는 사용할 배치 그룹 및 종료 정책을 업데이트합니다.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --placement-group my-placement-group \
    --termination-policies "OldestInstance"
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 Auto Scaling 그룹을 참조하세요. EC2 Auto Scaling

예 4: 시작 템플릿의 최신 버전을 사용하는 방법

이 예시에서는 지정된 시작 템플릿의 최신 버전을 사용하도록 지정된 Auto Scaling 그룹을 업데이트합니다.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateId=lt-1234567890abcde12,Version='$Latest'
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 시작 템플릿을 참조하세요.

예 5: 시작 템플릿의 특정 버전을 사용하는 방법

이 예시에서는 시작 템플릿의 최신 또는 기본 버전 대신 특정 버전을 사용하도록 지정된 Auto Scaling 그룹을 업데이트합니다.

```
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name my-asg \
    --launch-template LaunchTemplateName=my-template-for-auto-scaling,Version='2'
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 시작 템플릿을 참조하세요.

예 6: 혼합 인스턴스 정책을 정의하고 용량 재분배를 활성화하는 방법

이 예시에서는 혼합 인스턴스 정책을 사용하도록 지정된 Auto Scaling 그룹을 업데이트하고 용량 재분배를 활성화합니다. 이 구조를 통해 스팟 및 온디맨드 용량을 사용하는 그룹을 지정하고 아키텍처마다 다른 시작 템플릿을 사용할 수 있습니다.

```
aws autoscaling update-auto-scaling-group \
    --cli-input-json file://~/config.json
```

`config.json`의 콘텐츠:

```
{
    "AutoScalingGroupName": "my-asg",
    "CapacityRebalance": true,
    "MixedInstancesPolicy": {
        "LaunchTemplate": {
            "LaunchTemplateSpecification": {
                "LaunchTemplateName": "my-launch-template-for-x86",
                "Version": "$Latest"
            },
            "Overrides": [
                {
                    "InstanceType": "c6g.large",
                    "LaunchTemplateSpecification": {
                        "LaunchTemplateName": "my-launch-template-for-arm",
                        "Version": "$Latest"
                    }
                },
                {
                    "InstanceType": "c5.large"
                },
                {
```

```
                    "InstanceType": "c5a.large"
                }
            ]
        },
        "InstancesDistribution": {
            "OnDemandPercentageAboveBaseCapacity": 50,
            "SpotAllocationStrategy": "capacity-optimized"
        }
    }
}
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 Amazon Auto Scaling 사용 설명서의 여러 인스턴스 유형 및 구매 옵션이 있는 Auto Scaling 그룹을 참조하세요. EC2 Auto Scaling

- 자세한 API 내용은 명령 참조UpdateAutoScalingGroup의 섹션을 참조하세요. AWS CLI

Java

SDK Java 2.x용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static void updateAutoScalingGroup(AutoScalingClient
autoScalingClient, String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
```

```
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
 DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
 waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group
 " + groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 자세한 API 내용은 참조UpdateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for Java 2.x API

Kotlin

SDK Kotlin용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun updateAutoScalingGroup(
    groupName: String,
    launchTemplateNameVal: String,
    serviceLinkedRoleARNVal: String,
) {
```

```
    val templateSpecification =
        LaunchTemplateSpecification {
            launchTemplateName = launchTemplateNameVal
        }

    val groupRequest =
        UpdateAutoScalingGroupRequest {
            maxSize = 3
            serviceLinkedRoleArn = serviceLinkedRoleARNVal
            autoScalingGroupName = groupName
            launchTemplate = templateSpecification
        }

    val groupsRequestWaiter =
        DescribeAutoScalingGroupsRequest {
            autoScalingGroupNames = listOf(groupName)
        }

    AutoScalingClient { region = "us-east-1" }.use { autoScalingClient ->
        autoScalingClient.updateAutoScalingGroup(groupRequest)
        autoScalingClient.waitUntilGroupExists(groupsRequestWaiter)
        println("You successfully updated the Auto Scaling group  $groupName")
    }
}
```

- API 자세한 내용은 UpdateAutoScalingGroup의 에서 AWS SDK Kotlin API 참조 를 참조하세요.

PHP

PHP용 SDK

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public function updateAutoScalingGroup($autoScalingGroupName, $args)
    {
```

```
        if (array_key_exists('MaxSize', $args)) {
            $maxSize = ['MaxSize' => $args['MaxSize']];
        } else {
            $maxSize = [];
        }
        if (array_key_exists('MinSize', $args)) {
            $minSize = ['MinSize' => $args['MinSize']];
        } else {
            $minSize = [];
        }
        $parameters = ['AutoScalingGroupName' => $autoScalingGroupName];
        $parameters = array_merge($parameters, $minSize, $maxSize);
        return $this->autoScalingClient->updateAutoScalingGroup($parameters);
    }
```

- 자세한 API 내용은 참조UpdateAutoScalingGroup의 섹션을 참조하세요. AWS SDK for PHP API

PowerShell

용 도구 PowerShell

예제 1: 이 예제는 지정된 Auto Scaling 그룹의 최소 및 최대 크기를 업데이트합니다.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -MaxSize 5 -MinSize 1
```

예제 2: 이 예제는 지정된 Auto Scaling 그룹의 기본 휴지 기간을 업데이트합니다.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -DefaultCooldown 10
```

예제 3: 이 예제는 지정된 Auto Scaling 그룹의 가용 영역을 업데이트합니다.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -AvailabilityZone @("us-
west-2a", "us-west-2b")
```

예제 4: 이 예제는 Elastic Load Balancing 상태 확인을 사용하도록 지정된 Auto Scaling 그룹을 업데이트합니다.

```
Update-ASAutoScalingGroup -AutoScalingGroupName my-asg -HealthCheckType ELB -
HealthCheckGracePeriod 60
```

- API 자세한 내용은 Cmdlet 참조UpdateAutoScalingGroup의 섹션을 참조하세요. AWS Tools for PowerShell

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class AutoScalingWrapper:
    """Encapsulates Amazon EC2 Auto Scaling actions."""

    def __init__(self, autoscaling_client):
        """
        :param autoscaling_client: A Boto3 Amazon EC2 Auto Scaling client.
        """
        self.autoscaling_client = autoscaling_client


    def update_group(self, group_name: str, **kwargs: Any) -> None:
        """
        Updates an Auto Scaling group.

        :param group_name: The name of the group to update.
        :param kwargs: Keyword arguments to pass through to the service.
        :return: None
        :raises ClientError: If there is an error updating the Auto Scaling
 group.
        """
        try:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, **kwargs
            )
            logger.info(f"Successfully updated Auto Scaling group {group_name}.")
```

```
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            logger.error(f"Failed to update Auto Scaling group {group_name}.")
            if error_code == "ResourceInUse":
                logger.error(
                    "The Auto Scaling group '%s' is currently in use and cannot
 be modified. Please try again later.",
                    group_name,
                )
            elif error_code == "ScalingActivityInProgress":
                logger.error(
                    f"A scaling activity is currently in progress for the Auto
 Scaling group '{group_name}'."
                    "Please wait for the activity to complete before attempting
 to update the group."
                )
            logger.error(f"Full error:\n\t{err}")
            raise
```

- API 자세한 내용은 UpdateAutoScalingGroup의 AWS SDK Python(Boto3) API 참조 섹션을 참조하세요.

Rust

SDK Rust용

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn update_group(client: &Client, name: &str, size: i32) -> Result<(),
 Error> {
    client
        .update_auto_scaling_group()
        .auto_scaling_group_name(name)
        .max_size(size)
```

```
        .send()
        .await?;

    println!("Updated AutoScaling group");

    Ok(())
}
```

- API 자세한 내용은 Rust 참조 UpdateAutoScalingGroup 의 섹션을 참조하세요. AWS SDK API

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

를 사용한 Auto Scaling 시나리오 AWS SDKs

다음 코드 예제에서는 를 사용하여 Auto Scaling에서 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS SDKs. 이 시나리오에서는 Auto Scaling 내에서 여러 함수를 호출하거나 다른 와 결합하여 특정 작업을 수행하는 방법을 보여줍니다 AWS 서비스. 각 시나리오에는 전체 소스 코드에 대한 링크가 포함되어 있습니다. 여기에서 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

시나리오는 컨텍스트에 맞는 서비스 작업을 이해하는 데 도움이 되도록 중급 수준의 경험을 대상으로 합니다.

예시

- 를 사용하여 탄력적 서비스 구축 및 관리 AWS SDK

를 사용하여 탄력적 서비스 구축 및 관리 AWS SDK

다음 코드 예제는 책, 영화, 노래 추천을 반환하는 로드 밸런싱 웹 서비스를 만드는 방법을 보여줍니다. 이 예제에서는 서비스가 장애에 대응하는 방법과 장애 발생 시 복원력을 높이기 위해 서비스를 재구성하는 방법을 보여줍니다.

- Amazon EC2 Auto Scaling 그룹을 사용하여 시작 템플릿을 기반으로 Amazon Elastic Compute Cloud(AmazonEC2) 인스턴스를 생성하고 인스턴스 수를 지정된 범위로 유지합니다.
- Elastic Load Balancing 을 사용하여 HTTP 요청을 처리하고 배포합니다.
- Auto Scaling 그룹의 인스턴스 상태를 모니터링하고 요청을 정상 인스턴스로만 전달합니다.

- 각 EC2 인스턴스에서 Python 웹 서버를 실행하여 HTTP 요청을 처리합니다. 웹 서버는 추천 및 상태 확인으로 응답합니다.
- Amazon DynamoDB 테이블을 사용하여 추천 서비스를 시뮬레이션합니다.
- AWS Systems Manager 파라미터를 업데이트하여 요청 및 상태 확인에 대한 웹 서버 응답을 제어합니다.

.NET

AWS SDK for .NET

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
```

```
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
```

```
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
```

```
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
```

```
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
```

```
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
```

```
                    "\nFor this example to work, the default security group for
your default VPC must\n"
                    + "allows access from this computer. You can either add it
automatically from this\n"
                    + "example or add it yourself using the AWS Management
Console.\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
```

```
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
```

```
                            $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
                            "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
```

```
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
```

```
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");
```

```
        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
```

```
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

Auto Scaling 및 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
```

```
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
 a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
 granted to
```

```
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
 the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                     "\"Service\": [" +
                                         "\"ec2.amazonaws.com\"" +
                                     "]" +
                                     "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
```

```
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
```

```
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
 _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
```

```
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
 Scaling.
    /// The launch template specifies a Bash script in its user data field that
 runs after
```

```
   /// the instance is started. This script installs the Python packages and
starts a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
                           {
                               Name = _instanceProfileName
                           },
                       KeyName = _keyPairName,
                       UserData = System.Convert.ToBase64String(plainTextBytes)
```

```
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
 zones.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
 exists.");
        }
```

```
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
 permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
 {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
```

```
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
```

```
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
```

```
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
```

```
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);
```

```
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
```

```
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
```

```
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
```

```
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
```

```
                    Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                        "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
```

```
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
```

```
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

Elastic Load Balancing 작업을 래핑하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }
```

```
    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
```

```
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
```

```
    {
        var createResponse = await
_amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
```

```
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
```

```
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
```

```
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
 describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
 _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
```

```
                    new DeleteTargetGroupRequest() { TargetGroupArn =
 targetArn });
                Console.WriteLine($"Deleted load balancing target group
 {groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
```

```
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
```

```
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
```

```
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
```

```
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- API 자세한 내용은 AWS SDK for .NET API 참조 의 다음 주제를 참조하세요.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations

- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

SDK Java 2.x용

> **Note**
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
```

```
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
```

```
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
```

```
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
```

```
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
```

```
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
 InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
 different parts of the system
                        to create situations where the web service fails, and
 shows how using a resilient
                        architecture can keep the web service running in spite
 of these failures.

                        At the start, the load balancer endpoint returns
 recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
 recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
 parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
 let's set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
 endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
 checks don't check for failure of the recommendation service.
                       """);
```

```
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
```

```
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);
```

```
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load
balancing rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance
is running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
```

```
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();
```

```
                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
balancer targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
```

```
    }
}
```

Auto Scaling 및 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
```

```
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
```

```
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
```

```
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
```

```
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
```

```
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
```

```
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }
```

```
                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();
```

```
getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
```

```
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
```

```
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
```

```
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
```

```
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();
```

```
        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

Elastic Load Balancing 작업을 래핑하는 클래스를 생성합니다.

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
 getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
```

```
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
```

```
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load
balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
```

```
    * Creates an Elastic Load Balancing target group. The target group specifies
    * how
    * the load balancer forward requests to instances in the group and how
instance
    * health is checked.
    */
   public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
       CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
               .healthCheckPath("/healthcheck")
               .healthCheckTimeoutSeconds(5)
               .port(port)
               .vpcId(vpcId)
               .name(targetGroupName)
               .protocol(protocol)
               .build();

       CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
       String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
       String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
       System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
       return targetGroupArn;
   }

   /*
    * Creates an Elastic Load Balancing load balancer that uses the specified
    * subnets
    * and forwards requests to the specified target group.
    */
   public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
```

```
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();
```

```
            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
```

```
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
```

```
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);
```

```
        DynamoDbTable<Recommendation> mappedTable =
 enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
```

```
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- API 자세한 내용은 AWS SDK for Java 2.x API 참조 의 다음 주제를 참조하세요.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups

- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

JavaScript

SDK 용 JavaScript (v3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0


import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
```

```
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios);
}
```

모든 리소스를 배포하기 위한 단계를 생성합니다.

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
```

```
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
```

```
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
```

```
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
```

```
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
```

```
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
```

```
    ),
    new ScenarioAction("createInstanceProfile", async (state) => {
      const client = new IAMClient({});
      const {
        InstanceProfile: { Arn },
      } = await client.send(
        new CreateInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
      state.instanceProfileArn = Arn;

      await waitUntilInstanceProfileExists(
        { client },
        { InstanceProfileName: NAMES.instanceProfileName },
      );
    }),
    new ScenarioOutput("createdInstanceProfile", (state) =>
      MESSAGES.createdInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
    ),
    new ScenarioOutput(
      "addingRoleToInstanceProfile",
      MESSAGES.addingRoleToInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
    ),
    new ScenarioAction("addRoleToInstanceProfile", () => {
      const client = new IAMClient({});
      return client.send(
        new AddRoleToInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    }),
    new ScenarioOutput(
      "addedRoleToInstanceProfile",
      MESSAGES.addedRoleToInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
    ),
    ...initParamsSteps,
```

```
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
```

```
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
```

```
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
```

```
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
```

```
    new ScenarioAction("createListener", async (state) => {
      const client = new ElasticLoadBalancingV2Client({});
      const { Listeners } = await client.send(
        new CreateListenerCommand({
          LoadBalancerArn: state.loadBalancerArn,
          Protocol: state.targetGroupProtocol,
          Port: state.targetGroupPort,
          DefaultActions: [
            { Type: "forward", TargetGroupArn: state.targetGroupArn },
          ],
        }),
      );
      const listener = Listeners[0];
      state.loadBalancerListenerArn = listener.ListenerArn;
    }),
    new ScenarioOutput("createdListener", (state) =>
      MESSAGES.createdLoadBalancerListener.replace(
        "${LB_LISTENER_ARN}",
        state.loadBalancerListenerArn,
      ),
    ),
    new ScenarioOutput(
      "attachingLoadBalancerTargetGroup",
      MESSAGES.attachingLoadBalancerTargetGroup
        .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
    ),
    new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new AttachLoadBalancerTargetGroupsCommand({
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          TargetGroupARNs: [state.targetGroupArn],
        }),
      );
    }),
    new ScenarioOutput(
      "attachedLoadBalancerTargetGroup",
      MESSAGES.attachedLoadBalancerTargetGroup,
    ),
    new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
    new ScenarioAction(
      "verifyInboundPort",
      /**
```

```
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
```

```
        );
      } else {
        return MESSAGES.noIpRules;
      }
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      } else {
        return MESSAGES.noIpRules;
      }
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
```

```
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    } else {
      return false;
    }
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

데모를 실행하기 위한 단계를 생성합니다.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
```

```
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
```

```
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
```

```
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
```

```
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
```

```
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
```

```
    await createSsmOnlyInstanceProfile();
    const autoScalingClient = new AutoScalingClient({});
    const { AutoScalingGroups } = await autoScalingClient.send(
      new DescribeAutoScalingGroupsCommand({
        AutoScalingGroupNames: [NAMES.autoScalingGroupName],
      }),
    );
    state.targetInstance = AutoScalingGroups[0].Instances[0];
    const ec2Client = new EC2Client({});
    const { IamInstanceProfileAssociations } = await ec2Client.send(
      new DescribeIamInstanceProfileAssociationsCommand({
        Filters: [
          { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
        ],
      }),
    );
    state.instanceProfileAssociationId =
      IamInstanceProfileAssociations[0].AssociationId;
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      ec2Client.send(
        new ReplaceIamInstanceProfileAssociationCommand({
          AssociationId: state.instanceProfileAssociationId,
          IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
        }),
      ),
    );

    await ec2Client.send(
      new RebootInstancesCommand({
        InstanceIds: [state.targetInstance.InstanceId],
      }),
    );

    const ssmClient = new SSMClient({});
    await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
      const { InstanceInformationList } = await ssmClient.send(
        new DescribeInstanceInformationCommand({}),
      );

      const instance = InstanceInformationList.find(
        (info) => info.InstanceId === state.targetInstance.InstanceId,
      );

      if (!instance) {
```

```
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
```

```
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
```

```
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
```

```
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
```

```
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

모든 리소스를 폐기하는 단계를 생성합니다.

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
```

```
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    } else {
```

```
      return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
    }
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    } else {
      return MESSAGES.deletedKeyPair.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
```

```
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    } else {
      return MESSAGES.detachedPolicyFromRole
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    } else {
      return MESSAGES.deletedPolicy.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
```

```
      );
    }
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    } else {
      return MESSAGES.removedRoleFromInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
```

```
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    } else {
      return MESSAGES.deletedInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    } else {
      return MESSAGES.deletedInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
```

```
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    } else {
      return MESSAGES.deletedAutoScalingGroup.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    } else {
      return MESSAGES.deletedLaunchTemplate.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
```

```
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    } else {
      return MESSAGES.deletedLoadBalancer.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
```

```
        }),
      ),
    );
  } catch (e) {
    state.deleteLoadBalancerTargetGroupError = e;
  }
}),
new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
  if (state.deleteLoadBalancerTargetGroupError) {
    console.error(state.deleteLoadBalancerTargetGroupError);
    return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  } else {
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }
}),
new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new RemoveRoleFromInstanceProfileCommand({
        InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        RoleName: NAMES.ssmOnlyRoleName,
      }),
    );
  } catch (e) {
    state.detachSsmOnlyRoleFromProfileError = e;
  }
}),
new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
  if (state.detachSsmOnlyRoleFromProfileError) {
    console.error(state.detachSsmOnlyRoleFromProfileError);
    return MESSAGES.detachSsmOnlyRoleFromProfileError
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  } else {
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
```

```
    }
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    } else {
      return MESSAGES.detachedSsmOnlyCustomRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
```

```
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    } else {
      return MESSAGES.detachedSsmOnlyAWSRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
```

```
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyRole.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
  }),
```

```
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
 state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    } else {
      return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
    }
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
```

```
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    } else {
      console.log(err.name);
      throw err;
    }
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
```

```
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- API 자세한 내용은 AWS SDK for JavaScript API 참조 의 다음 주제를 참조하세요.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones

- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Python

SDK Python용(Boto3)

> **Note**
>
> 에 대한 자세한 내용은 를 참조하세요 GitHub. AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
```

```
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)
```

```
        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
 Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()

        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
```

```
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
```

```
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
```

```
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()

        logging.info(
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
```

```
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
```

```
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
manually to avoid unexpected charges."
            )


def main() -> None:
```

```
    """
    Main function to parse arguments and run the appropriate actions for the
demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../workflows/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
```

```
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Auto Scaling 및 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
```

```
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
```

```
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
 definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' created successfully. ARN:
 {policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
```

```
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")

    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
```

```
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
```

```
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
```

```
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")


    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
```

```
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
```

```
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
```

```
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
```

```
                                        to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
```

```
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
```

```
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
```

```
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
```

```
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.
```

```
        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
```

```
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
 to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
```

```
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
 "
                    "Ensure that your AWS IAM user or role has the correct
 permissions."
                )
            elif error_code == "InvalidParameterValue":
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.
```

```
        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
```

```
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
```

```
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
```

```
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

Elastic Load Balancing 작업을 래핑하는 클래스를 생성합니다.

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
 specifies how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
 times and
        lower thresholds. In production, you might want to decrease the
 sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
```

```
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
```

```
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)

            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
```

```
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
```

```
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
```

```
                                load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
```

```
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.

        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.
```

```
        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
```

```
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
                    "Ensure the load balancer exists, is in the correct AWS
region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
 parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """
        Creates a DynamoDB table to use as a recommendation service. The table
 has a
        hash key named 'MediaType' that defines the type of media recommended,
 such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
 MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
```

```
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )
```

```
    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.
```

```
        :param table_name: The name of the DynamoDB table that is used as a
recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
```

```
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- API 자세한 내용은 의 AWS SDK Python(Boto3) API 참조에 대한 다음 주제를 참조하세요.
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation

- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

개발자 가이드 및 코드 예제의 AWS SDK 전체 목록은 섹션을 참조하세요에서 이 서비스 사용 AWS SDK. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon EC2 Auto Scaling의 문제 해결

Amazon EC2 Auto Scaling은 문제를 해결하는 데 도움이 되는 구체적이고 설명적인 오류를 제공합니다. 크기 조정 활동에 관한 설명에서 오류 메시지를 확인할 수 있습니다.

주제

- 크기 조정 활동에서 오류 메시지 검색
- 스케일링 활동을 끕니다.
- 추가 문제 해결 리소스
- Amazon EC2 Auto Scaling 문제 해결: EC2 인스턴스 시작 실패
- Amazon EC2 Auto Scaling 문제 해결: AMI 문제
- Amazon EC2 Auto Scaling: 로드 밸런서 문제 해결
- Amazon EC2 Auto Scaling: 시작 템플릿 문제 해결

크기 조정 활동에서 오류 메시지 검색

조정 활동 설명에서 오류 메시지를 검색하려면 명령을 사용하십시오. describe-scaling-activities 6주 이전까지 크기 조정 활동에 대한 레코드가 있습니다. 크기 조정 활동은 시작 시간을 기준으로 정렬되며, 가장 최근의 크기 조정 활동이 가장 먼저 열거됩니다.

> **Note**
>
> 조정 활동은 Amazon Auto Scaling 콘솔의 EC2 Auto Scaling 그룹에 대한 활동 탭의 활동 기록에도 표시됩니다.

특정 Auto Scaling 그룹에 대한 크기 조정 활동을 보려면 다음 명령을 사용합니다.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg
```

다음은 `StatusCode`에 활동의 현재 상태가 포함되고 `StatusMessage`에 오류 메시지가 포함된 응답의 예입니다.

```
{
```

```
    "Activities": [
        {
            "ActivityId": "3b05dbf6-037c-b92f-133f-38275269dc0f",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance: i-003a5b3ffe1e9358e.  Status
 Reason: Instance failed to complete user's Lifecycle Action: Lifecycle Action with
 token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned: Lifecycle Action Completed
 with ABANDON Result",
            "Cause": "At 2021-01-11T00:35:52Z a user request created an
 AutoScalingGroup changing the desired capacity from 0 to 1.  At 2021-01-11T00:35:53Z
 an instance was started in response to a difference between desired and actual
 capacity, increasing the capacity from 0 to 1.",
            "StartTime": "2021-01-11T00:35:55.542Z",
            "EndTime": "2021-01-11T01:06:31Z",
            "StatusCode": "Cancelled",
            "StatusMessage": "Instance failed to complete user's Lifecycle Action:
 Lifecycle Action with token e85eb647-4fe0-4909-b341-a6c42d8aba1f was abandoned:
 Lifecycle Action Completed with ABANDON Result",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
     ...
    ]
}
```

출력의 필드에 대한 설명은 Amazon EC2 Auto Scaling API 참조의 활동을 참조하십시오.

삭제된 그룹의 조정 활동을 보려면

Auto Scaling 그룹이 삭제된 후 조정 활동을 보려면 다음과 같이 describe-scaling-activities명령에 `--include-deleted-groups` 옵션을 추가합니다.

```
aws autoscaling describe-scaling-activities --auto-scaling-group-name my-asg --include-
deleted-groups
```

다음은 삭제된 그룹에 대한 크기 조정 활동이 포함된 응답의 예입니다.

```
{
```

```
    "Activities": [
        {
            "ActivityId": "e1f5de0e-f93e-1417-34ac-092a76fba220",
            "AutoScalingGroupName": "my-asg",
            "Description": "Launching a new EC2 instance.  Status Reason: Your Spot
 request price of 0.001 is lower than the minimum required Spot request fulfillment
 price of 0.0031. Launching EC2 instance failed.",
            "Cause": "At 2021-01-13T20:47:24Z a user request update of AutoScalingGroup
 constraints to min: 1, max: 5, desired: 3 changing the desired capacity from 0 to 3.
  At 2021-01-13T20:47:27Z an instance was started in response to a difference between
 desired and actual capacity, increasing the capacity from 0 to 3.",
            "StartTime": "2021-01-13T20:47:30.094Z",
            "EndTime": "2021-01-13T20:47:30Z",
            "StatusCode": "Failed",
            "StatusMessage": "Your Spot request price of 0.001 is lower than the
 minimum required Spot request fulfillment price of 0.0031. Launching EC2 instance
 failed.",
            "Progress": 100,
            "Details": "{\"Subnet ID\":\"subnet-5ea0c127\",\"Availability Zone\":\"us-
west-2b\"...}",
            "AutoScalingGroupState": "Deleted",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-
west-2:123456789012:autoScalingGroup:283179a2-
f3ce-423d-93f6-66bb518232f7:autoScalingGroupName/my-asg"
        },
     ...
    ]
}
```

스케일링 활동을 끕니다.

규모 조정 정책 또는 예정된 조치의 방해 없이 문제를 조사해야 하는 경우 다음과 같은 옵션을 사용할 수 있습니다.

- 및 `ScheduledActions` 프로세스를 일시 중단하여 모든 동적 조정 정책 및 예약된 작업으로 인해 그룹의 원하는 용량이 변경되지 않도록 하십시오. `AlarmNotification` 자세한 내용은 Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개 단원을 참조하십시오.
- 개별 동적 조정 정책을 비활성화하여 부하 변화에 따라 그룹이 원하는 용량을 변경하지 않도록 합니다. 자세한 내용은 Auto Scaling 그룹에 대한 조정 정책 비활성화 단원을 참조하십시오.
- 정책의 스케일 인 부분을 비활성화하여 스케일 아웃 (용량 추가) 만 하도록 개별 대상 추적 조정 정책을 업데이트하십시오. 이 방법을 사용하면 그룹의 원하는 용량이 줄어드는 것을 방지할 수 있지만 부

하가 증가할 때 용량을 늘릴 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling의 대상 추적 조정 정책 단원을 참조하십시오.

- 예측 조정 정책을 예측 전용 모드로 업데이트하십시오. 예측 전용 모드에서는 예측 스케일링을 통해 계속해서 예측치가 생성되지만 용량이 자동으로 증가하지는 않습니다. 자세한 내용은 Auto Scaling 그룹에 대한 예측 조정 정책 생성 단원을 참조하십시오.

추가 문제 해결 리소스

다음 페이지는 Amazon EC2 Auto Scaling과 관련된 문제를 해결하기 위한 추가 정보를 제공합니다.

- Auto Scaling 그룹에 대한 크기 조정 활동 확인
- Amazon EC2 Auto Scaling 콘솔에서 모니터링 그래프 보기
- Auto Scaling 그룹의 인스턴스에 대한 상태 확인
- 수명 주기 후크에 대한 고려 사항 및 제한 사항
- Auto Scaling 그룹에서 라이프사이클 작업 완료
- Amazon VPC를 사용하여 Auto Scaling 인스턴스에 네트워크 연결 제공
- Auto Scaling 그룹에서 일시적으로 인스턴스 제거
- Auto Scaling 그룹에 대한 조정 정책 비활성화
- Amazon EC2 Auto Scaling 프로세스 일시 중지 및 재개
- 축소 시 해지할 Auto Scaling 인스턴스 제어
- Auto Scaling 인프라 삭제
- Auto Scaling 리소스 및 그룹 할당량

다음 AWS 리소스도 도움이 될 수 있습니다.

- AWS 지식 센터의 Amazon EC2 Auto Scaling 주제
- AWS re:Post에 대한 Amazon EC2 Auto Scaling 관련 질문
- Amazon EC2 Auto Scaling이 AWS 컴퓨트 블로그에 게시한 게시물
- AWS CloudFormation 사용 CloudFormation 설명서의 문제 해결

문제 해결에는 종종 전문가 또는 도우미 커뮤니티의 반복적인 질의와 발견이 필요한 경우가 많습니다. 이 섹션의 제안 사항을 시도한 후에도 문제가 계속 발생하는 경우 Amazon EC2 Auto Scaling 태그를

사용하여 re:Post에 문의하거나 AWS Support (지원 센터 클릭) AWS re:Post에서 질문하십시오. AWS Management Console

Amazon EC2 Auto Scaling 문제 해결: EC2 인스턴스 시작 실패

이 페이지에서는 시작에 실패한 EC2 인스턴스, 잠재적 원인, 문제 해결을 위해 취할 수 있는 조치에 대한 정보를 제공합니다.

오류 메시지를 검색하려면 크기 조정 활동에서 오류 메시지 검색를 참조하세요.

EC2인스턴스 시작에 실패할 경우 다음 오류 메시지 중 하나 이상이 표시될 수 있습니다.

시작 문제

- 요청된 구성이 현재 지원되지 않습니다.
- 보안 그룹 <보안 그룹 명칭>이(가) 존재하지 않습니다. EC2인스턴스 시작에 실패했습니다.
- 키 페어 EC2 <인스턴스에 연결된 키 페어>가 존재하지 않습니다. 인스턴스 EC2 시작에 실패했습니다.
- 요청된 인스턴스 유형(<인스턴스 유형>)이 요청된 가용 영역(<인스턴스 가용 영역>)에서 지원되지 않습니다...
- 스팟 요청 가격인 0.015가 필요한 최소 스팟 요청 이행 가격인 0.0735보다 낮습니다.
- 잘못된 디바이스 명칭 <device name>/잘못된 디바이스 명칭 업로드. EC2인스턴스 시작에 실패했습니다.
- <name associated with the instance storage device>파라미터 값 () virtualName 이 유효하지 않습니다... EC2인스턴스 시작에 실패했습니다.
- EBS블록 디바이스 매핑은 인스턴스 스토어에 지원되지 않습니다. AMIs
- 배치 그룹은 유형 '<인스턴스 유형>'의 인스턴스와 함께 사용할 수 없습니다. EC2인스턴스 시작에 실패했습니다.
- 클라이언트. InternalError: 시작 시 클라이언트 오류가 발생했습니다.
- 요청한 가용 영역에 현재 <인스턴스 유형> 용량이 부족합니다... EC2인스턴스 시작에 실패했습니다.
- 요청된 예약에는 이 요청에 사용할 수 있는 호환 및 사용 가능한 용량이 충분하지 않습니다. EC2인스턴스 시작에 실패했습니다.
- 용량 블록 <reservation id> 예약이 아직 활성화되지 않았습니다. EC2인스턴스 시작에 실패했습니다.
- 요청과 일치하는 스팟 용량이 없습니다. EC2인스턴스 시작에 실패했습니다.

- <인스턴스 수>개의 인스턴스가 이미 실행 중입니다. EC2인스턴스 시작에 실패했습니다.

요청된 구성이 현재 지원되지 않습니다.

원인: 시작 템플릿 또는 시작 구성의 일부 옵션이 인스턴스 유형과 호환되지 않거나 요청한 AWS 지역 또는 가용 영역에서 인스턴스 구성이 지원되지 않을 수 있습니다.

해결 방법: 다른 인스턴스 구성을 사용해 보십시오. 요구 사항을 충족하는 인스턴스 유형을 검색하려면 Amazon EC2사용 설명서에서 Amazon EC2 인스턴스 유형 찾기를 참조하십시오.

이 문제를 해결하기 위한 추가 지침은 다음 내용을 확인하세요.

- 인스턴스 유형에서 AMI 지원하는 인스턴스를 선택했는지 확인하십시오. 예를 들어 인스턴스 유형이 인텔 제온 프로세서 대신 ARM 기반 AWS Graviton 프로세서를 사용하는 경우 ARM과 호환되는 프로세서가 필요합니다. AMI 호환되는 인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 Amazon EC2 User Guide의 인스턴스 유형 변경을 위한 호환성을 참조하십시오.
- 요청한 지역 및 가용 영역에서 인스턴스 유형을 사용할 수 있는지 테스트합니다. 최신 세대 인스턴스 유형을 지정된 지역 또는 가용 영역에서 아직 사용하지 못할 수 있습니다. 이전 세대 인스턴스 유형을 최신 지역 또는 가용 영역에서 아직 사용하지 못할 수 있습니다. 위치 (지역 또는 가용 영역) 별로 제공되는 인스턴스 유형을 검색하려면 describe-instance-type-offerings명령을 사용합니다. 자세한 내용은 Amazon EC2사용 설명서에서 Amazon EC2 인스턴스 유형 찾기를 참조하십시오.
- 전용 인스턴스 또는 전용 호스트를 사용하는 경우, 전용 인스턴스 또는 전용 호스트로 지원되는 인스턴스 유형을 선택했는지 확인합니다.

보안 그룹 <보안 그룹 명칭>이(가) 존재하지 않습니다. EC2인스턴스 시작에 실패했습니다.

원인: 시작 템플릿 또는 시작 구성에 지정된 보안 그룹이 삭제되었을 수 있습니다.

해결 방법:

1. describe-security-groups명령어를 사용하여 계정과 연결된 보안 그룹 목록을 가져올 수 있습니다.
2. 목록에서 사용할 보안 그룹을 선택합니다. 대신 보안 그룹을 만들려면 create-security-group명령어를 사용하세요.
3. 새 시작 템플릿 또는 시작 구성을 생성합니다.
4. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

키 페어 EC2 <인스턴스에 연결된 키 페어>가 존재하지 않습니다. 인스턴스 EC2 시작에 실패했습니다.

원인: 인스턴스를 시작할 때 사용한 키 페어가 삭제되었을 수 있습니다.

해결 방법:

1. describe-key-pairs명령을 사용하여 사용 가능한 키 페어 목록을 가져올 수 있습니다.
2. 목록에서 사용할 키 페어를 선택합니다. 대신 key Pair를 생성하려면 create-key-pair명령을 사용하십시오.
3. 새 시작 템플릿 또는 시작 구성을 생성합니다.
4. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

요청된 인스턴스 유형(<인스턴스 유형>)이 요청된 가용 영역(<인스턴스 가용 영역>)에서 지원되지 않습니다...

오류 메시지: 요청한 인스턴스 유형 (<instance type>) 이 요청된 가용 영역 (<instance Availability Zone>) 에서 지원되지 않습니다... EC2 인스턴스 시작에 실패했습니다.

원인: Auto Scaling 그룹에 지정된 가용성 영역이 선택한 인스턴스 유형을 지원하지 않습니다.

해결 방법:

1. describe-instance-type-offerings명령을 사용하거나 Amazon EC2 콘솔에서 인스턴스 유형 페이지의 네트워킹 창에서 가용 영역 값을 확인하여 선택한 인스턴스 유형을 지원하는 가용 영역을 확인합니다.
2. 명령을 사용하여 Auto Scaling 그룹 설정에서 지원되지 않는 영역의 서브넷을 업데이트하거나 제거합니다. update-auto-scaling-group 자세한 내용은 가용 영역 추가 단원을 참조하십시오.

스팟 요청 가격인 0.015가 필요한 최소 스팟 요청 이행 가격인 0.0735보다 낮습니다.

원인: 요청의 스팟 최고 가격이 선택한 인스턴스 유형의 스팟 가격보다 낮습니다.

솔루션: 더 높은 스팟 최고 가격(온디맨드 가격)으로 새 요청을 제출합니다. 이전에 지불한 스팟 가격은 입찰을 기준으로 책정되었습니다. 오늘은 현재 스팟 가격을 지불합니다. 최고 가격을 더 높게 설정하

면 Amazon EC2 Spot 서비스를 시작하고 필요한 용량을 유지할 수 있는 더 나은 기회를 얻을 수 있습니다.

잘못된 디바이스 명칭 <device name>/잘못된 디바이스 명칭 업로드. EC2인스턴스 시작에 실패했습니다.

원인 1: 시작 템플릿 또는 시작 구성의 블록 디바이스 매핑에 사용할 수 없거나 현재 지원되지 않는 블록 디바이스 명칭이 포함되었을 수 있습니다.

해결 방법:

1. 특정 인스턴스 구성에 사용할 수 있는 디바이스 명칭을 확인합니다. 디바이스 이름 지정에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Linux 인스턴스의 디바이스 이름을 참조하십시오.
2. Auto Scaling 그룹에 속하지 않는 Amazon EC2 인스턴스를 수동으로 생성하고 문제를 조사하십시오. 블록 디바이스 이름 지정 구성이 Amazon Machine Image (AMI) 의 이름과 충돌하는 경우 인스턴스 시작에 실패합니다. 자세한 내용은 Amazon EC2 사용 설명서의 블록 디바이스 매핑을 참조하십시오.
3. 인스턴스가 성공적으로 시작된 것을 확인한 후 describe-volumes 명령을 사용하여 볼륨이 인스턴스에 어떻게 노출되는지 확인합니다.
4. 볼륨 설명에 나와 있는 디바이스 명칭을 사용하여 새 시작 템플릿 또는 시작 구성을 생성합니다.
5. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

<name associated with the instance storage device>파라미터 값 () virtualName 이 유효하지 않습니다... EC2인스턴스 시작에 실패했습니다.

원인: 블록 디바이스와 연결된 가상 이름에 지정된 형식이 잘못되었습니다.

해결 방법:

1. `virtualName` 파라미터의 디바이스 명칭을 지정하여 새 시작 템플릿 또는 시작 구성을 생성합니다. 디바이스 이름 형식에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Linux 인스턴스에서의 디바이스 이름 지정을 참조하십시오.
2. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

EBS블록 디바이스 매핑은 인스턴스 스토어에 지원되지 않습니다. AMIs

원인: 시작 템플릿 또는 시작 구성에 지정된 블록 디바이스 매핑이 사용자 인스턴스에 대해 지원되지 않습니다.

해결 방법:

1. 사용자 인스턴스 유형에서 지원되는 블록 디바이스 매핑으로 새 시작 템플릿 또는 시작 구성을 생성합니다. 자세한 내용은 Amazon EC2 사용 설명서의 블록 디바이스 매핑을 참조하십시오.
2. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

배치 그룹은 유형 '<인스턴스 유형>'의 인스턴스와 함께 사용할 수 없습니다. EC2인스턴스 시작에 실패했습니다.

원인: 클러스터 배치 그룹에 잘못된 인스턴스 유형이 포함되어 있습니다.

해결 방법:

1. 배치 그룹에서 지원하는 유효한 인스턴스 유형에 대한 자세한 내용은 Amazon EC2 User Guide의 배치 그룹을 참조하십시오.
2. 배치 그룹에 자세히 나와 있는 지침에 따라 새 배치 그룹을 생성합니다.
3. 또는 지원되는 인스턴스 유형을 사용하여 새 시작 템플릿 또는 시작 구성을 생성합니다.
4. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 배치 그룹, 시작 템플릿 또는 시작 구성으로 업데이트하십시오.

클라이언트. InternalError: 시작 시 클라이언트 오류가 발생했습니다.

문제: Amazon EC2 Auto Scaling에서 암호화된 EBS 볼륨이 있는 인스턴스를 시작하려고 하지만 서비스 연결 역할은 이를 암호화하는 데 사용된 AWS KMS 고객 관리 키에 액세스할 수 없습니다. 자세한 내용은 암호화된 볼륨과 함께 사용하는 데 필요한 AWS KMS 키 정책 단원을 참조하십시오.

원인 1: 적절한 서비스 연결 역할에 고객 관리형 키를 사용할 수 있는 권한을 부여하는 키 정책이 필요합니다.

솔루션 1: 다음과 같이 서비스 연결 역할이 고객 관리형 키를 사용하도록 허용합니다.

1. 이 Auto Scaling 그룹에 어떤 서비스 연결 역할을 사용할지 결정합니다.
2. 고객 관리형 키의 키 정책을 업데이트하여 서비스 연결 역할이 고객 관리형 키를 사용하도록 허용합니다.
3. Auto Scaling 그룹이 서비스 연결 역할을 사용하도록 업데이트합니다.

서비스 연결 역할이 고객 관리형 키를 사용할 수 있도록 하는 주요 정책의 예는 예 1: 고객 관리형 키에 대한 액세스를 허용하는 키 정책 섹션을 참조하세요.

원인 2: 고객 관리 키와 Auto Scaling 그룹이 서로 다른 AWS 계정에 있는 경우, 고객 관리 키를 적절한 서비스 연결 역할에 사용할 권한을 부여하려면 고객 관리 키에 대한 교차 계정 액세스를 구성해야 합니다.

솔루션 2: 다음과 같이 외부 계정의 서비스 연결 역할이 로컬 계정의 고객 관리형 키를 사용할 수 있도록 허용합니다.

1. Auto Scaling 그룹 계정에서 고객 관리형 키에 액세스할 수 있도록 고객 관리형 키의 키 정책을 업데이트합니다.
2. Auto Scaling 그룹 계정에서 권한을 생성할 수 있는 IAM 사용자 또는 역할을 정의합니다.
3. 이 Auto Scaling 그룹에 어떤 서비스 연결 역할을 사용할지 결정합니다.
4. 피부여자 보안 주체로서의 적절한 서비스 연결 역할로 고객 관리형 키에 권한 부여를 생성합니다.
5. Auto Scaling 그룹이 서비스 연결 역할을 사용하도록 업데이트합니다.

자세한 설명은 예 2: 고객 관리형 키에 대한 교차 계정 액세스를 허용하는 키 정책 섹션 섹션을 참조하세요.

솔루션 3: Auto Scaling 그룹과 동일한 AWS 계정의 고객 관리형 키를 사용합니다.

1. 스냅샷을 복사하고 Auto Scaling 그룹과 동일한 계정에 속한 다른 고객 관리형 키로 다시 암호화합니다.
2. 서비스 연결 역할이 새 고객 관리형 키를 사용하도록 허용합니다. 솔루션 1의 단계를 참조하세요.

요청한 가용 영역에 현재 <인스턴스 유형> 용량이 부족합니다... EC2인스턴스 시작에 실패했습니다.

오류 메시지: 현재 요청한 가용 영역(<요청한 가용 영역>)에 충분한 <인스턴스 유형> 용량을 갖고 있지 않습니다. 시스템에서 추가 용량을 프로비전하기 위한 작업이 진행 중입니다. 요청에서 가용 영역을 지

정하지 않거나 <현재 이 인스턴스 유형을 지원하는 가용 영역 목록>을 선택하여 지금 <인스턴스 유형> 용량을 확보할 수 있습니다. EC2인스턴스 시작에 실패했습니다.

원인: 현재 요청된 인스턴스 유형과 가용 영역 조합은 지원되지 않습니다.

해결 방법: 문제를 해결하려면 다음을 시도해 보십시오.

- Amazon EC2 Auto Scaling이 활성화된 다른 가용 영역에서 이 인스턴스 유형의 용량을 찾을 때까지 몇 분 정도 기다리십시오.
- Auto Scaling 그룹을 추가 가용 영역으로 스케일 아웃합니다. 자세한 내용은 가용 영역 추가 단원을 참조하십시오.
- 한 가지 특정 인스턴스 유형에 의존하지 않도록 다양한 인스턴스 유형 세트를 사용하는 모범 사례를 따르세요. 자세한 설명은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.

요청된 예약에는 이 요청에 사용할 수 있는 호환 및 사용 가능한 용량이 충분하지 않습니다. EC2인스턴스 시작에 실패했습니다.

원인 1: `targeted` 온디맨드 용량 예약으로 시작할 수 있는 인스턴스 수 제한에 도달했습니다.

솔루션 1: `targeted` 온디맨드 용량 예약으로 시작할 수 있는 인스턴스 수를 늘리거나 용량 예약 그룹을 사용하여 예약 용량을 초과하는 모든 인스턴스가 일반 온디맨드 용량으로 시작되도록 합니다. 자세한 설명은 용량 예약을 통해 특정 가용 영역의 용량을 예약할 수 있습니다. 섹션을 참조하세요.

원인 2: 어떤 용량 블록으로 시작시킬 수 있는 인스턴스의 수가 한도에 도달했습니다.

용량 블록을 사용하면 원래 구매한 용량의 제약을 받습니다. 예상보다 많은 수의 실행이 발생하여 사용 가능한 용량을 모두 사용하면 실행이 실패하게 됩니다. 해지하는 인스턴스는 완전히 해지되기 전에 긴 정리 프로세스를 거칩니다. 이 시간 동안에는 재사용할 수 없습니다. 이로 인해 실행이 실패할 수도 있습니다. 자세한 내용은 사용 Capacity Blocks 기계 학습 워크로드용 단원을 참조하십시오.

해결 방법 2: 문제를 해결하려면 다음을 시도해 보십시오.

- 요청을 그대로 유지하세요. Capacity Block 인스턴스가 종료되는 경우 인스턴스 종료를 완료하고 용량을 다시 사용할 수 있을 때까지 몇 분 정도 기다려야 합니다. Amazon EC2 Auto Scaling은 용량이 확보될 때까지 계속해서 자동으로 시작을 요청합니다.
- 이 오류가 자주 발생하지 않도록 최대 워크로드를 수용할 수 있는 충분한 용량을 구매해야 합니다.

용량 블록 <reservation id> 예약이 아직 활성화되지 않았습니다. EC2인스턴스 시작에 실패했습니다.

원인: 지정된 용량 블록이 아직 활성화되지 않았습니다.

솔루션: 용량 블록에 대한 권장 접근 방식을 따르고 예약된 스케일링을 사용합니다. 이렇게 하면 예약이 활성 상태일 때만 Auto Scaling 그룹의 원하는 용량을 늘리고 예약이 해지되기 전에 용량을 줄일 수 있습니다.

요청과 일치하는 스팟 용량이 없습니다. EC2인스턴스 시작에 실패했습니다.

원인: 현재 스팟 인스턴스에 대한 요청을 이행하기에 충분한 여유 용량이 없습니다.

해결 방법: 문제를 해결하려면 다음을 시도해 보십시오.

- 몇 분 정도 기다리세요. 용량은 자주 변할 수 있습니다. Amazon EC2 Auto Scaling은 용량이 확보될 때까지 계속해서 자동으로 시작을 요청합니다.
- Auto Scaling 그룹을 추가 가용 영역으로 스케일 아웃합니다. 자세한 내용은 가용 영역 추가 단원을 참조하십시오.
- 한 가지 특정 인스턴스 유형에 의존하지 않도록 다양한 인스턴스 유형 세트를 사용하는 모범 사례를 따르세요. 자세한 설명은 여러 인스턴스 유형 및 구매 옵션이 포함된 Auto Scaling 그룹 섹션을 참조하세요.

<인스턴스 수>개의 인스턴스가 이미 실행 중입니다. EC2인스턴스 시작에 실패했습니다.

원인: 지역에서 시작할 수 있는 인스턴스 수가 한도에 도달했습니다. AWS 계정을 생성할 때 지역별로 실행할 수 있는 인스턴스 수에 대한 기본 한도가 설정됩니다.

해결 방법: 문제를 해결하려면 다음을 시도해 보세요.

- 현재 한도가 필요에 적합하지 않은 경우, 지역별로 할당량 증가를 요청할 수 있습니다. 자세한 내용은 Amazon EC2사용 설명서의 Amazon EC2 서비스 할당량을 참조하십시오.
- 인스턴스 수를 줄인 새 요청을 제출합니다(인스턴스 수는 이후 단계에서 늘릴 수 있음).

Amazon EC2 Auto Scaling 문제 해결: AMI 문제

이 페이지에서는 귀하와 관련된 문제AMIs, 잠재적 원인, 문제 해결을 위해 취할 수 있는 조치에 대한 정보를 제공합니다.

오류 메시지를 검색하려면 크기 조정 활동에서 오류 메시지 검색를 참조하세요.

관련 문제로 인해 EC2 인스턴스 시작에 실패할 경우 다음 오류 메시지 중 하나 이상이 표시될 수 AMI 있습니다.

AMI문제

- AMIID <귀하의 ID AMI > 이 (가) 존재하지 않습니다. EC2인스턴스 시작에 실패했습니다.
- AMI< AMI ID>는 보류 중이므로 실행할 수 없습니다. EC2인스턴스 시작에 실패했습니다.
- 디바이스 명칭 <device name>이(가) 잘못되었습니다. EC2인스턴스 시작에 실패했습니다.
- 지정된 인스턴스 유형의 'arm64' 아키텍처가 지정된 아키텍처 'x86_64'와 일치하지 않습니다... 인스턴스 시작에 실패했습니다. AMI EC2
- AMI'< AMI ID>'는 비활성화되어 실행할 수 없습니다. 인스턴스 EC2 시작에 실패했습니다.

> **Important**
>
> AWS 권한을 수정하여 다른 AWS 계정과 AMI 비공개로 공유할 수 있습니다. AMI 를 AMI 공유하지 않고 비공개로 설정하면 새 인스턴스를 시작할 때 권한 부여 오류가 발생할 수 있습니다. AMIsAMI비공개로 공유하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 특정 AWS 계정과 공유를 참조하십시오.

AMIID <귀하의 ID AMI > 이 (가) 존재하지 않습니다. EC2인스턴스 시작에 실패했습니다.

- 원인: 시작 템플릿 또는 시작 구성을 생성한 후 AMI 삭제되었을 수 있습니다.
- 해결 방법:
  1. 유효한 템플릿을 사용하여 새 시작 템플릿 또는 시작 구성을 AMI 생성하십시오.
  2. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

AMI< AMI ID>는 보류 중이므로 실행할 수 없습니다. EC2인스턴스 시작에 실패했습니다.

원인: 실행 중인 인스턴스의 스냅샷을 찍거나 다른 방법으로 인스턴스를 방금 생성했는데 아직 사용할 수 없을 수 있습니다. AMI

해결 방법: 사용할 수 있을 때까지 기다린 다음 시작 템플릿 또는 시작 구성을 생성해야 합니다. AMI

디바이스 명칭 <device name>이(가) 잘못되었습니다. EC2인스턴스 시작에 실패했습니다.

원인: EBS 볼륨을 EC2 인스턴스에 연결할 때 볼륨에 유효한 디바이스 이름을 제공해야 합니다. 선택한 디바이스는 이 디바이스 이름을 AMI 지원해야 합니다.

해결 방법:

1. 새 시작 템플릿이나 시작 구성을 만들고 해당 템플릿에 맞는 올바른 장치 이름을 지정하십시오AMI. 권장 명명 규칙은 의 가상화 유형에 따라 달라집니다. AMI 자세한 내용은 Amazon EC2 사용 설명서의 장치 이름을 참조하십시오.
2. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

지정된 인스턴스 유형의 'arm64' 아키텍처가 지정된 아키텍처 'x86_64'와 일치하지 않습니다... 인스턴스 시작에 실패했습니다. AMI EC2

원인 1: 시작 템플릿 또는 시작 구성에 사용된 인스턴스 유형과 의 아키텍처가 동일하지 않은 경우, Amazon EC2 Auto Scaling에서 호환되지 않는 인스턴스 구성을 사용하여 인스턴스를 시작하려고 하면 오류가 발생합니다. AMI

솔루션 1:

1. describe-images 명령을 AMI 사용하거나 Amazon EC2 콘솔에서 Amazon 머신 이미지 () AMIs 페이지의 세부 정보 창에서 아키텍처 값을 확인하여 아키텍처를 확인합니다.
2. describe-instance-types명령을 AMI 사용하거나 Amazon EC2 콘솔에서 Instance types 화면의 Architecture 열을 확인하여 아키텍처가 동일한 인스턴스 유형을 찾으십시오. 호환되는 인스턴스 유형을 선택하는 방법에 대한 자세한 내용은 Amazon EC2 User Guide의 인스턴스 유형 변경을 위한 호환성을 참조하십시오.

3. 기존 인스턴스와 동일한 아키텍처를 가진 인스턴스 유형을 사용하여 새 시작 템플릿 또는 시작 구성을 생성하십시오AMI.
4. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

원인 2: Amazon EC2 Auto Scaling이 Auto Scaling 그룹의 혼합 인스턴스 정책에 지정된 인스턴스 유형을 시작하려고 하지만, 인스턴스 유형이 시작 템플릿에 AMI 지정된 것과 동일한 아키텍처를 가지고 있지 않습니다.

솔루션 1: 혼합 인스턴스 정책에 아키텍처가 다른 인스턴스 유형을 포함시키지 마세요.

1. describe-images 명령을 AMI 사용하거나 Amazon EC2 콘솔에서 Amazon 머신 이미지 () AMIs 페이지의 세부 정보 창에서 아키텍처 값을 확인하여 아키텍처를 확인합니다.
2. describe-instance-types명령을 사용하거나 Amazon EC2 콘솔에서 Instance types 화면의 Architecture 열을 확인하여 혼합 인스턴스 정책에 포함하려는 각 인스턴스 유형의 아키텍처를 확인합니다. 호환되는 인스턴스 유형 선택에 대한 자세한 내용은 Amazon EC2 User Guide의 인스턴스 유형 변경을 위한 호환성을 참조하십시오.
3. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹에서 호환되지 않는 인스턴스 유형을 업데이트하거나 제거합니다.

해결 방법 2: 동일한 Auto Scaling 그룹에서 Arm (Graviton2) 및 x86_64 (Intel) 인스턴스를 모두 시작하려면 ARM과 호환되는 시작 AMI 템플릿과 Intel AMI x86 호환이 각각 지원되는 시작 템플릿을 사용하여 혼합 인스턴스 정책의 인스턴스 유형과 일치해야 합니다.

1. describe-images 명령을 사용하거나 Amazon AMI EC2 콘솔에서 Amazon Machine Images () AMIs 페이지의 세부 정보 창에서 아키텍처 값을 확인하여 기존 시작 템플릿의 아키텍처를 확인합니다.
2. 사용하려는 다른 아키텍처와 AMI 일치하는 템플릿을 사용하여 새 시작 템플릿을 생성하십시오.
3. Auto Scaling 그룹을 업데이트하여 기존 시작 템플릿을 재정의하고 update-auto-scaling-group명령을 사용하여 호환되는 각 인스턴스 유형에 대해 새 시작 템플릿을 지정합니다. 자세한 내용은 인스턴스 유형에 대해 서로 다른 시작 템플릿 사용 단원을 참조하십시오.

AMI'< AMI ID>'는 비활성화되어 실행할 수 없습니다. 인스턴스 EC2 시작에 실패했습니다.

원인: AMI 비활성화된 인스턴스에서 인스턴스를 시작하려고 합니다. 자세한 내용은 Amazon 사용 EC2 설명서의 AMI비활성화를 참조하십시오.

해결 방법:

1. 새 시작 템플릿 또는 시작 구성을 생성하고 비활성화되지 AMI 않은 항목을 지정하십시오.
2. update-auto-scaling-group명령을 사용하여 Auto Scaling 그룹을 새 시작 템플릿 또는 시작 구성으로 업데이트합니다.

Amazon EC2 Auto Scaling: 로드 밸런서 문제 해결

이 페이지에는 Auto Scaling 그룹과 연결된 로드 밸런서로 인해 발생한 문제, 잠재적인 원인, 문제 해결을 위해 취할 수 있는 조치에 대한 정보가 나와 있습니다.

오류 메시지를 검색하려면 크기 조정 활동에서 오류 메시지 검색를 참조하세요.

Auto Scaling 그룹과 연결된 로드 밸런서 관련 문제로 인해 EC2 인스턴스가 시작되지 않는 경우 다음 오류 메시지 중 하나 이상이 표시될 수 있습니다.

로드 밸런서 문제

- One or more target groups not found.(하나 이상의 대상 그룹을 찾을 수 없습니다.) 로드 밸런서 구성의 유효성 검사에 실패했습니다.
- Cannot find Load Balancer <your load balancer>.(로드 밸런서 <사용자의 로드 밸런서>를 찾을 수 없습니다.) 로드 밸런서 구성의 유효성 검사에 실패했습니다.
- <load balancer name>이름이 지정된 ACTIVE Load Balancer가 없습니다. 로드 밸런서 구성을 업데이트하는 데 실패했습니다.
- EC2<instance ID>인스턴스가 들어있지 않습니다. VPC 로드 밸런서 구성을 업데이트하는 데 실패했습니다.

> **Note**
>
> Reachability Analyzer를 사용하여 로드 밸런서를 통해 Auto Scaling 그룹의 인스턴스에 연결할 수 있는지 여부를 확인하여 연결 문제를 해결할 수 있습니다. Reachability Analyzer에서 자동으로 감지되는 다양한 네트워크 구성 오류 문제에 대해 알아보려면 Reachability Analyzer 사용

자 가이드의 Reachability Analyzer explanation codes(VPC Reachability Analyzer 설명 코드)를 참조하세요.

One or more target groups not found.(하나 이상의 대상 그룹을 찾을 수 없습니다.) 로드 밸런서 구성의 유효성 검사에 실패했습니다.

문제: Auto Scaling 그룹에서 인스턴스를 시작하면 Amazon EC2 Auto Scaling은 Auto Scaling 그룹과 연결된 Elastic Load Balancing 리소스가 존재하는지 검증하려고 합니다. 대상 그룹을 찾을 수 없는 경우, 조정 작업이 실패하고 `One or more target groups not found. Validating load balancer configuration failed.`(하나 이상의 대상 그룹을 찾을 수 없습니다. 로드 밸런서 구성을 검증하지 못했습니다.) 오류가 발생합니다.

원인 1: Auto Scaling 그룹에 연결된 대상 그룹이 삭제되었습니다.

솔루션 1: Amazon EC2 Auto Scaling 콘솔 또는 detach-load-balancer-target-groups 명령을 사용하여 대상 그룹 없이 새 Auto Scaling 그룹을 생성하거나 Auto Scaling 그룹에서 사용하지 않는 대상 그룹을 제거할 수 있습니다.

원인 2: 대상 그룹이 존재하지만 Auto Scaling 그룹을 생성할 ARN 때 대상 그룹을 지정하는 데 문제가 발생했습니다. 리소스가 올바른 순서로 생성되지 않습니다.

솔루션 2: 새 Auto Scaling 그룹을 생성하고 마지막에 대상 그룹을 지정합니다.

Cannot find Load Balancer <your load balancer>.(로드 밸런서 <사용자의 로드 밸런서>를 찾을 수 없습니다.) 로드 밸런서 구성의 유효성 검사에 실패했습니다.

문제: Auto Scaling 그룹에서 인스턴스를 시작하면 Amazon EC2 Auto Scaling은 Auto Scaling 그룹과 연결된 Elastic Load Balancing 리소스가 존재하는지 검증하려고 합니다. Classic Load Balancer를 찾을 수 없는 경우, 조정 작업이 실패하고 `Cannot find Load Balancer <your load balancer>. Validating load balancer configuration failed.`(로드 밸런서 <사용자의 로드 밸런서>를 찾을 수 없습니다. 로드 밸런서 구성을 검증하지 못했습니다.) 오류가 발생합니다.

원인 1: Classic Load Balancer가 삭제되었습니다.

솔루션 1: Amazon Auto Scaling 콘솔 또는 명령을 사용하여 로드 밸런서 없이 새 Auto Scaling 그룹을 생성하거나 Auto Scaling 그룹에서 사용하지 않는 로드 밸런서를 제거할 수 있습니다. EC2 detach-load-balancers

원인 2: Classic Load Balancer가 존재하지만 Auto Scaling 그룹을 생성할 때 로드 밸런서 이름을 지정하는 데 문제가 있었습니다. 리소스가 올바른 순서로 생성되지 않습니다.

솔루션 2: 새 Auto Scaling 그룹을 생성하여 마지막에 로드 밸런서 이름을 지정합니다.

<load balancer name>이름이 지정된 ACTIVE Load Balancer가 없습니다. 로드 밸런서 구성을 업데이트하는 데 실패했습니다.

원인: 지정된 로드 밸런서가 삭제되었을 수 있습니다.

솔루션: 새 로드 밸런서를 생성한 다음 새 Auto Scaling 그룹을 생성하거나 로드 밸런서 없이 새 Auto Scaling 그룹을 생성할 수 있습니다.

EC2<instance ID>인스턴스가 들어있지 않습니다. VPC 로드 밸런서 구성을 업데이트하는 데 실패했습니다.

원인: 지정된 인스턴스가 에 없습니다VPC.

솔루션: 인스턴스에 연결된 로드 밸런서를 삭제하거나 새 Auto Scaling 그룹을 생성할 수 있습니다.

Amazon EC2 Auto Scaling: 시작 템플릿 문제 해결

다음 정보를 사용하여 Auto Scaling 그룹에 대한 시작 템플릿을 지정하고자 할 때 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

인스턴스를 시작할 수 없음

이미 지정된 시작 템플릿으로 인스턴스를 시작할 수 없는 경우, 일반적인 문제 해결을 위해 Amazon EC2 Auto Scaling 문제 해결: EC2 인스턴스 시작 실패를 확인하세요.

완전한 형태의 유효한 시작 템플릿을 사용해야 합니다(잘못된 값).

문제: Auto Scaling 그룹에 대한 시작 템플릿을 지정하려고 하면 `You must use a valid fully-formed launch template` 오류를 받습니다. 시작 템플릿을 사용하는 Auto Scaling 그룹이 생성 또는 업데이트될 때만 시작 템플릿의 값이 검증되기 때문에 이 오류가 발생할 수 있습니다.

원인 1: `You must use a valid fully-formed launch template` 오류가 발생하는 경우 Amazon EC2 Auto Scaling에서 시작 템플릿에 대해 유효하지 않은 것으로 간주하는 문제가 발생한 것입니다. 이 오류는 여러 가지 원인이 있을 수 있는 일반 오류입니다.

솔루션 1: 문제 해결을 위해 다음 단계를 시도하세요.

1. 자세한 내용을 확인하려면 오류 메시지의 두 번째 부분에 주의해야 합니다. `You must use a valid fully-formed launch template` 오류 이후 해결해야 할 문제를 식별하는 보다 구체적인 오류 메시지를 참조하세요.
2. 원인을 찾을 수 없는 경우, run-instances 명령을 사용하여 시작 템플릿을 테스트하세요. 다음 예와 같이 `--dry-run` 옵션을 사용합니다. 이를 통해 문제를 재현하고 원인에 대한 인사이트를 제공할 수 있습니다.

```
aws ec2 run-instances --launch-template LaunchTemplateName=my-template,Version='1' --
dry-run
```

3. 값이 유효하지 않은 경우, 지정된 리소스가 존재하며 올바른지 확인합니다. 예를 들어 Amazon EC2 키 페어를 지정하는 경우 Auto Scaling 그룹을 생성 또는 업데이트하려는 계정 및 지역에 리소스가 있어야 합니다.
4. 예상한 정보가 누락된 경우, 설정을 확인하고 필요에 따라 시작 템플릿을 조정합니다.
5. 변경한 후 `--dry-run` 옵션을 사용해 run-instances 명령을 다시 실행하여 시작 템플릿에 유효한 값이 사용되는지 확인할 수 있습니다.

자세한 설명은 Auto Scaling 그룹에 대한 시작 템플릿 생성 섹션을 참조하세요.

시작 템플릿을 사용할 권한이 없음(권한 부족)

문제: Auto Scaling 그룹에 대한 시작 템플릿을 지정하려고 하면 `You are not authorized to use launch template` 오류를 받습니다.

원인 1: 시작 템플릿을 사용하려고 하는데 사용 중인 IAM 자격 증명에 충분한 권한이 없는 경우 시작 템플릿을 사용할 권한이 없다는 오류 메시지가 나타납니다.

해결 방법 1: 문제를 해결하려면 다음을 시도해 보십시오.

- 요청하는 데 사용하는 IAM 자격 증명에 작업을 포함하여 필요한 EC2 API 작업을 호출할 수 있는 권한이 있는지 확인하십시오. `ec2:RunInstances` 시작 템플릿에 태그를 지정한 경우, `ec2:CreateTags` 작업을 사용할 수 있는 권한도 있어야 합니다.
- 또는 요청하는 데 사용하는 IAM 자격 증명에 `AmazonEC2FullAccess` 정책이 할당되었는지 확인하십시오. 이 AWS 관리형 정책은 Amazon EC2 Auto Scaling 및 Elastic Load Balancing을 비롯한 모든 Amazon EC2 리소스 및 관련 서비스에 대한 전체 액세스 권한을 부여합니다. CloudWatch

예제 IAM 정책을 포함하여 시작 템플릿을 사용하는 데 필요한 권한에 대한 자세한 내용은 Amazon EC2 User Guide의 IAM 권한으로 시작 템플릿에 대한 액세스 제어를 참조하십시오. 다른 예제 IAM 정책은 을 참조하십시오Auto Scaling 그룹에서 Amazon EC2 시작 템플릿 사용을 제어합니다..

원인 2: 인스턴스 프로필을 지정하는 시작 템플릿을 사용하려는 경우 인스턴스 프로필과 연결된 IAM 역할을 전달할 IAM 권한이 있어야 합니다.

해결 방법 2: 요청하는 데 사용하는 IAM 자격 증명에 Amazon EC2 Auto Scaling 서비스에 지정된 역할을 전달할 수 있는 올바른 `iam:PassRole` 권한이 있는지 확인하십시오. 자세한 내용 및 예제 IAM 정책은 을 참조하십시오Amazon EC2 인스턴스에서 실행되는 애플리케이션에 대한 IAM 역할. 인스턴스 프로필과 관련된 추가 문제 해결 주제는 Amazon 문제 해결 EC2 및 IAM IAM 사용 설명서를 참조하십시오.

원인 3: 다른 AWS 계정시작 AMI 템플릿에서 a를 지정하는 시작 템플릿을 사용하려고 하는데 해당 AMI 템플릿이 비공개이고 사용 중인 템플릿과 공유되지 않는 경우, 시작 템플릿을 사용할 권한이 없다는 오류 메시지가 나타납니다. AWS 계정

해결 방법 3: 사용 중인 계정이 해당 권한에 AMI 포함되어 있는지 확인하십시오. 자세한 내용은 Amazon EC2 사용 설명서의 특정 AMI AWS 계정사용자와의 공유를 참조하십시오.

관련 정보

다음의 관련 리소스는 이 서비스를 이용할 때 도움이 될 수 있습니다.

Resource	설명
Amazon EC2 Auto Scaling API 참조 문서	각 API 작업을 위한 설명서는 shows 요청 파라미터와 XML 응답을 보여주며 언어별 SDK 참조 항목에 대한 링크를 제공합니다.
AWS CLI 명령 참조 문서에서의 자동 규모 조정	Auto Scaling 그룹 작업에 사용할 수 있는 AWS CLI 명령에 대한 설명입니다.
AWS Tools for PowerShell cmdlet 참조	AWS 도구를 PowerShell 사용하면 PowerShell 명령줄에서 AWS 리소스에 대한 작업을 스크립팅할 수 있습니다.
AWS CloudFormation을 이용한 Auto Scaling 그룹 생성	AWS::AutoScaling::AutoScalingGroup AWS::AutoScaling::AutoScaling 리소스를 사용하면 수동 조치 없이 Auto Scaling 그룹을 구축, 모델링 및 관리할 수 있습니다.
AWS 일반 참조에서의 Amazon EC2 Auto Scaling 엔드포인트 및 할당량	Amazon EC2 Auto Scaling 지역 및 엔드포인트에 대한 정보
제품 페이지	Amazon EC2 Auto Scaling에 대한 정보를 제공하는 기본 웹 페이지입니다.
AWS re:POST	AWS 기술적인 질문에 대해 크라우드 소싱되고 전문가가 검토한 답변을 제공하는 관리형 질문 및 답변 (Q & A) 서비스입니다.
Amazon EC2 사용 설명서에서 AMI를 생성합니다.	맞춤 인스턴스에서 Amazon Machine Image(AMI)를 만드는 방법에 대해 알아봅시다.
Amazon EC2 사용 설명서에서 Linux 인스턴스에 연결하는 방법	실행하는 Linux 인스턴스에 연결하는 방법을 알아보세요.

Resource	설명
Amazon EC2 사용 설명서에서 Windows 인스턴스에 연결하는 방법	출범시키는 Windows 인스턴스에 연결하는 방법을 알아보세요.
Amazon 사용 CloudWatch 설명서에서 예상 AWS 요금을 모니터링하기 위한 청구 경보 생성	를 사용하여 예상 요금을 모니터링하는 방법을 알아보십시오 CloudWatch.
Auto Scaling 애플리케이션 사용 설명서	Amazon EC2 이외의 Amazon Web Services를 위한 확장 가능한 리소스를 위한 Auto Scaling을 구성하는 방법을 알아보십시오.

자세히 AWS알아보는 데 도움이 되는 다음과 같은 일반 리소스를 이용할 수 있습니다.

- 수업 및 워크숍 — 역할 기반 및 전문 과정에 대한 링크와 함께 AWS 기술을 연마하고 실제 경험을 쌓는 데 도움이 되는 자습형 실습을 제공합니다.
- AWS 개발자 센터 — 튜토리얼을 탐색하고, 도구를 다운로드하고, 개발자 이벤트에 대해 알아보십시오. AWS
- AWS 개발자 도구 — 애플리케이션 개발 및 관리를 위한 개발자 도구, SDK, IDE 툴킷 및 명령줄 도구에 대한 링크입니다. AWS
- 시작하기 리소스 센터 — 애플리케이션을 설치하고, AWS 커뮤니티에 가입하고 AWS 계정, 첫 번째 애플리케이션을 시작하는 방법을 알아보세요.
- 실습 튜토리얼 — 튜토리얼을 따라 step-by-step 첫 애플리케이션을 시작하세요. AWS
- AWS 백서 — 아키텍처, 보안, 경제 등의 주제를 다루고 솔루션스 아키텍트 또는 기타 기술 전문가가 작성한 포괄적인 기술 AWS 백서 목록에 대한 링크입니다. AWS
- AWS Support 센터 — 사례 작성 및 관리를 위한 허브. AWS Support 포럼, 기술 FAQ, 서비스 상태 등과 같은 기타 유용한 리소스에 대한 링크도 포함되어 있습니다. AWS Trusted Advisor
- AWS Support— 클라우드에서 애플리케이션을 구축하고 실행하는 데 도움이 되는 one-on-one 신속한 지원 채널에 대한 AWS Support정보를 제공하는 기본 웹 페이지입니다.
- Contact Us(문의처) - AWS 결제, 계정, 이벤트, 침해 및 기타 문제에 대해 문의할 수 있는 중앙 연락 창구입니다.
- AWS 사이트 약관 — 당사의 저작권 및 상표, 사용자 계정, 라이선스, 사이트 액세스, 기타 주제에 대한 자세한 정보.

문서 이력

다음 표에서는 2018년 7월 이후의 Amazon EC2 Auto Scaling 설명서에 대한 중요 추가 사항을 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드에 가입하면 됩니다.

변경 사항	설명	날짜
보안 IAM 업데이트	AutoScalingServiceRolePolicy관리형 정책은 이제 Amazon EC2 (`ec2:GetSecurityGroupsForVpc` 및`ec2:GetInstanceTypesFromInstanceRequirements`) 에 추가 권한을 부여합니다.	2024년 2월 29일
웜 풀 최대 절전 모드가 추가로 지원됩니다. AWS 리전	이제 2개의 추가 지역, 즉 AWS GovCloud (미국 동부) 및 (미국 서부) 의 웜 풀에서 인스턴스를 하이버네이트할 수 있습니다. AWS GovCloud 웜 풀에 대한 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 Amazon EC2 Auto Scaling을 위한 웜 풀을 참조하세요.	2024년 2월 26일
웜풀 하이버네이션은 추가로 지원됩니다. AWS 리전	이제 유럽 (취리히) 과 중동 (UAE) 이라는 두 개의 추가 지역에서 웜 풀의 인스턴스를 하이버네이트할 수 있습니다. 웜 풀에 대한 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 Amazon EC2 Auto Scaling을 위한 웜 풀을 참조하세요.	2024년 2월 21일

교차 계정 매개변수 사용 지원	이제 Amazon EC2 Auto Scaling에서 다른 AWS Systems Manager AWS 계정 파라미터와 공유한 파라미터를 사용할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 시작 템플릿에서 AMI ID 대신 AWS Systems Manager 파라미터 사용을 참조하십시오.	2024년 2월 21일
새로운 스팟 가격 보호 옵션	이제 속성 기반 인스턴스 유형 선택을 사용할 때 스팟 인스턴스의 가격 보호 임계값을 온디맨드 가격의 백분율로 정의할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 가격 보호를 참조하십시오.	2024년 1월 29일
인스턴스 정비 정책	이제 인스턴스 정비 정책을 사용하여 인스턴스 새로 고침 등 인스턴스를 교체하는 이벤트가 발생하는 동안 기존 인스턴스가 해지되기 전 또는 후에 인스턴스를 출범할지 여부를 정의할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 정비 정책을 참조하십시오.	2023년 11월 15일

ML용 용량 블록	이제 출범 템플릿을 생성할 때 용량 블록 예약 ID를 지정하여 인스턴스를 용량 블록으로 시작할 수 있습니다. 용량 블록을 사용하면 미래 날짜의 GPU 인스턴스를 예약하여 단기간의 기계 학습(ML) 워크로드를 지원할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 기계 학습 워크로드에 용량 블록 사용을 참조하십시오.	2023년 10월 31일
새 인스턴스 새로 고침 기능	이제 인스턴스 새로 고침을 구성하여 상태를 실패로 설정하고, 지정된 CloudWatch 경보가 해당 상태로 전환된 것을 감지하면 선택적으로 롤백할 수 있습니다. `ALARM` 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 롤백으로 변경 취소하기를 참조하세요.	2023년 7월 31일
설명서 변경 사항	온디맨드 인스턴스를 용량 예약으로 시작하는 방법에 대한 새로운 주제가 설명서에 추가되었습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 온디맨드 용량 예약을 통한 용량 예약을 참조하세요.	2023년 7월 28일

설명서 변경 사항	시작 구성에서 시작 템플릿으로 AWS CloudFormation 스택을 마이그레이션하는 방법에 대한 새 주제가 가이드에 추가되었습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 출범 구성의 AWS CloudFormation 스택을 출범 템플릿으로 마이그레이션하기를 참조하세요.	2023년 4월 18일

새로운 API 작업 지원	이번 릴리스에서는 `AttachTrafficSources` , `DetachTrafficSources` , `DescribeTrafficSources` 의 세 가지 새로운 API 연산이 추가되었습니다. 또한 `DescribeAutoScalingGroups` 작업의 결과에 새로운 `TrafficSources` 필드가 추가되었습니다. `DescribeScalingActivities` 작업 결과에 새로운 활동 상태인 `WaitingForConnectionDraining` 이(가) 추가되었습니다. Amazon EC2 Auto Scaling은 `CreateAutoScalingGroup` , `UpdateAutoScalingGroup` , `DescribeAutoScalingGroups` 작업에서 `HealthCheckType` 필드에 대해 새로운 값인 `VPC_LATTICE` 을(를) 지원합니다. 자세한 설명은 Amazon EC2 Auto Scaling API Reference(Amazon EC2 Auto Scaling API 레퍼런스)를 참조하세요.	2023년 3월 31일

Amazon VPC Lattice 지원	이 릴리스는 Amazon EC2 Auto Scaling을 위한 VPC Lattice의 정식 버전입니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 VPC Lattice 대상 그룹과 함께 오토 스케일링으로 트래픽 라우팅을 참조하세요.	2023년 3월 31일
설명서 변경 사항	Elastic Load Balancing 작업에 대한 AWS CLI 예제가 있는 섹션에 이제 신규 및 업데이트된 예제가 포함되어 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 AWS Command Line Interface (AWS CLI) 을 사용하여 Elastic Load Balancing을 사용하는 예를 참조하십시오.	2023년 3월 31일
추가 예측 스케일링 지원 AWS 리전	이제 중동 (UAE) 및 AWS GovCloud (미국 동부) 지역에서 예측 규모 조정 정책을 만들 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 예측 스케일링을 참조하세요.	2023년 3월 16일

새 인스턴스 새로 고침 기능	이제 대기 중인 인스턴스를 해지하거나 무시하도록 선택하고 인스턴스가 교체 가능해질 때까지 기다리지 않고 스케일 인 방비된 인스턴스를 교체하거나 무시하도록 선택할 수 있습니다. 실패한 인스턴스 새로 고침에서 변경 사항을 롤백할 수도 있습니다. 이 업데이트의 일환으로 인스턴스 새로 고침 롤백, 인스턴스 새로 고침 취소 및 인스턴스 새로 고침의 구성 가능한 파라미터 기본값 이해에 대한 주제를 포함하도록 설명서가 스케일 아웃되었습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 새로 고침을 기준으로 Auto Scaling 인스턴스 교체를 참조하세요.	2023년 2월 10일
AMI ID의 AWS Systems Manager 파라미터 사용 지원	이제 출범 템플릿에 AMI ID 대신 Systems Manager 파라미터를 사용할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 출범 템플릿에 AMI ID 대신 AWS Systems Manager 파라미터 사용을 참조하세요.	2023년 1월 19일

예측적 조정 권장 사항	이제 Amazon EC2 Auto Scaling 콘솔에서 올바른 예측적 조정 정책을 평가하고 선택하기 위한 권장 사항을 얻을 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 예측적 조정 평가를 참조하세요.	2023년 1월 18일
예측 스케일링 예측	예측 스케일링으로 생성된 예측은 이제 매일이 아닌 6시간마다 업데이트됩니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 예측 스케일링을 참조하세요.	2023년 1월 6일
CloudWatch 미터법 수학 지원	이제 대상 추적 조정 정책을 생성할 때 지표 수학을 사용할 수 있습니다. 메트릭 수학을 사용하면 여러 CloudWatch 메트릭을 쿼리하고 수학 식을 사용하여 이러한 메트릭을 기반으로 새 시계열을 만들 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Create a target tracking scaling policy for Amazon EC2 Auto Scaling using metric math(지표 수학을 사용하여 Amazon EC2 Auto Scaling에서 대상 추적 조정 정책 생성)를 참조하세요.	2022년 12월 8일

IAM 서비스 연결 역할 권한 업데이트	`AutoScalingServiceRolePolicy` 정책은 이제 Amazon EC2 Auto Scaling 에 추가 권한을 부여합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling의AWS 관리형 정책을 참조하세요.	2022년 12월 6일
새로운 스팟 할당 전략	이제 가격 및 용량이 최적화된 할당 전략을 사용하여 중단 가능성이 가장 낮고 가격이 가장 낮은 스팟 풀에서 스팟 인스턴스를 요청할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 할당 전략을 참조하세요.	2022년 11월 10일
아시아 태평양(자카르타) 지역의 예측 조정 지원	이제 아시아 태평양(자카르타) 지역에서 예측 조정 정책을 생성할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 예측 스케일링을 참조하세요.	2022년 10월 13일
콘솔에서 예측 조정을 위한 맞춤 지표 지원	이제 Amazon EC2 Auto Scaling 콘솔에서 예측 조정 정책을 생성할 때 맞춤 지표를 사용할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 예측 스케일링을 참조하세요.	2022년 10월 13일

CloudWatch 예측 척도 지표에 대한 모니터링	이제 를 사용하여 예측 규모 조정을 위한 모니터링 데이터에 액세스할 수 있습니다. CloudWatch 이렇게 하면 지표 수식을 사용하여 예측 데이터의 정확도를 표시하는 새 시계열을 만들 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 CloudWatch 설명서에서 예측 규모 조정 지표 모니터링을 참조하십시오.	2022년 7월 7일
아시아 태평양(오사카) 지역의 예측 조정 지원	이제 아시아 태평양(오사카) 지역에서 예측 조정 정책을 생성할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 예측 스케일링을 참조하세요.	2022년 7월 6일
추가 지역에서 웜 풀 최대 절전 모드 지원	이제 아프리카(케이프타운), 아시아 태평양(자카르타), 아시아 태평양(오사카), 유럽(밀라노) 등 4개의 추가 지역에서 웜 풀의 인스턴스를 최대 절전 모드로 전환할 수 있습니다. 웜 풀에 대한 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 Amazon EC2 Auto Scaling을 위한 웜 풀을 참조하세요.	2022년 7월 5일

건전성 체크 업데이트	건전성 체크를 수행할 때 이제 Amazon EC2 Auto Scaling을 사용하여 일시적인 문제 또는 잘못 구성된 건전성 체크로 인해 발생할 수 있는 가동 중지 시간을 최소화할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling의 가동 중지 시간 최소화 방법을 참조하세요.	2022년 5월 21일
기본 인스턴스 워밍업	이제 Auto Scaling 그룹의 모든 웜업 및 쿨다운 설정을 통합하고 기본 인스턴스 워밍업을 활성화하여 지속적으로 확장되는 조정 정책의 성능을 최적화할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Auto Scaling 그룹의 기본 인스턴스 워밍업 설정을 참조하세요.	2022년 4월 19일
설명서 변경 사항	다른 AWS 서비스와의 통합에 관한 새로운 장이 가이드에 추가되었습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling와 통합된AWS 서비스를 참조하세요.	2022년 3월 29일

IAM 서비스 연결 역할 권한 업데이트	`AutoScalingServiceRolePolicy` 정책은 이제 Amazon EC2 Auto Scaling에 추가 읽기 권한을 부여합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling의AWS 관리형 정책을 참조하세요.	2022년 3월 28일
인스턴스 메타데이터에서 대상 라이프사이클 상태 제공	인스턴스 메타데이터에서 Auto Scaling 인스턴스의 대상 라이프사이클 상태를 검색할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 메타데이터를 통해 대상 라이프사이클 상태 검색을 참조하세요.	2022년 3월 24일
새로운 웜 풀 기능 지원	이제 웜 풀에서 인스턴스를 최대 절전 모드로 전환하여 메모리 콘텐츠(RAM)을 삭제하지 않고 인스턴스를 중지할 수 있습니다. 이제 나중에 필요한 인스턴스 용량을 항상 해지하지 않고 대신 축소 시 인스턴스를 웜 풀로 반환할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling을 위한 웜 풀을 참조하세요.	2022년 2월 24일

설명서 변경 사항	Amazon EC2 Auto Scaling 콘솔은 매칭 건너뛰기를 사용하고 원하는 구성을 지정한 상태로 인스턴스 새로 고침을 시작하도록 도와주는 추가 옵션으로 업데이트되었습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 새로 고침 시작 또는 취소(콘솔)를 참조하세요.	2022년 2월 3일
예측적 조정 정책에 대한 사용자 정의 지표	이제 예측적 조정 정책을 생성할 때 사용자 정의 지표를 사용할지 여부를 선택할 수 있습니다. 지표 수학을 사용하여 정책에 포함하는 지표를 추가로 맞춤할 수도 있습니다. 자세한 설명은 사용자 정의 지표를 사용하는 고급 예측 스케일링 정책 구성을 참조하세요.	2021년 11월 24일
새로운 온디맨드 할당 전략	이제 혼합 인스턴스 정책을 사용하는 Auto Scaling 그룹을 생성할 때 가격(최저가의 인스턴스 타입 먼저)을 기준으로 온디맨드 인스턴스를 출범할지 여부를 선택할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 할당 전략을 참조하세요.	2021년 10월 27일

속성 기반 인스턴스 타입 선택	Amazon EC2 Auto Scaling은 속성 기반 인스턴스 타입 선택에 대한 지원을 추가합니다. 인스턴스 타입을 수동으로 선택하는 대신 인스턴스 요건을 vCPU, 메모리 및 스토리지와 같은 속성 집합으로 표현할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 속성 기반 인스턴스 타입 선택을 사용하여 Auto Scaling 그룹 생성을 참조하세요.	2021년 10월 27일
태그로 그룹 필터링 지원	이제 `describe-auto-scaling-groups` 명령을 사용하여 Auto Scaling 그룹에 대한 정보를 검색할 때 태그 필터를 사용하여 Auto Scaling 그룹을 필터링할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 태그를 사용하여 Auto Scaling 그룹 필터링을 참조하세요.	2021년 10월 14일
설명서 변경 사항	Amazon EC2 Auto Scaling 콘솔이 업데이트되어 사용자 지정 종료 정책을 생성할 수 있습니다. AWS Lambda이에 따라 콘솔 문서가 수정되었습니다. 자세한 설명은 다른 해지 정책 사용(콘솔)을 참조하세요.	2021년 10월 14일

출범 템플릿에 출범 구성 복사 지원	이제 Amazon EC2 Auto Scaling 콘솔에서 한 AWS 지역의 모든 시작 구성을 새 시작 템플릿에 복사할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 출범 템플릿에 출범 구성 복사를 참조하세요.	2021년 8월 9일
인스턴스 새로 고침 기능 스케일 아웃	이제 `start-instance-refresh` 명령에 원하는 구성을 추가하여 인스턴스를 교체할 때 출범 템플릿의 새 버전과 같은 업데이트를 포함할 수 있습니다. 일치하는 경우, 건너뛰기를 활성화하여 원하는 구성이 이미 있는 인스턴스의 교체를 건너뛸 수도 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 새로 고침을 기준으로 Auto Scaling 인스턴스 교체를 참조하세요.	2021년 8월 5일
맞춤 해지 정책 지원	이제 를 사용하여 사용자 지정 종료 정책을 생성할 수 있습니다. AWS Lambda자세한 설명은 Lambda를 사용하여 맞춤 해지 정책 생성을 참조하세요. 이에 따라 해지 정책 지정에 관한 설명서가 업데이트되었습니다.	2021년 7월 29일

설명서 변경 사항	Amazon EC2 Auto Scaling 콘솔은 지정된 표준 시간대를 사용하여 예약된 작업을 생성하는 데 유용한 추가 기능으로 업데이트 및 강화되었습니다. 이에 따라 예약 조정에 관한 문서가 개정되었습니다.	2021년 6월 3일
출범 구성의 gp3 볼륨	이제 출범 구성에 대한 블록 디바이스 매핑에서 gp3 볼륨을 지정할 수 있습니다.	2021년 6월 2일
예측 조정 지원	이제 예측 조정을 사용하여 조정 정책에 따라 Amazon EC2 Auto Scaling 그룹을 사전 대응적으로 크기 조정할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 예측 스케일링을 참조하세요. 이번 업데이트를 통해 이제 AutoScalingServiceRolePolicy관리형 정책에 `cloudwatch:GetMetricData` API 작업을 호출할 수 있는 권한이 포함됩니다.	2021년 5월 19일
설명서 변경 사항	이제 에서 라이프사이클 후크의 예제 템플릿에 액세스할 수 GitHub 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 라이프사이클 후크를 참조하세요.	2021년 4월 9일

웜 풀 지원	이제 Auto Scaling 그룹에 웜 풀을 추가하여 처음 부팅 시간이 긴 애플리케이션의 성능(콜드 스타트 최소화)과 비용(과도한 프로비저닝 중지)을 균형 있게 조정할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling을 위한 웜 풀을 참조하세요.	2021년 4월 8일
체크포인트 지원	이제 인스턴스 새로 고침에 체크포인트를 추가하여 단계별로 인스턴스를 교체하고 특정 지점에서 인스턴스에 대한 확인을 수행할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 새로 고침에 체크포인트 추가를 참조하세요.	2021년 3월 18일
설명서 변경 사항	Amazon EC2 Auto Scaling 이벤트 및 라이프사이클 후크와 EventBridge 함께 사용하기 위한 설명서가 개선되었습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon EC2 Auto Scaling 사용 설명서 EventBridge 및 자습서: Lambda 함수를 호출하는 수명 주기 후크 구성을 참조하십시오.	2021년 3월 18일

현지 표준 시간대 지원	이제 put-scheduled-update-group-action 명령에 `--time-zone` 옵션을 추가하여 반복되는 예약된 작업을 현지 표준 시간대로 생성할 수 있습니다. 사용 중인 표준 시간대가 DST(일광 절약 시간)를 따르는 경우, 반복 작업은 일광 절약 시간에 맞춰 자동으로 조정됩니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 예약 조정을 참조하세요.	2021년 3월 9일
혼합 인스턴스 정책을 위한 기능 스케일 아웃	이제 혼합 인스턴스 정책을 사용할 때 스팟 용량에 맞춰 인스턴스 타입의 우선순위를 지정할 수 있습니다. Amazon EC2 Auto Scaling에서는 최선을 다해 인스턴스 타입 우선순위를 이행하려고 하지만 먼저 용량을 최적화합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 여러 인스턴스 타입 및 구매 옵션이 포함된 Auto Scaling 그룹을 참조하세요.	2021년 3월 8일

삭제된 그룹에 대한 크기 조정 활동	이제 describe-scaling-activities 명령에 `--include-deleted-groups` 옵션을 추가하여 삭제된 Auto Scaling 그룹에 대한 크기 조정 활동을 볼 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 문제 해결을 참조하세요.	2021년 2월 23일
콘솔 개선 사항	이제 Amazon EC2 Auto Scaling 콘솔에서 Application Load Balancer 또는 Network Load Balancer를 생성하고 연결할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 새 Application Load Balancer 또는 Network Load Balancer 생성 및 연결(콘솔)을 참조하세요.	2020년 11월 24일
여러 네트워크 인터페이스	이제 여러 네트워크 인터페이스를 지정하는 Auto Scaling 그룹에 대한 출범 템플릿을 구성할 수 있습니다. 자세한 설명은 VPC의 네트워크 인터페이스를 참조하세요.	2020년 11월 23일
여러 출범 템플릿	이제 Auto Scaling 그룹에서 출범 템플릿을 여러 개 사용할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 인스턴스 타입에 대한 다른 출범 템플릿 지정을 참조하세요.	2020년 11월 19일

Gateway Load Balancer	Amazon EC2 Auto Scaling에서 시작한 어플라이언스 인스턴스가 로드 밸런서에서 자동으로 등록 및 등록 취소되도록 Gateway Load Balancer를 Auto Scaling 그룹에 연결하는 방법을 보여주는 업데이트된 설명서입니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Elastic Load Balancing 타입 및 로드 밸런서를 Auto Scaling 그룹에 연결을 참조하세요.	2020년 11월 10일
최대 인스턴스 수명	이제 최대 인스턴스 수명을 하루(86,400초)로 줄일 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 최대 인스턴스 수명을 기준으로 Auto Scaling 인스턴스 교체를 참조하세요.	2020년 11월 9일
용량 재조정	Amazon EC2에서 재조정 권장 사항이 생성될 때 교체 스팟 인스턴스를 출범하도록 Auto Scaling 그룹을 구성할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 용량 재조정을 참조하세요.	2020년 11월 4일

인스턴스 메타데이터 서비스 버전 2	출범 구성을 사용할 때 인스턴스 메타데이터를 요청하기 위한 세션 지향 방법인 인스턴스 메타데이터 서비스 버전 2를 사용해야 할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 메타데이터 옵션 구성을 참조하세요.	2020년 7월 28일
설명서 변경 사항	Amazon EC2 Auto Scaling 사용자 가이드의 축소 중 해지할 Auto Scaling 인스턴스 제어, Auto Scaling 인스턴스 및 그룹 모니터링, 출범 템플릿 및 출범 구성 섹션에서 다양한 부분이 개선되었습니다.	2020년 7월 28일
인스턴스 새로 고침	구성을 변경할 때 인스턴스 새로 고침을 시작하여 Auto Scaling 그룹의 모든 인스턴스를 업데이트합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 새로 고침을 기준으로 Auto Scaling 인스턴스 교체를 참조하세요.	2020년 6월 16일

설명서 변경 사항	Amazon EC2 Auto Scaling 사용자 가이드의 최대 인스턴스 수명을 기준으로 Auto Scaling 인스턴스 교체, 여러 인스턴스 타입 및 구매 옵션이 포함된 Auto Scaling 그룹, Amazon SQS 기반 조정 및 Auto Scaling 그룹 및 인스턴스에 태그 지정 섹션에서 다양한 부분이 개선되었습니다.	2020년 5월 6일
설명서 변경 사항	IAM 문서의 다양한 부분이 개선되었습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 출범 템플릿 지원 및 Amazon EC2 Auto Scaling 자격 증명 기반 정책 예를 참조하세요.	2020년 3월 4일
조정 정책 비활성화	이제 조정 정책을 비활성화했다가 다시 활성화할 수 있습니다. 이 기능을 사용하면 나중에 정책을 다시 활성화할 수 있도록 구성 세부 정보를 유지하면서 조정 정책을 일시적으로 비활성화할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드에서 Auto Scaling 그룹에 대한 조정 정책 비활성화를 참조하세요.	2020년 2월 18일

알림 기능 추가	이제 Amazon EC2 Auto Scaling은 보안 그룹 또는 시작 템플릿이 없어서 Auto Scaling 그룹을 확장할 수 없을 AWS Health Dashboard 때 이벤트를 사용자에게 보냅니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling에 대한AWS Health Dashboard 알림을 참조하세요.	2020년 2월 12일
설명서 변경 사항	Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling이 IAM과 작동하는 방식,Amazon EC2 Auto Scaling 자격 증명 기반 정책 예, 암호화된 볼륨에서 사용할 경우, 필요한 CMK 키 정책 및 Auto Scaling 인스턴스 및 그룹 모니터링 섹션에서 다양한 부분이 개선 및 수정되었습니다.	2020년 2월 10일
설명서 변경 사항	인스턴스 가중치를 사용하는 Auto Scaling 그룹에 대한 문서가 개선되었습니다. “용량 유닛”을 사용하여 원하는 용량을 측정할 때 조정 정책을 사용하는 방법을 알아봅니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 조정 정책 작동 방식 및 조정 조절 타입을 참조하세요.	2020년 2월 6일

새로운 “보안” 장	Amazon EC2 Auto Scaling 사용자 가이드의 새로운 보안 장은 Amazon EC2 Auto Scaling을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 이번 업데이트에서는 사용자 가이드의 "Amazon EC2 Auto Scaling 리소스에 대한 액세스 통제" 장이 새롭고 더욱 유용한 Amazon EC2 Auto Scaling의 자격 증명 및 액세스 관리 섹션으로 교체되었습니다.	2020년 2월 4일
인스턴스 타입에 대한 권장 사항	AWS Compute Optimizer 성능 향상, 비용 절감 또는 두 가지 모두에 도움이 되는 Amazon EC2 인스턴스 권장 사항을 제공합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 인스턴스 타입에 대한 권장 사항 가져오기를 참조하세요.	2019년 12월 3일
전용 호스트 및 호스트 리소스 그룹	호스트 리소스 그룹을 지정하는 출범 템플릿을 만드는 방법을 보여주는 설명서가 업데이트되었습니다. 이렇게 하면 전용 호스트에서 사용할 BYOL AMI를 지정하는 출범 템플릿이 있는 Auto Scaling 그룹을 생성할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Auto Scaling 그룹을 위한 출범 템플릿 생성을 참조하세요.	2019년 12월 3일

Amazon VPC 엔드포인트에 대한 지원	이제 VPC와 Amazon EC2 Auto Scaling 간에 프라이빗 연결을 설정할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling 및 인터페이스 VPC 엔드포인트를 참조하세요.	2019년 11월 22일
최대 인스턴스 수명	이제 인스턴스가 서비스될 수 있는 최대 시간을 지정하여 인스턴스를 자동으로 교체할 수 있습니다. Amazon EC2 Auto Scaling은 이 제한에 가까워지는 인스턴스를 점진적으로 교체합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 최대 인스턴스 수명을 기준으로 Auto Scaling 인스턴스 교체를 참조하세요.	2019년 11월 19일
인스턴스 가중치 부여	여러 인스턴스 타입이 포함된 Auto Scaling 그룹의 경우, 각 인스턴스 타입이 그룹의 용량에 기여하는 용량 유닛 수를 선택적으로 지정할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon EC2 Auto Scaling에 대해 인스턴스 가중치 구성을 참조하세요.	2019년 11월 19일

최소 인스턴스 타입 수	스팟, 온디맨드 및 예약 인스턴스 그룹에 대한 추가 인스턴스 타입을 더 이상 지정할 필요가 없습니다. 모든 Auto Scaling 그룹에 대해 이제 최소 인스턴스 타입은 한 개입니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 여러 인스턴스 타입 및 구매 옵션이 포함된 Auto Scaling 그룹을 참조하세요.	2019년 9월 16일
새로운 스팟 할당 전략 지원	Amazon EC2 Auto Scaling은 이제 새로운 "용량 최적화" 스팟 할당 전략을 지원합니다. 이 전략은 사용 가능한 스팟 용량에 따라 최적으로 선택된 스팟 인스턴스 풀을 사용하여 사용자의 요청을 이행합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 여러 인스턴스 타입 및 구매 옵션이 포함된 Auto Scaling 그룹을 참조하세요.	2019년 8월 12일
설명서 변경 사항	Amazon EC2 Auto Scaling 설명서의 서비스 연결 역할 사용 및 암호화된 볼륨에 사용할 경우, 필요한 CMK 키 정책 주제가 개선되었습니다.	2019년 8월 1일

태깅에 대한 지원 향상	이제 Amazon EC2 Auto Scaling은 인스턴스를 출범하는 동일한 API 호출의 일부로 Amazon EC2 인스턴스에 태그를 추가합니다. 자세한 설명은 Auto Scaling 그룹 및 인스턴스에 태그 지정을 참조하세요.	2019년 7월 26일
설명서 변경 사항	Amazon EC2 Auto Scaling 설명서의 조정 프로세스의 일시 중지 및 재개 주제의 내용이 개선되었습니다. 사용자가 특정 맞춤 접미사 서비스 연결 역할만 Amazon EC2 Auto Scaling에 전달할 수 있도록 허용하는 정책 예를 포함하도록 고객 관리형 정책 예가 업데이트되었습니다.	2019년 6월 13일
새로운 Amazon EBS 기능 지원	출범 템플릿 주제에 새로운 Amazon EBS 기능에 대한 지원 내용을 추가했습니다. 스냅샷에서 복원할 때 EBS 볼륨의 암호화 상태를 변경합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Auto Scaling 그룹을 위한 출범 템플릿 생성을 참조하세요.	2019년 5월 13일

설명서 변경 사항	Amazon EC2 Auto Scaling 설명서의 축소 중 해지할 Auto Scaling 인스턴스 제어,Auto Scaling 그룹,여러 인스턴스 타입 및 구매 옵션이 포함된 Auto Scaling 그룹 및 Amazon EC2 Auto Scaling에 대한 동적 조정 섹션의 내용이 개선되었습니다.	2019년 3월 12일
인스턴스 타입 및 구매 옵션 조합 지원	단일 Auto Scaling 그룹 내에서 구매 옵션(스팟, 온디맨드, 예약 인스턴스) 및 인스턴스 타입 간에 인스턴스를 프로비저닝하고 자동으로 스케일 아웃합니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 여러 인스턴스 타입 및 구매 옵션이 포함된 Auto Scaling 그룹을 참조하세요.	2018년 11월 13일
Amazon SQS에 따라 조정과 관련된 주제가 업데이트됨	맞춤 지표를 사용하여 Amazon SQS 대기열의 수요 변화에 따라 Auto Scaling 그룹을 조정할 수 있는 방법을 설명하기 위해 설명서를 업데이트했습니다. 자세한 설명은 Amazon EC2 Auto Scaling 사용자 가이드의 Amazon SQS 기반 조정을 참조하세요.	2018년 7월 26일

다음 표에서는 2018년 7월 이전에 Amazon EC2 Auto Scaling 설명서에서 변경된 중요 사항에 대해 설명합니다.

기능	설명	릴리스 날짜
대상 추적 조정 정책을 지원	몇 가지 간단한 단계를 통해 애플리케이션용 동적 조정을 설정할 수 있습니다. 자세한 설명은 Amazon EC2 Auto Scaling에 대한 대상 추적 조정 정책을 참조하세요.	2017년 7월 12일
리소스 수준 권한에 대한 지원	리소스 수준에서 액세스를 제어하려면 IAM 정책을 만듭니다. 자세한 설명은 Amazon EC2 Auto Scaling 리소스에 대한 액세스 통제를 참조하세요.	2017년 5월 15일
모니터링 개선	Auto Scaling 그룹 지표는 더 이상 세부 모니터링 활성화를 요구하지 않습니다. 이제 콘솔의 Monitoring(모니터링) 탭에서 그룹 지표 수집을 활성화하고 지표 그래프를 볼 수 있습니다. 자세한 내용은 Amazon을 사용하여 Auto Scaling 그룹 및 인스턴스 모니터링을 참조하십시오 CloudWatch.	2016년 8월 18일
Application Load Balancer 지원	새 그룹 또는 기존 Auto Scaling 그룹에 하나 이상의 대상 그룹을 연결합니다. 자세한 설명은 로드 밸런서를 Auto Scaling 그룹에 연결을 참조하세요.	2016년 8월 11일
라이프사이클 후크 이벤트	Amazon EC2 Auto Scaling은 라이프사이클 후크를 EventBridge 호출할 때 이벤트를 전송합니다. 자세한 내용은 Auto Scaling 그룹 확장 EventBridge 시기 확인을 참조하십시오.	2016년 2월 24일
인스턴스 보호	축소 시 Amazon EC2 Auto Scaling이 특정 해지 인스턴스를 선택하지 않도록 합니다. 자세한 설명은 인스턴스 보호를 참조하세요.	2015년 12월 07일
단계별 조정 정책	경보 위반의 크기에 따라 조정할 수 있는 조정 정책을 생성합니다. 자세한 설명은 조정 정책 타입을 참조하세요.	2015년 7월 06일
로드 밸런서 업데이트	기존 Auto Scaling 그룹에 로드 밸런서를 연결하거나 로드 밸런서에서 분리합니다. 자세한 설명은 로드 밸런서를 Auto Scaling 그룹에 연결을 참조하세요.	2015년 6월 11일
에 대한 지원 ClassicLink	Auto Scaling 그룹의 EC2-Classic 인스턴스를 VPC에 연결하고, 연결된 EC2-Classic 인스턴스와 VPC의 인스턴스 간에 프라이빗 IP 주소로 서로 통신합니다. 자세한 설명은	2015년 1월 19일

기능	설명	릴리스 날짜
	VPC에 대한 EC2-Classic 인스턴스 링크 설정을 참조하세요.	
라이프사이클 후크	새로 시작된 인스턴스나 해지되는 인스턴스에 대해 작업을 수행하는 동안 보류 상태로 유지합니다. 자세한 설명은 Amazon EC2 Auto Scaling 라이프사이클 후크를 참조하세요.	2014년 7월 30일
인스턴스 분리	Auto Scaling 그룹에서 인스턴스를 분리합니다. 자세한 설명은 Auto Scaling 그룹에서 EC2 인스턴스 분리를 참조하세요.	2014년 7월 30일
인스턴스를 대기 상태로 설정	`InService` 상태인 인스턴스를 `Standby` 상태로 변경합니다. 자세한 설명은 Auto Scaling 그룹에서 일시적으로 인스턴스 제거를 참조하세요.	2014년 7월 30일
태그 관리	AWS Management Console을 사용하여 Auto Scaling 그룹을 관리합니다. 자세한 설명은 Auto Scaling 그룹 및 인스턴스에 태그 지정을 참조하세요.	2014년 5월 1일
전용 인스턴스 지원	출범 구성을 생성할 때 배치 테넌시 속성을 지정하여 전용 인스턴스를 출범합니다. 자세한 설명은 인스턴스 배치 테넌시를 참조하세요.	2014년 4월 23일
EC2 인스턴스에서 그룹 또는 출범 구성 생성	EC2 인스턴스를 사용하여 새 Auto Scaling 그룹 또는 새 출범 구성을 생성합니다. EC2 인스턴스를 사용하여 출범 구성을 생성하는 방법에 대한 자세한 설명은 EC2 인스턴스를 사용하여 출범 구성 만들기를 참조하세요. EC2 인스턴스를 사용하여 Auto Scaling 그룹을 생성하는 방법에 대한 자세한 설명은 EC2 인스턴스를 사용하여 Auto Scaling 그룹 생성하기를 참조하세요.	2014년 1월 02일
인스턴스 연결	기존 Auto Scaling 그룹에 인스턴스를 연결하여 EC2 인스턴스에 대한 자동 조정을 활성화합니다. 자세한 설명은 Auto Scaling 그룹에 EC2 인스턴스 연결을 참조하세요.	2014년 1월 02일

기능	설명	릴리스 날짜
계정 제한 보기	계정의 Auto Scaling 리소스에 대한 제한을 봅니다. 자세한 설명은 Auto Scaling 제한을 참조하세요.	2014년 1월 02일
Amazon EC2 Auto Scaling 콘솔 지원	를 사용하여 Amazon EC2 Auto Scaling에 액세스할 수 있습니다. AWS Management Console자세한 설명은 Amazon EC2 Auto Scaling 시작하기를 참조하세요.	2013년 12월 10일
퍼블릭 IP 주소 배정	VPC로 시작된 인스턴스에 퍼블릭 IP 주소를 배정합니다. 자세한 설명은 VPC에서 Auto Scaling 인스턴스 출범을 참조하세요.	2013년 9월 19일
인스턴스 해지 정책	EC2 해지 시 사용할 Amazon EC2 Auto Scaling에 대한 인스턴스 해지 정책을 지정합니다. 자세한 설명은 축소 중 해지할 Auto Scaling 인스턴스 제어를 참조하세요.	2012년 9월 17일
IAM 역할 지원	IAM 인스턴스 프로필로 EC2 인스턴스를 출범합니다. 이 기능을 사용하여 IAM 역할을 인스턴스에 할당하여 애플리케이션에서 기타 Amazon Web Services에 안전하게 액세스할 수 있습니다. 자세한 설명은 IAM 역할을 사용하여 Auto Scaling 인스턴스 출범을 참조하세요.	2012년 6월 11일
스팟 인스턴스 지원	출범 구성으로 스팟 인스턴스를 출범합니다. 자세한 설명은 내결함성 및 유연한 애플리케이션을 위한 스팟 인스턴스 요청을 참조하세요.	2012년 6월 7일
그룹 및 인스턴스 태그 지정	Auto Scaling 그룹의 태그를 지정하고 태그가 생성된 후 시작된 EC2 인스턴스에도 이 태그가 적용되도록 지정합니다. 자세한 설명은 Auto Scaling 그룹 및 인스턴스에 태그 지정을 참조하세요.	2012년 1월 26일

기능	설명	릴리스 날짜
Amazon SNS 지원	Amazon SNS를 사용하여 Amazon EC2 Auto Scaling이 EC2 인스턴스를 출범 또는 해지할 때마다 알림을 수신할 수 있습니다. 자세한 설명은 Auto Scaling 그룹 조정 시 SNS 알림 받기를 참조하세요. Amazon EC2 Auto Scaling에는 다음의 새 기능도 추가되었습니다. • Cron 구문을 사용하여 크기 조정 활동을 설정하는 기능. 자세한 설명은 `PutScheduledUpdateGroupAction` API 작업을 참조하세요. • 시작된 인스턴스를 로드 밸런서에 추가하지 않고 확장할 수 있는 새로운 구성 설정 ()LoadBalancer. 자세한 설명은 `ProcessType` API 데이터 형식을 참조하세요. • 인스턴스가 먼저 해지될 때까지 기다리지 않고 Auto Scaling 그룹과 여기에 연결된 인스턴스를 삭제하도록 Amazon EC2 Auto Scaling에 지시하는 `DeleteAutoScalingGroup` 작업의 `ForceDelete` 플래그. 자세한 설명은 `DeleteAutoScalingGroup` API 작업을 참조하세요.	2011년 7월 20일
예약된 조정 작업	예약된 조정 작업에 대한 지원을 추가했습니다. 자세한 설명은 Amazon EC2 Auto Scaling의 예약 조정을 참조하세요.	2010년 12월 2일
Amazon VPC 지원	Amazon VPC 지원이 추가되었습니다. 자세한 설명은 VPC에서 Auto Scaling 인스턴스 출범을 참조하세요.	2010년 12월 2일
HPC 클러스터 지원	HPC(고성능 컴퓨팅) 클러스터에 대한 지원이 추가되었습니다.	2010년 12월 2일
건전성 체크 지원	Amazon EC2 Auto Scaling 관리형 EC2 인스턴스에 Elastic Load Balancing 건전성 체크 사용에 대한 지원을 추가했습니다. 자세한 내용은 Auto Scaling 그룹의 인스턴스 상태 확인을 참조하십시오.	2010년 12월 2일

기능	설명	릴리스 날짜
CloudWatch 알람 지원	이전 트리거 메커니즘을 제거하고 알람 기능을 사용하도록 Amazon EC2 Auto Scaling을 CloudWatch 재설계했습니다. 자세한 설명은 Amazon EC2 Auto Scaling의 동적 조정을 참조하세요.	2010년 12월 2일
조정 일시 중지 및 재개	조정 프로세스의 일시 중지 및 재개에 대한 지원이 추가되었습니다.	2010년 12월 2일
IAM 지원	IAM에 대한 지원이 추가되었습니다. 자세한 설명은 Amazon EC2 Auto Scaling 리소스에 대한 액세스 통제를 참조하세요.	2010년 12월 2일

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