aws

사용자 가이드

EC2 Image Builder

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

EC2 Image Builder: 사용자 가이드

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

Amazon의 상표 및 트레이드 드레스는 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

Image Builder란 무엇입니까? 1
Image Builder의 기능 2
지원되는 운영 체제 3
지원되는 이미지 형식 4
기본 할당량 4
AWS 리전 및 엔드포인트 4
개념 4
요금 7
관련 AWS 서비스 8
Image Builder 작동 방식 8
AMI 요소 9
구성 요소 관리 9
생성할 리소스 10
배포 11
리소스 공유 11
규정 준수 12
의미 체계 버전 관리 12
설정 14
Image Builder 서비스 연결 역할 14
구성 요구 사항 15
컨테이너 이미지 파이프라인에 대한 컨테이너 리포지토리 16
macOS 이미지 전용 호스트 16
IAM 사전 조건 16
Systems Manager 에이전트 사전 조건 17
Image Builder 자습서 19
첫 번째 이미지 빌드 19
입력 파라미터를 사용하여 사용자 지정 구성 요소 만들기 19
Image Builder에서 Systems Manager 파라미터 사용 20
파이프라인 마법사: AMI 생성 20
1단계: 파이프라인 세부 정보 지정 20
2단계: 레시피 선택 21
3단계: 인프라 구성 정의 - 선택 사항 24
4단계: 배포 설정 정의 - 선택 사항 24
5단계: 검토 24

6단계: 정리 25
파이프라인 마법사: 컨테이너 이미지 생성 27
1단계: 파이프라인 세부 정보 지정 27
2단계: 레시피 선택 28
3단계: 인프라 구성 정의 - 선택 사항 31
4단계: 배포 설정 정의 - 선택 사항 31
5단계: 검토 32
6단계: 정리 32
파라미터가 있는 사용자 지정 구성 요소 34
YAML 구성 요소 문서의 파라미터 사용 34
콘솔에서 Image Builder 레시피에 구성 요소 파라미터 설정 39
레시피에서 기본 이미지 파라미터 사용 39
1단계: 파라미터 스토어 파라미터 찾기 또는 생성 40
2단계: IAM 권한 구성 41
3단계: 파라미터를 사용하는 이미지 레시피 생성 42
Components 43
구성 요소 나열 및 보기 45
Image Builder 구성 요소 나열 45
에서 구성 요소 빌드 버전 나열 AWS CLI 47
에서 구성 요소 세부 정보 가져오기 AWS CLI 48
에서 구성 요소 정책 세부 정보 가져오기 AWS CLI 49
AWS Marketplace 구성 요소 사용 49
AWS Marketplace 구성 요소 검색 49
AWS Marketplace 구성 요소 구독 51
레시피에서 AWS Marketplace 구성 요소 사용 51
관리형 구성 요소 사용 52
Distributor 패키지 Windows 애플리케이션 설치 52
CIS 하드닝 57
STIG 하드닝 구성 요소 57
사용자 지정 구성 요소 개발 96
YAML 구성 요소 문서 생성 96
Image Builder로 사용자 지정 구성 요소 생성 100
AWSTOE 구성 요소 관리자 애플리케이션 110
AWSTOE 다운로드 110
지원되는 리전 112
명령 참조 114

수동 설정 119
구성 요소 문서 프레임워크 사용 132
작업 모듈 181
입력 구성 285
이미지 리소스 289
이미지 및 빌드 버전 나열 289
이미지 나열 290
작업 대기 중인 이미지 나열 295
이미지 빌드 버전 나열 297
이미지 리소스 세부 정보 보기 300
Image Builder 콘솔에서 이미지 세부 정보 보기 300
에서 이미지 정책 세부 정보 가져오기 AWS CLI 307
Image Builder를 사용하여 사용자 지정 이미지 생성 307
에서 이미지 생성 취소 AWS CLI 310
VM 이미지 가져오기 및 내보내기 310
Image Builder로 VM 가져오기 311
의 이미지 빌드에서 VM 디스크 배포 AWS CLI 315
ISO 디스크 이미지 가져오기 315
ISO 디스크 이미지 가져오기에 지원되는 운영 체제 315
사전 조건 316
Image Builder로 ISO 디스크 이미지 가져오기 317
보안 결과 관리 321
보안 스캔 구성 321
보안 결과 관리 323
Image Builder 리소스 정리 325
이미지 수명 주기 관리 326
사전 조건 327
Image Builder 수명 주기 관리를 위한 IAM 역할 생성 328
Image Builder 계정 간 수명 주기 관리를 위한 IAM 역할 생성 329
수명 주기 정책 나열 330
수명 주기 정책 보기 333
Image Builder 콘솔에서 수명 주기 정책 세부 정보 보기 333
수명 주기 정책 생성 336
수명 주기 정책 생성(AMI 이미지) 336
수명 주기 정책 생성(컨테이너 이미지) 339
수명 주기 규칙 작동 방식 341

AMI 수명 주기 제외 규칙 342
수명 주기 관리 규칙 세부 정보 보기 343
사용자 지정 이미지 구성 344
레시피 344
이미지 레시피 나열 및 보기 345
컨테이너 레시피 나열 및 보기 347
이미지 레시피의 새 버전 생성 349
컨테이너 레시피의 새 버전 생성 361
리소스 정리 370
인프라 구성 370
인프라 구성 나열 및 보기 372
인프라 구성 생성 372
인프라 구성 업데이트 377
AWS PrivateLink VPC 엔드포인트 379
배포 설정 384
배포 구성 나열 및 보기 386
AMI 배포 생성 및 업데이트 387
컨테이너 이미지 배포 생성 및 업데이트 400
크로스 계정 AMI 배포 설정 403
AMI 시작 템플릿 지정하기 411
리소스 공유 414
리소스 소유자 415
Image Builder 리소스를 공유하기 위한 사전 조건 415
리소스 소비자 416
리소스 공유 생성 416
옵션 1: RAM 리소스 공유 생성 416
옵션 2: 리소스 정책을 적용하고 기존 리소스 공유로 승격하기 417
리소스 공유 해제 419
리소스 태깅 420
에서 리소스에 태그 지정 AWS CLI 420
에서 리소스의 태그를 해제합니다. AWS CLI 421
에서 특정 리소스에 대한 모든 태그 나열 AWS CLI 421
리소스 삭제하기 422
콘솔: 리소스 삭제 422
리소스 삭제하기(AWS CLI) 423
이미지 워크플로 426

워크플로 프레임워크: 단계 426
서비스 액세스 427
관리형 워크플로 사용 427
이미지 워크플로 나열 428
이미지 워크플로 생성 431
YAML 워크플로 문서 생성 434
YAML 워크플로 문서의 구조 434
단계 작업 443
동적 변수 460
조건문 463
파이프라인 관리 469
파이프라인 나열 및 보기 470
에서 이미지 파이프라인 나열 AWS CLI 470
에서 이미지 파이프라인 세부 정보 가져오기 AWS CLI 470
파이프라인 생성 및 업데이트(AMI) 470
에서 AMI 파이프라인 생성 AWS CLI 471
콘솔에서 파이프라인 업데이트 473
에서 파이프라인 업데이트 AWS CLI 476
파이프라인 생성 및 업데이트(컨테이너) 478
에서 파이프라인 생성 AWS CLI 479
콘솔에서 파이프라인 업데이트 480
에서 파이프라인 업데이트 AWS CLI 484
파이프라인 워크플로 구성 486
테스트 워크플로를 위한 테스트 그룹 정의 486
콘솔에서 Image Builder 파이프라인의 워크플로 파라미터 설정 487
Image Builder가 워크플로 작업을 실행하는 데 사용하는 IAM 서비스 역할 지정 487
파이프라인 실행 488
cron 표현식 사용 488
Image Builder에서 Cron 표현식에 대해 지원되는 값 489
Image Builder의 cron 표현식 예제 491
rate 표현식 493
EventBridge 규칙 사용 494
EventBridge 용어 494
Image Builder 파이프라인에 대한 EventBridge 규칙 보기 495
EventBridge 규칙을 사용하여 파이프라인 빌드를 예약하세요. 496
제품 및 서비스 통합 498

Amazon EventBridge ... 500
Image Builder에서 보내는 이벤트 메시지 ... 501
Amazon Inspector ... 504
AWS Marketplace ... 505
AWS Marketplace Image Builder의 구독 ... 505
AWS Marketplace Image Builder 콘솔에서 이미지 제품 검색 ... 506
AWS Marketplace Image Builder 레시피에서 이미지 제품 사용 ... 508
Amazon Simple Notification Service ... 509
암호화된 SNS 주제 ... 509
SNS 메시지 형식 ... 511
규정 준수 제품 ... 517
이벤트와 로그 모니터링 ... 518
CloudTrail 로그 ... 518
CloudTrail의 Image Builder 정보 ... 518
CloudWatch Logs ... 519
Image Builder의 보안 ... 521
데이터 보호 ... 521
암호화 및 키 관리 ... 522
데이터 스토리지 ... 528
인터네트워크 트래픽 프라이버시 ... 528
ID 및 액세스 관리 ... 528
대상 ... 529
ID를 통한 인증 ... 529
Image Builder가 IAM 정책 및 역할을 사용하는 방법 ... 530
데이터 경계 관리 ... 541
자격 증명 기반 정책 ... 541
리소스 기반 정책 ... 544
관리형 정책 ... 545
서비스 연결 역할 ... 558
문제 해결 ... 560
규정 준수 확인 ... 562
복원성 ... 562
인프라 보안 ... 563
패치 관리 ... 564
모범 사례 ... 565
빌드 후 정리 필요 ... 566

Linux 정리 스크립트를 오버라이드합니다. 577
Image Builder 문제 해결 580
파이프라인 빌드 문제 해결 580
문제 해결 시나리오 582
문서 기록 587
..... dxcix

Image Builder란 무엇입니까?

EC2 Image Builder는 안전하고 up-to-date의 사용자 지정 서버 이미지의 생성, 관리 및 배포를 자동화하는 데 도움이 AWS 서비스 되는 완전 관리형입니다. AWS Management Console AWS Command Line Interface또는 APIs를 사용하여에서 사용자 지정 이미지를 생성할 수 있습니다 AWS 계정.

Image Builder가 계정에서 생성하는 사용자 지정 이미지는 사용자가 소유합니다. 소유한 이미지에 대한 업데이트 및 시스템 패치를 자동화하도록 파이프라인을 구성할 수 있습니다. 독립 실행형 명령을 실행하여 정의한 구성 리소스로 이미지를 생성할 수도 있습니다.

Image Builder 파이프라인 마법사는 다음과 같이 사용자 지정 이미지를 생성하는 단계를 안내할 수 있습니다.

1단계: 파이프라인 세부 정보 지정

- 파이프라인의 이름을 지정하고 태그를 추가합니다.
- 메타데이터 및 취약성 스캔 설정을 정의합니다.
- 파이프라인 일정을 설정합니다.

2단계: 이미지 사용자 지정

기존 레시피를 선택하거나 새 레시피를 생성할 수 있습니다.

- 사용자 지정에 사용할 기본 이미지를 선택합니다.
- 기본 이미지에 소프트웨어를 추가하거나 제거합니다.
- 빌드 구성 요소로 설정 및 스크립트를 사용자 지정합니다.
- 실행할 테스트 구성 요소를 선택합니다.

3단계: 워크플로 정의

이미지 워크플로는 Image Builder가 이미지 생성 프로세스의 빌드 및 테스트 단계에서 수행하는 일련의 단계를 정의합니다. 이는 전체 Image Builder 워크플로 프레임워크의 일부입니다.

4단계: 빌드 인프라 구성

- 이미지 생성 프로세스 중에 Image Builder가 시작하는 인스턴스의 인스턴스 프로파일과 연결할 IAM 역할을 선택합니다.

- 시작 시 적용할 수 있는 인스턴스 유형을 하나 이상 선택합니다.
- Amazon Simple Notification Service(SNS) 주제를 선택하여 Image Builder로부터 알림을 받습니다.
- 이미지 생성 프로세스에 적용되는 VPC, 서브넷 및 보안 그룹을 지정합니다.
- Image Builder가 로그를 작성하는 위치, 장애 시 빌드 인스턴스를 종료할지(기본값) 아니면 추가 문제 해결을 위해 계속 실행할지와 같은 문제 해결 설정을 선택합니다.

5단계: 이미지 배포 정의

- Image Builder가 Amazon Machine Image(AMI) 또는 컨테이너 이미지를 배포할 AWS 리전 위치를 선택합니다.
- Image Builder 파이프라인에서 AMI를 생성하는 경우 Image Builder는 다음 구성도 지원합니다.
  - 암호화에 사용할 KMS 키를 선택합니다.
  - AWS 계정 및 조직 간에 AMI 공유를 구성합니다.
  - License Manager 자체 관리형 라이선스를 배포된 이미지와 연결합니다.
  - 이미지에 대한 시작 템플릿을 구성합니다.

Image Builder의 기능

EC2 Image Builder는 다음의 기능을 제공합니다.

규정을 준수하는 최신 이미지를 빌드하여 생산성 높이고 작업 줄이기

Image Builder는 빌드 파이프라인을 자동화하여 대규모로 이미지를 생성하고 관리하는 데 드는 작업량을 줄여줍니다. 원하는 빌드 실행 일정을 제공하여 빌드를 자동화할 수 있습니다. 자동화는 최신 운영 체제 패치로 소프트웨어를 유지 관리하는 데 드는 운영 비용을 줄여줍니다.

서비스 가동 시간 늘리기

Image Builder는 배포 전에 이미지를 테스트하는 데 사용할 수 있는 테스트 구성 요소에 대한 액세스를 제공합니다. AWS Task Orchestrator and Executor (AWSTOE)를 사용하여 사용자 지정 테스트 구성 요소를 생성하고 사용할 수도 있습니다. Image Builder는 구성된 모든 테스트가 성공한 경우에만 이미지를 배포합니다.

배포를 위한 보안 기준을 높이기

Image Builder를 사용하면 구성 요소 보안 취약성에 대한 불필요한 노출을 제거하는 이미지를 생성할 수 있습니다. AWS 보안 설정을 적용하여 업계 및 내부 보안 기준을 충족하는 안전한 out-of-the-box

이미지를 생성할 수 있습니다. Image Builder는 규제 대상 산업의 기업을 위한 설정 모음도 제공합니다. 이러한 설정을 사용하면 STIG 표준을 준수하는 이미지를 빠르고 쉽게 빌드할 수 있습니다. Image Builder를 통해 사용 가능한 STIG 구성 요소의 전체 목록은 Image Builder용 Amazon 관리형 STIG 하드닝 구성 요소 섹션을 참조하십시오.

중앙 집중식 적용 및 계보 추적

와 내장된 통합을 사용하는 AWS Organizations Image Builder를 사용하면 승인된 AMIs.

전체 리소스의 간소화된 공유 AWS 계정

EC2 Image Builder는 AWS Resource Access Manager (AWS RAM)와 통합되어 특정 리소스를 AWS 계정 또는를 통해 공유할 수 있습니다 AWS Organizations. 공유할 수 있는 EC2 Image Builder 리소스는 다음과 같습니다.

- Components
- 이미지
- 이미지 레시피
- 컨테이너 레시피

자세한 내용은 와 Image Builder 리소스 공유 AWS RAM 섹션을 참조하세요.

지원되는 운영 체제

Image Builder가 지원하는 운영 체제 버전은 다음과 같습니다.

운영 체제/배포	지원되는 버전
Amazon Linux	2 및 2023
CentOS	7 및 8
CentOS Stream	8
macOS	12.x(Monterey), 13.x(Ventura), 14.x(Sonoma), 15.x(Sequoia)
Red Hat Enterprise Linux(RHEL)	7, 8, 9 및 10

운영 체제/배포	지원되는 버전
SUSE Linux Enterprise Server(SUSE)	12 및 15
Ubuntu	18.04 LTS, 20.04 LTS, 22.04 LTS 및 24.04 LTS
Windows Server	2012 R2, 2016, 2019, 2022 및 2025

지원되는 이미지 형식

Amazon Machine Image(AMI)를 생성하는 사용자 지정 이미지의 경우 기존 AMI를 시작점으로 선택할 수 있습니다. Docker 컨테이너 이미지의 경우 DockerHub에서 호스팅되는 공용 이미지, Amazon ECR의 기존 컨테이너 이미지 또는 Amazon에서 관리하는 컨테이너 이미지 중에서 하나를 시작점으로 선택할 수 있습니다.

기본 할당량

Image Builder의 기본 할당량을 보려면, Image Builder 엔드포인트및 할당량을 참조하십시오.

AWS 리전 및 엔드포인트

Image Builder의 서비스 엔드포인트를 보려면 Image Builder 엔드포인트 및 할당량을 참조하십시오.

개념

다음 용어와 개념은 EC2 Image Builder의 이해와 사용에 매우 중요합니다.

AMI

Amazon Machine Image(AMI)는 Amazon EC2의 기본 배포 단위이며 Image Builder로 생성할 수 있는 이미지 유형 중 하나입니다. AMI는 EC2 인스턴스를 배포하기 위해 운영 체제(OS)와 사전 설치된 소프트웨어를 포함하는 사전 구성된 가상 머신 이미지입니다. 자세한 내용은 Amazon Machine Image(AMI)를 참조합니다.

이미지 파이프라인

이미지 파이프라인은 AWS에 보안 AMI 및 컨테이너 이미지를 빌드하기 위한 자동화 프레임워크를 제공합니다. Image Builder 이미지 파이프라인은 이미지 빌드 수명 주기의 빌드, 검증 및 테스트 단계를 정의하는 이미지 레시피 또는 컨테이너 레시피와 연결됩니다.

이미지 파이프라인을 이미지가 빌드되는 위치를 정의하는 인프라 구성과 연결할 수 있습니다. 인스턴스 유형, 서브넷, 보안 그룹, 로깅 및 기타 인프라 관련 구성과 같은 속성을 정의할 수 있습니다. 또한 이미지 파이프라인을 배포 구성과 연결하여 이미지를 배포할 방법을 정의할 수 있습니다.

관리 이미지

관리 이미지는 AMI 또는 컨테이너 이미지와 메타데이터(예: 버전 및 플랫폼)로 구성된 Image Builder의 리소스입니다. Image Builder 파이프라인은 관리 이미지를 사용하여 빌드에 사용할 기본 이미지를 결정합니다. 이 설명서에서는 관리 이미지를 '이미지'라고도 하지만 이미지는 AMI와 다릅니다.

이미지 레시피

Image Builder 이미지 레시피는 출력 AMI 이미지에 대해 원하는 구성을 생성하기 위해 기본 이미지 및 기본 이미지에 적용할 구성 요소를 정의하는 문서입니다. 이미지 레시피를 사용하여 빌드를 복제할 수 있습니다. Image Builder 이미지 레시피는 콘솔 마법사, AWS CLI또는 API를 사용하여 공유, 분기 및 편집할 수 있습니다. 버전 제어 소프트웨어와 함께 이미지 레시피를 사용하여 버전이 지정된 공유 가능한 이미지 레시피를 유지 관리할 수 있습니다.

컨테이너 레시피

Image Builder 컨테이너 레시피는 출력 컨테이너 이미지에 대해 원하는 구성을 생성하기 위해 기본 이미지 및 기본 이미지에 적용할 구성 요소를 정의하는 문서입니다. 컨테이너 레시피를 사용하여 빌드를 복제할 수 있습니다. Image Builder 이미지 레시피는 콘솔 마법사, AWS CLI또는 API를 사용하여 공유, 분기 및 편집할 수 있습니다. 버전 제어 소프트웨어와 함께 컨테이너 레시피를 사용하여 버전이 지정된 공유 가능한 컨테이너 레시피를 유지 관리할 수 있습니다.

기본 이미지

기본 이미지는 구성 요소와 함께 이미지 또는 컨테이너 레시피 문서에 사용되는 선택된 이미지 및 운영 체제입니다. 기본 이미지와 구성 요소 정의를 결합하여 원하는 출력 이미지 구성을 생성합니다.

Components

구성 요소는 이미지를 만들기 전에 인스턴스를 사용자 지정(빌드 구성 요소)하거나 생성된 이미지에서 시작된 인스턴스를 테스트(테스트 구성 요소)하는 데 필요한 단계 순서를 정의합니다.

구성 요소는 파이프라인에서 생성되는 인스턴스를 빌드하고 검증하거나 테스트하기 위한 런타임 구성을 설명하는 선언문, 평문 YAML 또는 JSON 문서에서 생성됩니다. 구성 요소는 구성 요소 관리 애플리케이션을 사용하여 인스턴스에서 실행됩니다. 구성 요소 관리 애플리케이션은 문서를 구문 분석하고 원하는 단계를 실행합니다.

구성 요소가 생성되면 이미지 레시피 또는 컨테이너 레시피를 사용하여 하나 이상의 구성 요소를 그룹화하여 가상 머신 또는 컨테이너 이미지를 빌드하고 테스트하기 위한 계획을 정의합니다. 에서 소유하고 관리하는 퍼블릭 구성 요소를 사용하거나 직접 생성할 AWS수 있습니다. 구성 요소에 대한 자세한 내용은 Image Builder가 AWS Task Orchestrator and Executor 애플리케이션을 사용하여 구성 요소를 관리하는 방법(을)를 참조합니다.

구성 요소 문서

이미지에 적용할 수 있는 사용자 지정용 구성을 설명하는 선언문, 평문 YAML 또는 JSON 문서입니다. 이 문서는 빌드 또는 테스트 구성 요소를 생성하는 데 사용됩니다.

런타임 단계

EC2 Image Builder에는 빌드 및 테스트라는 두 가지 런타임 단계가 있습니다. 각 런타임 단계에는 구성 요소 문서에 정의된 구성이 포함된 하나 이상의 단계가 있습니다.

구성 단계

다음 목록은 빌드 및 테스트 단계에서 실행되는 단계를 보여줍니다.

빌드 단계:

빌드 단계(phase)

이미지 파이프라인은 실행 시 빌드 단계의 빌드 단계에서 시작됩니다. 기본 이미지가 다운로드되고 구성 요소의 빌드 단계에 지정된 구성이 인스턴스를 빌드하고 시작하는 데 적용됩니다.

검증 단계

Image Builder가 인스턴스를 시작하고 모든 빌드 단계 사용자 지정을 적용한 후 검증 단계가 시작됩니다. 이 단계에서 Image Builder는 구성 요소가 검증 단계에 지정한 구성을 기반으로 모든 사용자 정의가 예상대로 작동하도록 합니다. 인스턴스 검증이 성공하면 Image Builder는 인스턴스를 중지하고 이미지를 만든 다음 테스트 단계를 계속합니다.

테스트 단계:

테스트 단계

이 페이즈에서 Image Builder는 검증 단계가 성공적으로 완료된 후 생성한 이미지에서 인스턴스를 시작합니다. Image Builder는 이 단계에서 테스트 구성 요소를 실행하여 인스턴스가 정상이고 예상대로 작동하는지 확인합니다.

컨테이너 호스트 테스트 단계

Image Builder가 컨테이너 레시피에서 선택한 모든 구성 요소에 대한 테스트 단계를 실행한 후 Image Builder는 컨테이너 워크플로를 위해 이 단계를 실행합니다. 컨테이너 호스트 테스트 단계에서는 컨테이너 관리 및 사용자 지정 런타임 구성을 검증하는 추가 테스트를 실행할 수 있습니다.

워크플로

워크플로는 Image Builder가 새 이미지를 생성할 때 수행하는 단계 순서를 정의합니다. 모든 이미지에는 빌드 및 테스트 워크플로가 있습니다. 컨테이너에는 배포를 위한 추가 워크플로가 있습니다.

워크플로 유형

`BUILD`

생성된 모든 이미지의 빌드 단계 구성을 다룹니다.

`TEST`

생성된 모든 이미지의 테스트 단계 구성을 다룹니다.

`DISTRIBUTION`

컨테이너 이미지의 배포 워크플로를 다룹니다.

요금

EC2 Image Builder를 사용하여 사용자 지정 AMI 또는 컨테이너 이미지를 생성하는 데는 비용이 들지 않습니다. 그러나 프로세스에 사용되는 다른 서비스에는 표준 요금이 적용됩니다. 다음 목록에는 구성에 따라 사용자 지정 AMI 또는 컨테이너 이미지를 생성, 빌드, 저장 및 배포할 때 비용이 발생할 수 AWS 서비스 있는 일부의 사용량이 포함되어 있습니다.

- EC2 인스턴스 시작
- Amazon S3에 로그 저장
- Amazon Inspector를 사용하여 이미지 검증하기
- AMI용 Amazon EBS 스냅샷 저장
- Amazon ECR에 컨테이너 이미지 저장
- 컨테이너 이미지를 Amazon ECR로 푸시하고 꺼내기
- Systems Manager Advanced Tier가 켜져 있고 Amazon EC2 인스턴스를 온프레미스 활성화와 함께 실행하는 경우, Systems Manager를 통해 리소스 요금이 부과될 수 있습니다.

관련 AWS 서비스

EC2 Image Builder는 Image Builder 레시피 구성에 따라 다른 AWS 서비스 를 사용하여 이미지를 빌드합니다. 사용자 지정 이미지의 제품 및 서비스 통합에 대한 자세한 내용은 Image Builder에서 제품 및 서비스 통합 섹션을 참조하세요.

EC2 Image Builder 작동 방식

EC2 Image Builder 콘솔을 사용하여 사용자 지정 이미지 파이프라인을 생성하면 시스템이 다음 단계를 안내합니다.

1. 파이프라인 세부 정보 지정 - 이름, 설명, 태그, 자동 빌드 실행 일정 등 파이프라인에 대한 정보를 입력합니다. 원하는 경우 수동 빌드를 선택할 수 있습니다.
2. 레시피 선택 — AMI 구축 또는 컨테이너 이미지 구축 중에서 선택합니다. 두 가지 유형의 출력 이미지 모두 레시피의 이름과 버전을 입력하고 기본 이미지를 선택한 다음, 빌드 및 테스트를 위해 추가할 구성 요소를 선택합니다. 또한 자동 버전 관리를 선택하여 기본 이미지에 사용 가능한 최신 운영 체제(OS) 버전을 항상 사용할 수 있습니다. 컨테이너 레시피는 Dockerfile을 추가로 정의하고 출력 Docker 컨테이너 이미지의 대상 Amazon ECR 리포지토리를 정의합니다.

   > **Note**
   >
   > 구성 요소는 이미지 레시피 또는 컨테이너 레시피가 사용하는 구성 요소입니다. 예를 들어 설치용 패키지, 보안 하드닝 단계, 테스트 등이 있습니다. 선택한 기본 이미지와 구성 요소가 이미지 레시피를 구성합니다.

3. 인프라 구성 정의 — Image Builder는 계정에서 EC2 인스턴스를 시작하여 이미지를 사용자 지정하고 검증 테스트를 실행합니다. 인프라 구성 설정은 빌드 프로세스 AWS 계정 중에에서 실행할 인스턴스에 대한 인프라 세부 정보를 지정합니다.
4. 배포 설정 정의 — 빌드가 완료되고 모든 테스트를 통과한 후 이미지를 배포할 AWS 리전을 선택합니다. 파이프라인은 빌드를 실행하는 리전에 이미지를 자동으로 배포하며, 다른 리전에 이미지 배포를 추가할 수 있습니다.

사용자 지정 기본 이미지로 빌드한 이미지는 AWS 계정에 있습니다. 빌드 일정을 입력하여 이미지의 업데이트 및 패치 버전을 생성하도록 이미지 파이프라인을 구성할 수 있습니다. 빌드가 완료되면 Amazon Simple Notification Service(SNS)를 통해 알림을 받을 수 있습니다. Image Builder 콘솔 마법사는 최종 이미지를 생성하는 것 외에도 반복 가능한 자동화를 위해 기존 버전 제어 시스템 및 CI/

CD(지속적 통합/지속적 배포) 파이프라인과 함께 사용할 수 있는 레시피를 생성합니다. 레시피를 공유하고 새 버전을 만들 수 있습니다.

섹션 목차

- AMI 요소
- 구성 요소 관리
- 생성할 리소스
- 배포
- 리소스 공유
- 규정 준수

AMI 요소

Amazon Machine Image(AMI)는 EC2 인스턴스를 배포하기 위한 OS 및 소프트웨어가 포함된 사전 구성된 가상 머신(VM) 이미지입니다.

AMI는 다음 요소를 포함합니다.

- VM의 루트 볼륨을 위한 템플릿입니다. Amazon EC2 VM을 시작하면, 인스턴스 부팅에 이미지가 루트 디바이스 볼륨에 저장됩니다. 인스턴스 스토어가 사용되는 경우, 루트 디바이스는 Amazon S3의 템플릿에서 생성된 인스턴스 스토어 볼륨입니다. 자세한 내용은 Amazon EC2 루트 디바이스 볼륨 섹션을 참조하십시오.
- Amazon EBS를 사용하는 경우, 루트 디바이스는 EBS 스냅샷에서 생성된 EBS 볼륨입니다.
- AMI를 사용하여 VMs 시작할 수 AWS 계정 있는를 결정하는 시작 권한입니다.
- 시작 후 인스턴스에 연결할 볼륨을 지정하는 블록 디바이스 매핑 데이터.
- 각 리전, 계정별 고유한 리소스 식별자.
- 메타데이터 페이로드(예: 태그) 및 속성(예: 리전, 운영 체제, 아키텍처, 루트 디바이스 유형, 공급자, 시작 권한, 루트 디바이스용 스토리지, 서명 상태)
- Windows 이미지의 AMI 시그니처는 무단 변조를 방지하기 위한 것입니다. 자세한 내용은 인스턴스 ID 문서를 참조하십시오.

구성 요소 관리

EC2 Image Builder는 복잡한 워크플로를 오케스트레이션하고, 시스템 구성을 수정하고, YAML 기반 스크립트 구성 요소로 시스템을 테스트하는 데 도움이 되는 구성 요소 관리 애플리케이션 AWS Task

Orchestrator and Executor (AWSTOE)을 사용합니다. AWSTOE 는 독립 실행형 애플리케이션이므로 추가 설정이 필요하지 않습니다. 모든 클라우드 인프라 및 온프레미스에서 실행할 수 있습니다. 를 독립 실행형 애플리케이션 AWSTOE 으로 사용하려면 섹션을 참조하세요를 사용하여 사용자 지정 구성 요소를 개발하도록 수동 설정 AWSTOE.

Image Builder는 AWSTOE 를 사용하여 모든 온인스턴스 활동을 수행합니다. 여기에는 스냅샷을 찍기 전에 이미지를 만들고 검증하고, 최종 이미지를 만들기 전에 예상대로 작동하는지 확인하는 스냅샷을 테스트하는 것이 포함됩니다. Image Builder가 AWSTOE 를 사용하여 구성 요소를 관리하는 방법에 대한 자세한 내용은 섹션을 참조하세요구성 요소를 사용하여 Image Builder 이미지 사용자 지정. AWSTOE(으)로 구성 요소를 생성하는 작업에 관한 자세한 내용은 Image Builder가 AWS Task Orchestrator and Executor 애플리케이션을 사용하여 구성 요소를 관리하는 방법 섹션을 참조하십시오.

이미지 테스트

테스트 AWSTOE 구성 요소를 사용하여 최종 이미지를 생성하기 전에 이미지를 검증하고 예상대로 작동하는지 확인할 수 있습니다.

일반적으로 각 테스트 구성 요소는 테스트 스크립트, 테스트 바이너리, 테스트 메타데이터가 포함된 YAML 문서로 구성됩니다. 테스트 스크립트에는 테스트 바이너리를 시작하기 위한 오케스트레이션 명령이 포함되어 있으며, OS에서 지원하는 모든 언어로 작성할 수 있습니다. 종료 상태 코드는 테스트 결과를 나타냅니다. 테스트 메타데이터는 이름, 설명, 테스트 바이너리 경로, 예상 기간 등 테스트와 해당 동작을 설명합니다.

생성할 리소스

파이프라인을 생성할 때, 다음과 같은 경우를 제외하고 Image Builder 외부 리소스는 생성되지 않습니다.

- 파이프라인 일정을 통해 이미지가 생성되는 경우
- Image Builder 콘솔의 작업 메뉴에서 파이프라인 실행을 선택하는 경우
- API에서 이러한 명령 중 하나를 실행하거나 AWS CLIStartImagePipelineExecution 또는 CreateImage

이미지 빌드 프로세스 중에 다음과 같은 리소스가 생성됩니다.

AMI 이미지 파이프라인

- EC2 인스턴스(임시)

- EC2 인스턴스의 Systems Manager 인벤토리 연결(EnhancedImageMetadata가 활성화된 경우 Systems Manager State Manager를 통해)
- Amazon EC2 AMI
- Amazon EC2 AMI와 관련된 Amazon EBS 스냅샷

컨테이너 이미지 파이프라인

- EC2 인스턴스에서 실행되는 Docker 컨테이너(임시)
- EC2 인스턴스에서 Systems Manager Inventory Association(Systems Manager State Manager를 통해) EnhancedImageMetadata(이)가 활성화됨
- Docker 컨테이너 이미지
- Dockerfile

이미지가 생성되면 모든 임시 리소스가 삭제됩니다.

배포

EC2 Image Builder는 AMIs 또는 컨테이너 이미지를 모든 AWS 리전에 배포할 수 있습니다. 이미지는 이미지를 빌드하는 데 사용된 계정에 지정한 각 리전에 복사됩니다.

AMI 출력 이미지의 경우 AMI 시작 권한을 정의하여 생성된 AMI로 EC2 인스턴스를 AWS 계정 시작할 수 있는 권한을 제어할 수 있습니다. 예를 들어 이미지를 프라이빗, 퍼블릭 또는 특정 계정과 공유하도록 설정할 수 있습니다. AMI를 다른 리전에 배포하고 다른 계정의 시작 권한을 정의하는 경우, 시작 권한은 AMI가 배포되는 모든 리전의 AMI에 전파됩니다.

또한 계정을 사용하여 멤버 AWS Organizations 계정에 제한을 적용하여 승인되고 규정을 준수하는 AMIs. 자세한 내용은 조직에서 관리를 참조 AWS 계정 하세요.

Image Builder 콘솔을 사용하여 배포 설정을 업데이트하려면, 콘솔에서 새 이미지 레시피 버전 생성 또는 콘솔을 사용하여 새 컨테이너 레시피 버전 생성 단계를 따르십시오.

리소스 공유

구성 요소, 레시피 또는 이미지를 다른 계정 또는 내부와 공유하려면 섹션을 AWS Organizations참조하세요와 Image Builder 리소스 공유 AWS RAM.

규정 준수

CIS(Center for Internet Security)의 경우, EC2 Image Builder는 Amazon Inspector를 사용하여 노출 및 취약성, 그리고 모범 사례 및 규정 준수 표준과의 편차를 평가합니다. 예를 들어 Image Builder는 의도하지 않은 네트워크 접근성, 패치가 적용되지 않은 CVE, 공용 인터넷 연결, 원격 루트 로그인 활성화 등을 평가합니다. Amazon Inspector는 이미지 레시피에 추가하도록 선택할 수 있는 테스트 구성 요소로 제공됩니다. Amazon Inspector에 대한 자세한 내용은 Amazon Inspector 사용 설명서를 참조하세요. 자세한 내용은 CIS(Center for Internet Security) 벤치마크를 참조하세요.

Image Builder는 STIG 하드닝 구성 요소를 제공하여 기본 STIG 표준을 준수하는 이미지를 보다 효율적으로 구축할 수 있도록 지원합니다. 이러한 STIG 구성 요소는 잘못된 구성을 검사하고 수정 스크립트를 실행합니다. STIG 규정을 준수하는 구성 요소를 사용하기 위해 별도의 비용은 필요하지 않습니다. Image Builder를 통해 사용 가능한 STIG 구성 요소의 전체 목록은 Image Builder용 Amazon 관리형 STIG 하드닝 구성 요소 섹션을 참조하십시오.

Image Builder의 의미 체계 버전 관리

Image Builder는 의미 체계 버전 관리를 사용하여 리소스를 구성하고 리소스가 고유한 ID를 갖도록 합니다. 의미 체계 버전에는 다음과 같은 4가지 노드가 있습니다.

<major>.*<minor>*.*<patch>*/<build>

처음 3개의 값을 할당하고 모든 값을 필터링할 수 있습니다.

의미 체계 버전 관리는 다음과 같이 각 객체의 Amazon 리소스 이름(ARN)에 해당 객체에 적용되는 수준으로 포함됩니다.

1. 버전이 없는 ARN과 이름 ARN은 어떤 노드에도 특정 값을 포함하지 않습니다. 노드는 완전히 제외되거나 와일드카드로 지정됩니다(예: x.x.x).
2. 버전 ARN에는 처음 세 개의 노드, <major>.<minor>.<patch>만 있습니다
3. 빌드 버전 ARN에는 노드 4개가 모두 있으며 객체의 특정 버전에 대한 특정 빌드를 가리킵니다.

할당: 처음 세 노드의 경우 각 노드에 대해 상한이 2^30-1 또는 1073741823인 양의 정수 값 또는 0을 할당할 수 있습니다. Image Builder는 자동으로 4번째 노드에 빌드 번호를 할당합니다.

패턴: 할당할 수 있는 노드에 대한 할당 요구 사항을 준수하는 모든 숫자 패턴을 사용할 수 있습니다. 예를 들어 1.0.0과 같은 소프트웨어 버전 패턴이나 2021.01.01과 같은 날짜를 선택할 수 있습니다.

필터링: 의미 체계 버전 관리를 사용하면 레시피에 대한 기본 이미지 또는 구성 요소를 선택할 때 와일드카드(x)를 사용하여 최신 버전 또는 노드를 지정할 수 있습니다. 임의의 노드에서 와일드카드를 사용하는 경우 첫 번째 와일드카드 오른쪽에 있는 모든 노드도 와일드카드여야 합니다.

예를 들어 2.2.4, 1.7.8, 1.6.8의 최신 버전에서 와일드카드를 사용하여 버전을 선택하면 다음과 같은 결과가 나타납니다.

- `x.x.x` = 2.2.4
- `1.x.x` = 1.7.8
- `1.6.x` = 1.6.8
- `x.2.x` 값은 유효하지 않아 오류가 발생합니다.
- `1.x.8` 값은 유효하지 않아 오류가 발생합니다.

Image Builder를 사용하여 사용자 지정 이미지를 빌드하도록 설정

EC2 Image Builder를 사용하여 이미지를 빌드하기 전에 다음 사전 조건을 충족하는지 확인하십시오. 달리 명시되지 않는 한, 모든 유형의 파이프라인에는 이 사전 조건이 필요합니다.

사전 조건

- Image Builder 서비스 연결 역할
- 구성 요구 사항
- 컨테이너 이미지 파이프라인에 대한 컨테이너 리포지토리
- macOS 이미지 전용 호스트
- IAM 사전 조건
- Systems Manager 에이전트 사전 조건

사전 조건을 충족한 후에는 다음 인터페이스 중 하나에서 EC2 Image Builder를 관리할 수 있습니다.

- EC2 Image Builder 콘솔
- 의 Image Builder 명령 AWS CLI
- EC2 Image Builder API 참조
- AWS SDKs 및 도구

Image Builder 서비스 연결 역할

EC2 Image Builder는 서비스 연결 역할을 사용하여 사용자를 대신하여 다른 AWS 서비스에 권한을 부여합니다. 서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS 관리 콘솔 AWS CLI, 또는 AWS API에서 첫 번째 Image Builder 리소스를 생성하면 Image Builder가 서비스 연결 역할을 생성합니다. Image Builder가 내 계정에서 생성하는 서비스 연결 역할에 관한 자세한 내용은 Image Builder의 IAM 서비스 연결 역할 사용 섹션을 참조하십시오.

구성 요구 사항

- Image Builder는 AWS PrivateLink(을)를 지원합니다. Image Builder의 VPC 엔드포인트 구성에 관한 자세한 내용은 Image Builder 및 AWS PrivateLink 인터페이스 VPC 엔드포인트 섹션을 참조하십시오.
- Image Builder가 컨테이너 이미지를 빌드하는 데 사용하는 인스턴스에는 Amazon S3 AWS CLI 에서를 다운로드하고 해당하는 경우 Docker Hub 리포지토리에서 기본 이미지를 다운로드할 수 있는 인터넷 액세스 권한이 있어야 합니다. Image Builder는 AWS CLI 를 사용하여 데이터로 저장되는 컨테이너 레시피에서 Dockerfile을 가져옵니다.
- Image Builder가 이미지를 빌드하고 테스트를 실행하는 데 사용하는 인스턴스에는 Systems Manager 서비스에 대한 액세스 권한이 있어야 합니다. 설치 요구 사항은 운영 체제에 따라 다릅니다.

  내 기본 이미지의 설치 요구 사항을 보려면 기본 이미지 운영 체제와 일치하는 탭을 선택하십시오.

  Linux

  Amazon EC2 Linux 인스턴스의 경우 Image Builder는 Systems Manager Agent가 아직 없는 경우 빌드 인스턴스에 Systems Manager 에이전트를 설치하고 이미지를 생성하기 전에 제거합니다.

  Windows

  Image Builder는 Amazon EC2 Windows Server 빌드 인스턴스에 Systems Manager Agent를 설치하지 않습니다. 기본 이미지가 Systems Manager Agent와 함께 사전 설치되어 있지 않은 경우 소스 이미지에서 인스턴스를 시작하고 인스턴스에 Systems Manager를 수동으로 설치한 다음 인스턴스에서 새 기본 이미지를 생성해야 합니다.

  Amazon EC2 Windows Server 인스턴스에 Systems Manager Agent를 수동으로 설치하려면 AWS Systems Manager 사용 설명서(User Guide)의 Windows 서버용 EC2 인스턴스에 Systems Manager Agent 수동 설치를 참조하십시오.

  macOS

  TBD - 사전 조건에 대한 정보가 필요합니다.

컨테이너 이미지 파이프라인에 대한 컨테이너 리포지토리

컨테이너 이미지 파이프라인의 경우 레시피는 대상 컨테이너 리포지토리에서 생성되고 저장되는 도커 이미지의 구성을 정의합니다. 도커 이미지의 컨테이너 레시피를 생성하기 전에 대상 리포지토리를 생성해야 합니다.

Image Builder는 Amazon ECR을 컨테이너 이미지의 대상 리포지토리로 사용합니다. Amazon ECR 리포지토리를 생성하려면 Amazon Elastic 컨테이너 레지스트리 사용 설명서의 리포지토리 생성에 설명된 단계를 따르십시오.

macOS 이미지 전용 호스트

Amazon EC2 Mac 인스턴스에는 메탈 인스턴스 유형에서 실행되는 전용 호스트가 필요합니다. 사용자 지정 macOS 이미지를 생성하기 전에 계정에 전용 호스트를 할당해야 합니다. Mac 인스턴스 및 macOS 운영 체제를 기본적으로 지원하는 인스턴스 유형 목록에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 Mac 인스턴스를 참조하세요.

전용 호스트를 생성하면 이미지의 인프라 구성 리소스에서 설정을 구성할 수 있습니다. 인프라 구성에는 이미지에서 시작하는 인스턴스가 이동해야 하는 호스트, 호스트 배치 그룹 또는 가용 영역을 지정할 수 있는 배치 속성이 포함됩니다.

IAM 사전 조건

인스턴스 프로파일에 연결하는 IAM 역할에는 이미지에 포함된 빌드 및 테스트 구성 요소를 실행할 수 있는 권한이 있어야 합니다. 다음 IAM 역할 정책은 인스턴스 프로파일과 관련된 IAM 역할에 연결되어야 합니다.

- EC2InstanceProfileForImageBuilder
- EC2InstanceProfileForImageBuilderECRContainerBuilds
- AmazonSSMManagedInstanceCore

로깅을 구성하는 경우 인프라 구성에 지정된 인스턴스 프로파일에 대상 버킷(`arn:aws:s3:::BucketName/*`)에 대한 `s3:PutObject` 권한이 있어야 합니다. 예시:

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

```
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::bucket-name/*"
    }
]
}
```

정책 연결

다음 단계는 IAM 정책을 IAM 역할에 연결하여 이전 권한을 부여하는 프로세스를 안내합니다.

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 목록을 EC2InstanceProfileForImageBuilder(으)로 필터링합니다.
4. 정책 옆의 글머리 기호를 선택하고 정책 작업 드롭다운 목록에서 연결을 선택합니다.
5. 정책을 연결할 IAM 역할의 이름을 선택합니다.
6. 정책 연결을 선택합니다.
7. EC2InstanceProfileForImageBuilderECRContainerBuilds 및 AmazonSSMManagedInstanceCore 정책에 대해 3~6단계를 반복합니다.

> **Note**
>
> Image Builder로 만든 이미지를 다른 계정에 복사하려면 모든 대상 계정에서 `EC2ImageBuilderDistributionCrossAccountRole` 역할을 생성하고 Ec2ImageBuilderCrossAccountDistributionAccess 정책 관리형 정책을 역할에 연결해야 합니다. 자세한 내용은 와 Image Builder 리소스 공유 AWS RAM 단원을 참조하십시오.

Systems Manager 에이전트 사전 조건

EC2 Image Builder는 시작한 EC2 인스턴스에서 AWS Systems Manager (Systems Manager) Agent를 실행하여 이미지를 빌드 및 테스트합니다. Image Builder는 Systems Manager Inventory를 사용하여 빌드 단계에서 사용된 인스턴스에 대한 추가 정보를 수집합니다. 이 정보에는 운영 체제(OS)이름 및 버전 뿐만 아니라 운영 체제에서 보고한 패키지 및 해당 버전 목록이 포함됩니다.

이 정보 수집을 거부하려면 원하는 환경에 적합한 방법을 선택합니다.

- Image Builder 콘솔-고급 메타데이터 수집 활성화 확인란의 선택을 취소합니다.
- AWS CLI---no-enhanced-image-metadata-enabled 옵션 지정
- Image Builder API 또는 SDK-enhancedImageMetadataEnabled 파라미터를 false로 설정합니다.

Image Builder는 이미지 빌드 및 테스트 워크플로의 일부로 빌드 및 테스트 인스턴스에 작업을 전송하는 데 RunCommand(을)를 사용합니다. 빌드 및 테스트 인스턴스에 작업을 보내는 RunCommand 사용을 거부할 수는 없습니다.

Image Builder 자습서를 사용하여 사용자 지정 이미지를 생성하는 방법을 알아보기

EC2 Image Builder를 사용하여 사용자 지정 이미지와 구성 요소를 구축하는 방법은 다양합니다. 자습서는 주요 Image Builder 개념에 대해 배우는 데 도움이 됩니다. 각 자습서에는 처음부터 따라할 수 있는 단계가 포함된 사용 사례가 나와 있습니다. 지침은 전체 프로세스를 학습하는 데 도움이 되도록 가능한 경우 기본값을 사용합니다. 자습서 중 하나를 사용한 후 이미지를 사용자 지정하는 더 많은 방법을 살펴볼 수 있습니다.

첫 번째 이미지 빌드

다음 자습서에서는 Image Builder 콘솔 마법사를 사용하여 첫 번째 이미지를 빌드하는 방법을 보여줍니다. 자습서가 끝나면 다음과 같은 Image Builder 리소스 세트를 생성하게 됩니다. 자습서의 마지막 단계는 생성한 리소스를 정리하는 것입니다.

- Amazon Machine Image(AMI)의 이미지 레시피 또는 컨테이너 이미지의 컨테이너 레시피입니다.
- 기본 설정이 있는 인프라 구성 리소스입니다.
- 출력을 소스 리전(콘솔 마법사를 실행하는 데 사용하는 리전의 계정)에 배포하는 기본 설정이 있는 배포 설정 리소스입니다.
- 나열된 리소스를 사용하여 기본 이미지 빌드 워크플로로 출력 이미지를 빌드하는 이미지 파이프라인입니다.
- 출력 AMI 또는 컨테이너 이미지입니다.

콘솔 마법사 자습서

- 파이프라인 마법사: AMI 생성
- 파이프라인 마법사: 컨테이너 이미지 생성

입력 파라미터를 사용하여 사용자 지정 구성 요소 만들기

다음 자습서에서는 입력 파라미터를 정의하는 사용자 지정 구성 요소를 생성한 다음 Image Builder 레시피에서 값을 설정하는 방법을 보여줍니다.

파라미터가 있는 사용자 지정 구성 요소

Image Builder에서 Systems Manager 파라미터 사용

다음 자습서에서는 AWS Systems Manager 파라미터 스토어 파라미터를 생성하고 이미지 레시피에 사용하는 방법을 보여줍니다.

레시피에서 기본 이미지 파라미터 사용

AMI 배포 설정에서 파라미터 스토어 파라미터를 사용하여 출력 이미지 ID와 사용자 지정 구성 요소에 저장할 수도 있습니다. 자세한 내용은 배포AMI 배포 구성 생성 및 업데이트는 단원을 참조하고 사용자 지정 구성 요소는 Systems Manager Parameter Store 파라미터 사용 단원을 참조하십시오.

자습서: Image Builder 콘솔 마법사에서 출력 AMI를 사용하여 이미지 파이프라인 생성

이 자습서에서는 이미지 파이프라인 생성 콘솔 마법사를 사용하여 사용자 지정된 EC2 Image Builder 이미지를 구축하고 유지 관리하기 위한 자동 파이프라인을 생성하는 방법을 안내합니다. 단계를 효율적으로 진행할 수 있도록 기본 설정이 있을 때는 기본 설정을 사용하고 선택 사항인 섹션은 생략했습니다.

이미지 파이프라인 워크플로 생성

- 1단계: 파이프라인 세부 정보 지정
- 2단계: 레시피 선택
- 3단계: 인프라 구성 정의 - 선택 사항
- 4단계: 배포 설정 정의 - 선택 사항
- 5단계: 검토
- 6단계: 정리

1단계: 파이프라인 세부 정보 지정

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 파이프라인 생성을 시작하려면 이미지 파이프라인 생성을 선택하십시오.
3. 일반 섹션에 파이프라인 이름(필수)을 입력합니다.

> **Tip**
>
> 향상된 메타데이터 수집은 기본적으로 활성화되어 있습니다. 구성 요소와 기본 이미지 간의 호환성을 보장하려면 이 기능을 계속 켜두십시오.

4. 빌드 일정 섹션에서 일정 옵션 기본값을 유지할 수 있습니다. 기본 일정에 표시된 시간대는 협정 세계 표준시(UTC)입니다. 협정 세계시(UTC)에 대한 자세한 내용과 시간대에 대한 오프셋을 찾으려면 전 세계 시간대 약어 목록을 참조하십시오.

   종속성 업데이트 설정의 경우, 종속성 업데이트가 있는 경우 예약된 시간에 파이프라인 실행 옵션을 선택하십시오. 이 설정을 사용하면 파이프라인이 빌드를 시작하기 전에 업데이트를 확인합니다. 업데이트가 없는 경우, 예약된 파이프라인 빌드를 건너뛰게 됩니다.

   > **Note**
   >
   > 파이프라인이 종속성 업데이트와 빌드를 예상대로 인식하도록 하려면, 기본 이미지와 구성 요소에 시맨틱 버전 관리(x.x.x)를 사용해야 합니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

5. 다음을 선택하여 다음 단계로 계속합니다.

2단계: 레시피 선택

1. Image Builder는 레시피 섹션의 기존 레시피 사용을 기본값으로 설정합니다. 처음인 경우, 새 레시피 생성 옵션을 선택하십시오.
2. 이미지 유형 섹션에서 Amazon Machine Image(AMI) 옵션을 선택하여 AMI를 생성하고 배포할 이미지 파이프라인을 생성합니다.
3. 일반 섹션에 다음과 같은 필수 상자를 입력합니다.

   - 이름 – 레시피 이름
   - 버전 — 레시피 버전(<major>.<minor>.<patch> 형식을 사용하십시오. 여기에서 major, minor, patch는 정수 값입니다.) 새 레시피는 일반적으로 `1.0.0`로 시작합니다.

4. 소스 이미지 섹션에서 이미지 선택, 이미지 운영 체제(OS) 및 이미지 출처를 기본값으로 유지합니다. 그러면 Amazon에서 관리하는 Linux AMIs 목록이 생성됩니다. 이 자습서에서는 `Amazon Linux 2 x86` 이미지를 선택합니다.

a. 이미지 이름 드롭다운에서 이미지를 선택합니다.

b. 자동 버전 관리 옵션(사용 가능한 최신 OS 버전 사용)의 기본값을 유지합니다.

> **Note**
>
> 이 설정을 사용하면 파이프라인이 기본 이미지에 대한 시맨틱 버전 관리를 사용하여 자동으로 예약된 작업에 대한 종속성 업데이트를 감지할 수 있습니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

5. 인스턴스 구성 섹션에서 Systems Manager 에이전트의 기본값을 유지합니다. 따라서 Image Builder는 빌드 및 테스트가 완료된 후에도 Systems Manager 에이전트를 유지하여 새 이미지에 Systems Manager 에이전트를 포함시킵니다.

   이 자습서에서는 사용자 데이터를 비워 두십시오. 다른 시간에 이 영역을 사용하여 명령을 제공하거나 빌드 인스턴스를 시작할 때 실행할 명령 스크립트를 사용할 수 있습니다. 그러나 이는 Systems Manager가 설치되었는지 확인하기 위해 Image Builder에서 추가했을 수 있는 모든 명령을 대체합니다. 사용할 때는 Systems Manager 에이전트가 기본 이미지에 사전 설치되어 있는지 또는 사용자 데이터에 설치를 포함해야 합니다.

6. 구성 요소 섹션에서 하나 이상의 빌드 구성 요소를 선택해야 합니다.

   빌드 구성 요소 패널의 빌드 구성 요소 추가를 선택하고 구성 요소 소유자 필터 목록에서 `Amazon managed`를 선택합니다. 그러면 사용 가능한 구성 요소를 찾아보고 필터링할 수 있는 콘솔 인터페이스 오른쪽에 선택 패널이 열립니다.

   이 자습서에서는 다음과 같이 Linux를 최신 보안 업데이트로 업데이트하는 구성 요소를 선택하십시오.

   a. 패널 상단에 있는 검색 창에 `update` 단어를 입력하여 결과를 필터링합니다.

   b. `update-linux` 빌드 구성 요소의 확인란을 선택합니다.

   c. 버전 관리 옵션의 기본값을 유지합니다(사용 가능한 최신 버전 사용).

   > **Note**
   >
   > 이 설정을 사용하면 파이프라인이 선택한 구성 요소에 대한 시맨틱 버전 관리를 사용하여, 자동으로 예약된 작업에 대한 종속성 업데이트를 감지할 수 있습니다. Image

Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

d. 레시피에 추가를 선택하여 레시피에 구성 요소를 추가합니다. 그러면 구성 요소 선택 패널이 닫힙니다.

e. 구성 요소 빌드 패널로 돌아가면 추가한 구성 요소가 표시됩니다.

7. 구성 요소 재정렬(선택 사항)

이미지에 포함할 구성 요소를 두 개 이상 선택한 경우, 드래그 앤 드롭 작업을 사용하여 빌드 프로세스 중에 실행해야 하는 순서로 구성 요소를 재배열할 수 있습니다.

> **Note**
>
> CIS 강화 구성 요소는 Image Builder 레시피의 표준 구성 요소 순서 지정 규칙을 따르지 않습니다. 출력 이미지에 대해 벤치마크 테스트가 실행되도록 하기 위해 CIS 하드닝 구성 요소는 항상 마지막에 실행됩니다.

a. 이전 단계를 반복하여 레시피에 `update-linux-kernel-5` 구성 요소를 추가합니다.

b. 방금 추가한 구성 요소에는 커널 버전에 대한 입력 파라미터가 있습니다. 버전 관리 옵션 또는 입력 파라미터 설정을 확장하려면 설정 이름 옆에 있는 화살표를 선택하면 됩니다. 선택한 모든 구성 요소의 모든 설정을 확장하려면 모두 확장 스위치를 끄고 켤 수 있습니다. 구성 요소의 입력 파라미터 사용 및 레시피에서 입력 파라미터를 설정하는 방법에 대한 자세한 내용은 자습서: 입력 파라미터를 사용하여 사용자 지정 구성 요소 만들기 섹션을 참조하십시오.

c. 구성 요소 중 하나를 선택하고 위 또는 아래로 드래그하여 구성 요소가 실행되는 순서를 변경합니다.

d. 구성 요소를 제거하려면, `update-linux-kernel-5` 구성 요소 상자의 오른쪽 X 상단에서 선택합니다.

이 단계를 반복하여 추가한 다른 구성 요소를 제거하고 `update-linux` 구성 요소만 선택한 상태로 둡니다.

8. 다음을 선택하여 다음 단계로 계속합니다.

3단계: 인프라 구성 정의 - 선택 사항

Image Builder는 계정에서 EC2 인스턴스를 시작하여 이미지를 사용자 지정하고 검증 테스트를 실행합니다. 인프라 구성 설정은 빌드 프로세스 AWS 계정 중에에서 실행할 인스턴스에 대한 인프라 세부 정보를 지정합니다.

인프라 구성 섹션에서 구성 옵션은 `Create infrastructure configuration using service defaults`(이)가 기본값으로 설정됩니다. 그러면 이미지를 구성하는 데 사용되는 EC2 빌드 및 테스트 인스턴스에 대한 IAM 역할 및 관련 인스턴스 프로파일이 생성됩니다. 인프라 구성 설정에 대한 자세한 내용은 EC2 Image Builder API 참조의 CreateInfrastructureConfiguration을 참조하십시오.

이 자습서에서는 기본 설정을 사용합니다.

> **Note**
>
> 프라이빗 VPC에 사용할 서브넷을 지정하려면 사용자 지정 인프라 구성을 만들거나 이미 만든 설정을 사용할 수 있습니다.

- 다음을 선택하여 다음 단계로 계속합니다.

4단계: 배포 설정 정의 - 선택 사항

배포 구성에는 출력 AMI 이름, 암호화에 대한 특정 리전 설정, 시작 권한 AWS 계정, 출력 AMI를 시작할 수 있는 조직 및 조직 단위(OUs) 및 라이선스 구성이 포함됩니다.

배포 설정 섹션의 구성 옵션은 기본적으로 `Create distribution settings using service defaults`(으)로 설정됩니다. 이 옵션은 출력 AMI를 현재 지역에 배포합니다. 배포 설정에 대한 자세한 내용은 Image Builder 배포 설정 관리 섹션을 참조하십시오.

이 자습서에서는 기본 설정을 사용합니다.

- 다음을 선택하여 다음 단계로 계속합니다.

5단계: 검토

검토 섹션에는 구성한 모든 설정이 표시됩니다. 특정 섹션의 정보를 편집하려면 단계 섹션의 오른쪽 상단에 있는 편집 버튼을 선택합니다. 예를 들어 파이프라인 이름을 변경하려면 1단계: 파이프라인 세부 정보 섹션의 오른쪽 상단에 있는 편집 버튼을 선택합니다.

1. 설정을 검토한 후 파이프라인 생성을 선택하여 파이프라인을 생성합니다.
2. 배포 설정, 인프라 구성, 새 레시피, 파이프라인에 대한 리소스가 생성되므로 페이지 상단에서 성공 또는 실패 메시지를 확인할 수 있습니다. 리소스 식별자를 포함하여 리소스의 세부 정보를 보려면 세부 정보 보기를 선택합니다.
3. 리소스의 세부 정보를 확인한 후 탐색 창에서 리소스 유형을 선택하여 다른 리소스에 대한 세부 정보를 볼 수 있습니다. 예를 들어 새 파이프라인의 세부 정보를 보려면 탐색 창에서 이미지 파이프라인을 선택하십시오. 빌드가 성공하면 새 파이프라인이 이미지 파이프라인 목록에 표시됩니다.

6단계: 정리

Image Builder 환경도 가정과 마찬가지로 정기적인 유지 관리가 필요합니다. 그래야 혼잡함 없이 필요한 것을 찾고 작업을 완료할 수 있습니다. 테스트용으로 만든 임시 리소스를 정기적으로 정리하십시오. 그렇지 않으면 해당 리소스를 잊어버리고 나중에는 해당 리소스가 어디에 사용되었는지 기억하지 못할 수 있습니다. 그때쯤이면 안전하게 제거할 수 있을지 확실하지 않을 수도 있습니다.

> **Tip**
>
> 리소스를 삭제할 때 종속성 오류를 방지하려면 다음 순서대로 리소스를 삭제하십시오.
>
> 1. 이미지 파이프라인
> 2. 이미지 레시피
> 3. 나머지 모든 리소스

이 자습서에서 생성한 리소스를 정리하려면, 다음 단계를 따르십시오.

파이프라인 삭제

1. 계정에서 생성된 빌드 파이프라인 목록을 보려면 탐색 창에서 이미지 파이프라인을 선택하십시오.
2. 삭제할 파이프라인을 선택하려면 파이프라인 이름 옆의 확인란을 선택합니다.
3. 이미지 파이프라인 패널 상단의 작업 메뉴에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 `Delete`을(를) 입력한 후 삭제를 선택합니다.

레시피 삭제

1. 계정에서 생성된 레시피 목록을 보려면 탐색 창에서 이미지 레시피를 선택합니다.
2. 삭제할 레시피를 선택하려면 레시피 이름 옆의 확인란을 선택합니다.
3. 이미지 레시피 패널 상단의 작업 메뉴에서 레시피 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 `Delete`을(를) 입력한 후 삭제를 선택합니다.

인프라 구성 삭제

1. 계정에서 생성된 인프라 구성 목록을 보려면 탐색 창에서 인프라 구성을 선택하십시오.
2. 구성 이름 옆의 확인란을 선택하여 삭제할 인프라 구성을 선택합니다.
3. 인프라 구성 패널 상단에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 `Delete`을(를) 입력한 후 삭제를 선택합니다.

배포 설정 삭제

1. 계정에서 생성된 배포 설정 목록을 보려면 탐색 창에서 배포 설정을 선택합니다.
2. 구성 이름 옆의 확인란을 선택하여 이 자습서를 위해 만든 배포 설정을 선택합니다.
3. 배포 설정 패널 상단에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 `Delete`을(를) 입력한 후 삭제를 선택합니다.

이미지 삭제

다음 단계에 따라 자습서 파이프라인에서 만든 이미지를 모두 삭제했는지 확인하십시오. 이 자습서에서는 빌드 일정에 따라 실행되는 파이프라인을 생성한 이후 충분한 시간이 경과해야 이미지를 생성할 가능성이 높습니다.

1. 계정에서 생성된 이미지 목록을 보려면 탐색 창에서 이미지를 선택합니다.
2. 제거하려는 이미지의 이미지 버전을 선택합니다. 그러면 이미지 빌드 버전 페이지가 열립니다.
3. 삭제하려는 이미지의 버전 옆 확인란을 선택합니다. 한 번에 이미지 버전을 여러 개 선택할 수 있습니다.
4. 이미지 빌드 버전 패널 상단에서 버전 삭제를 선택합니다.
5. 삭제를 확인하려면 박스에 `Delete`을(를) 입력한 후 삭제를 선택합니다.

자습서: Image Builder 콘솔 마법사에서 출력 Docker 컨테이너 이미지를 사용하여 이미지 파이프라인 생성

이 자습서에서는 이미지 파이프라인 생성 콘솔 마법사를 사용하여 사용자 지정된 EC2 Image Builder 도커 이미지를 구축하고 유지 관리하기 위한 자동화된 파이프라인을 생성하는 방법을 안내합니다. 단계를 효율적으로 진행할 수 있도록 기본 설정이 있을 때는 기본 설정을 사용하고 선택 사항인 섹션은 생략했습니다.

이미지 파이프라인 워크플로 생성

- 1단계: 파이프라인 세부 정보 지정
- 2단계: 레시피 선택
- 3단계: 인프라 구성 정의 - 선택 사항
- 4단계: 배포 설정 정의 - 선택 사항
- 5단계: 검토
- 6단계: 정리

1단계: 파이프라인 세부 정보 지정

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 파이프라인 생성을 시작하려면 이미지 파이프라인 생성을 선택하십시오.
3. 일반 섹션에 파이프라인 이름(필수)을 입력합니다.
4. 빌드 일정 섹션에서 일정 옵션 기본값을 유지할 수 있습니다. 기본 일정에 표시된 시간대는 협정 세계 표준시(UTC)입니다. 협정 세계시(UTC)에 대한 자세한 내용과 시간대에 대한 오프셋을 찾으려면 전 세계 시간대 약어 목록을 참조하십시오.

   종속성 업데이트 설정의 경우, 종속성 업데이트가 있는 경우 예약된 시간에 파이프라인 실행 옵션을 선택하십시오. 이 설정을 사용하면 파이프라인이 빌드를 시작하기 전에 업데이트를 확인합니다. 업데이트가 없는 경우, 예약된 파이프라인 빌드를 건너뛰게 됩니다.

   > **Note**
   >
   > 파이프라인이 종속성 업데이트와 빌드를 예상대로 인식하도록 하려면, 기본 이미지와 구성 요소에 시맨틱 버전 관리(x.x.x)를 사용해야 합니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

5. 다음을 선택하여 다음 단계로 계속합니다.

2단계: 레시피 선택

1. Image Builder는 레시피 섹션의 기존 레시피 사용을 기본값으로 설정합니다. 처음인 경우, 새 레시피 생성 옵션을 선택하십시오.
2. 이미지 유형 섹션에서 도커 이미지 옵션을 선택하여 도커 이미지를 생성하고 이를 대상 리전의 Amazon ECR 리포지토리에 배포하는 컨테이너 파이프라인을 생성합니다.
3. 일반 섹션에 다음과 같은 필수 상자를 입력합니다.

   - 이름 – 레시피 이름
   - 버전 — 레시피 버전(<major>.<minor>.<patch> 형식을 사용하십시오. 여기에서 major, minor, patch는 정수 값입니다.) 새 레시피는 일반적으로 `1.0.0`로 시작합니다.

4. 소스 이미지 섹션에서 이미지 선택, 이미지 운영 체제(OS) 및 이미지 출처를 기본값으로 유지합니다. 그러면 Amazon에서 관리하는 Amazon Linux 2 컨테이너 이미지 목록이 생성되며 기본 이미지로 선택할 수 있습니다.

   a. 이미지 이름 드롭다운에서 이미지를 선택합니다.

   b. 자동 버전 관리 옵션(사용 가능한 최신 OS 버전 사용)의 기본값을 유지합니다.

   > **Note**
   >
   > 이 설정을 사용하면 파이프라인이 기본 이미지에 대한 시맨틱 버전 관리를 사용하여 자동으로 예약된 작업에 대한 종속성 업데이트를 감지할 수 있습니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

5. 구성 요소 섹션에서 하나 이상의 빌드 구성 요소를 선택해야 합니다.

   빌드 구성 요소 – Amazon Linux 패널에서 페이지에 나열된 구성 요소를 탐색할 수 있습니다. 오른쪽 상단의 페이지 매김 제어를 사용하여 기본 이미지 OS에 사용할 수 있는 추가 구성 요소를 탐색할 수 있습니다. 또한 구성 요소 관리자를 사용하여 특정 구성 요소를 검색하거나 자체 빌드 구성 요소를 만들 수 있습니다.

   이 자습서에서는 다음과 같이 Linux를 최신 보안 업데이트로 업데이트하는 구성 요소를 선택하십시오.

a. 패널 상단에 있는 검색 창에 update 단어를 입력하여 결과를 필터링합니다.

b. update-linux 빌드 구성 요소의 확인란을 선택합니다.

c. 아래로 스크롤하여 선택한 구성 요소 목록의 오른쪽 상단에서 모두 확장을 선택합니다.

d. 버전 관리 옵션(사용 가능한 최신 OS 버전 사용)의 기본값을 유지합니다.

> **Note**
>
> 이 설정을 사용하면 파이프라인이 선택한 구성 요소에 대한 시맨틱 버전 관리를 사용하여, 자동으로 예약된 작업에 대한 종속성 업데이트를 감지할 수 있습니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

입력 파라미터가 있는 구성 요소를 선택한 경우 이 영역에 파라미터도 표시됩니다. 이 자습서에서는 파라미터를 다루지 않습니다. 구성 요소의 입력 파라미터 사용 및 레시피에서 입력 파라미터를 설정하는 방법에 대한 자세한 내용은 자습서: 입력 파라미터를 사용하여 사용자 지정 구성 요소 만들기 섹션을 참조하십시오.

구성 요소 재정렬(선택 사항)

이미지에 포함할 구성 요소를 두 개 이상 선택한 경우 드래그 앤 드롭 작업을 사용하여 빌드 프로세스 중에 실행되어야 하는 순서로 구성 요소를 재정렬할 수 있습니다.

> **Note**
>
> CIS 강화 구성 요소는 Image Builder 레시피의 표준 구성 요소 순서 지정 규칙을 따르지 않습니다. 출력 이미지에 대해 벤치마크 테스트가 실행되도록 하기 위해 CIS 강화 구성 요소는 항상 마지막에 실행됩니다.

1. 다시 위로 스크롤하여 사용 가능한 구성 요소 목록을 확인하십시오.
2. update-linux-kernel-mainline 빌드 구성 요소(또는 원하는 다른 구성 요소)의 확인란을 선택합니다.
3. 선택한 구성 요소 목록으로 스크롤하여 결과가 두 개 이상 있는지 확인합니다.

4. 새로 추가된 구성 요소의 버전이 확장되지 않을 수 있습니다. 버전 관리 옵션을 확장하려면 버전 관리 옵션 옆의 화살표를 선택하거나 모두 확장을 토글해서 선택한 모든 구성 요소의 버전 관리를 확장할 수 있습니다.
5. 구성 요소 중 하나를 선택하고 위 또는 아래로 드래그하여 구성 요소가 실행되는 순서를 변경합니다.
6. 구성 요소를 제거하려면, `update-linux-kernel-mainline` 구성 요소 상자의 오른쪽 X 상단에서 선택합니다.
7. 이전 단계를 반복하여 추가했을 수 있는 다른 구성 요소를 모두 제거하고 해당 `update-linux` 구성 요소만 선택합니다.

6. Dockerfile 템플릿 섹션에서 예제 사용 옵션을 선택합니다. 콘텐츠 패널에서 Image Builder가 내 컨테이너 이미지 레시피에 따라 빌드 정보 또는 스크립트를 배치하는 상황적 변수가 있는 것을 볼 수 있습니다.

   기본적으로 Image Builder는 Dockerfile에서 다음과 같은 컨텍스트 변수를 사용합니다.

   parentImage(필수)

   빌드 시 이 변수는 레시피의 기본 이미지로 확인됩니다.

   예시

   ```
   FROM
   {{{ imagebuilder:parentImage }}}
   ```

   환경(구성 요소가 지정된 경우 필수)

   이 변수는 구성 요소를 실행하는 스크립트로 확인됩니다.

   예시

   ```
   {{{ imagebuilder:environments }}}
   ```

   구성 요소(선택 사항)

   Image Builder는 컨테이너 레시피에 포함된 구성 요소에 대한 빌드 및 테스트 구성 요소 스크립트를 확인합니다. 이 변수는 환경 변수 뒤에 Dockerfile의 아무 곳에나 배치할 수 있습니다.

예시

```
{{{ imagebuilder:components }}}
```

7. 대상 리포지토리 섹션에서 이 자습서의 사전 조건으로 생성한 Amazon ECR 리포지토리의 이름을 지정합니다. 이 리포지토리는 파이프라인이 실행되는 리전(리전 1)에서 배포 구성의 기본 설정으로 사용됩니다.

> **Note**
>
> 배포하기 전에 대상 리포지토리가 모든 대상 리전의 Amazon ECR에 있어야 합니다.

8. 다음을 선택하여 다음 단계로 계속합니다.

3단계: 인프라 구성 정의 - 선택 사항

Image Builder는 계정에서 EC2 인스턴스를 시작하여 이미지를 사용자 지정하고 검증 테스트를 실행합니다. 인프라 구성 설정은 빌드 프로세스 AWS 계정 중에에서 실행할 인스턴스에 대한 인프라 세부 정보를 지정합니다.

인프라 구성 섹션에서 구성 옵션은 `Create infrastructure configuration using service defaults`(이)가 기본값으로 설정됩니다. 이렇게 빌드 인스턴스가 컨테이너 이미지를 구성하는 데 사용하는 IAM 역할 및 관련 인스턴스 프로파일이 생성됩니다. 고유의 사용자 지정 인프라 구성을 생성하거나 이미 생성한 설정을 사용할 수도 있습니다. 인프라 구성 설정에 대한 자세한 내용은 EC2 Image Builder API 참조의 CreateInfrastructureConfiguration을 참조하십시오.

이 자습서에서는 기본 설정을 사용합니다.

- 다음을 선택하여 다음 단계로 계속합니다.

4단계: 배포 설정 정의 - 선택 사항

배포 설정은 대상 리전과 대상 Amazon ECR 리포지토리 이름으로 구성됩니다. 출력 Docker 이미지는 각 리전의 지정된 Amazon ECR 리포지토리에 배포됩니다.

배포 설정 섹션의 구성 옵션은 기본적으로 `Create distribution settings using service defaults`(으)로 설정됩니다. 이 옵션은 파이프라인이 실행되는 리전(리전 1)의 컨테이너 레시피에

지정된 Amazon ECR 리포지토리에 출력 도커 이미지를 배포합니다. `Create new distribution settings`(을)를 선택하면 현재 리전의 ECR 리포지토리를 재정의하고 배포할 리전을 더 추가할 수 있습니다.

이 자습서에서는 기본 설정을 사용합니다.

- 다음을 선택하여 다음 단계로 계속합니다.

5단계: 검토

검토 섹션에는 구성한 모든 설정이 표시됩니다. 특정 섹션의 정보를 편집하려면 단계 섹션의 오른쪽 상단에 있는 편집 버튼을 선택합니다. 예를 들어 파이프라인 이름을 변경하려면 1단계: 파이프라인 세부 정보 섹션의 오른쪽 상단에 있는 편집 버튼을 선택합니다.

1. 설정을 검토한 후 파이프라인 생성을 선택하여 파이프라인을 생성합니다.
2. 배포 설정, 인프라 구성, 새 레시피, 파이프라인에 대한 리소스가 생성되므로 페이지 상단에서 성공 또는 실패 메시지를 확인할 수 있습니다. 리소스 식별자를 포함하여 리소스의 세부 정보를 보려면 세부 정보 보기를 선택합니다.
3. 리소스의 세부 정보를 확인한 후 탐색 창에서 리소스 유형을 선택하여 다른 리소스에 대한 세부 정보를 볼 수 있습니다. 예를 들어 새 파이프라인의 세부 정보를 보려면 탐색 창에서 이미지 파이프라인을 선택하십시오. 빌드가 성공하면 새 파이프라인이 이미지 파이프라인 목록에 표시됩니다.

6단계: 정리

Image Builder 환경도 가정과 마찬가지로 정기적인 유지 관리가 필요합니다. 그래야 혼잡함 없이 필요한 것을 찾고 작업을 완료할 수 있습니다. 테스트용으로 만든 임시 리소스를 정기적으로 정리하십시오. 그렇지 않으면 해당 리소스를 잊어버리고 나중에는 해당 리소스가 어디에 사용되었는지 기억하지 못할 수 있습니다. 그때쯤이면 안전하게 제거할 수 있을지 확실하지 않을 수도 있습니다.

> **Tip**
>
> 리소스를 삭제할 때 종속성 오류를 방지하려면 다음 순서대로 리소스를 삭제하십시오.
>
> 1. 이미지 파이프라인
> 2. 이미지 레시피
> 3. 나머지 모든 리소스

이 자습서에서 생성한 리소스를 정리하려면, 다음 단계를 따르십시오.

파이프라인 삭제

1. 계정에서 생성된 빌드 파이프라인 목록을 보려면 탐색 창에서 이미지 파이프라인을 선택하십시오.
2. 삭제할 파이프라인을 선택하려면 파이프라인 이름 옆의 확인란을 선택합니다.
3. 이미지 파이프라인 패널 상단의 작업 메뉴에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

컨테이너 레시피 삭제

1. 계정에서 생성된 컨테이너 레시피 목록을 보려면 탐색 창에서 컨테이너 레시피를 선택합니다.
2. 삭제할 레시피를 선택하려면 레시피 이름 옆의 확인란을 선택합니다.
3. 컨테이너 레시피 패널 상단의 작업 메뉴에서 레시피 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

인프라 구성 삭제

1. 계정에서 생성된 인프라 구성 목록을 보려면 탐색 창에서 인프라 구성을 선택하십시오.
2. 구성 이름 옆의 확인란을 선택하여 삭제할 인프라 구성을 선택합니다.
3. 인프라 구성 패널 상단에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

배포 설정 삭제

1. 계정에서 생성된 배포 설정 목록을 보려면 탐색 창에서 배포 설정을 선택합니다.
2. 구성 이름 옆의 확인란을 선택하여 이 자습서를 위해 만든 배포 설정을 선택합니다.
3. 배포 설정 패널 상단에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

이미지 삭제

다음 단계에 따라 자습서 파이프라인에서 만든 이미지를 모두 삭제했는지 확인하십시오. 이 자습서에서는 빌드 일정에 따라 실행되는 파이프라인을 생성한 이후 충분한 시간이 경과해야 이미지를 생성할 가능성이 높습니다.

1. 계정에서 생성된 이미지 목록을 보려면 탐색 창에서 이미지를 선택합니다.
2. 제거하려는 이미지의 이미지 버전을 선택합니다. 그러면 이미지 빌드 버전 페이지가 열립니다.
3. 삭제하려는 이미지의 버전 옆 확인란을 선택합니다. 한 번에 이미지 버전을 여러 개 선택할 수 있습니다.
4. 이미지 빌드 버전 패널 상단에서 버전 삭제를 선택합니다.
5. 삭제를 확인하려면 박스에 `Delete`을(를) 입력한 후 삭제를 선택합니다.

자습서: 입력 파라미터를 사용하여 사용자 지정 구성 요소 만들기

EC2 Image Builder 콘솔, 또는 Image Builder API AWS CLI또는 SDKs. 이 섹션에서는 구성 요소에서 파라미터를 생성 및 사용하고 Image Builder 콘솔을 통해 구성 요소 파라미터를 설정하고 런타임 시 AWS CLI 명령을 설정하는 방법을 다룹니다.

> **Important**
>
> 구성 요소 파라미터는 일반 텍스트 값이며 AWS CloudTrail에 기록됩니다. AWS Secrets Manager 또는 AWS Systems Manager 파라미터 스토어를 사용하여 보안 암호를 저장하는 것이 좋습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager란 무엇입니까?를 참조하십시오. AWS Systems Manager Parameter Store에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Parameter Store 섹션을 참조하십시오.

YAML 구성 요소 문서의 파라미터 사용

구성 요소를 빌드하려면, YAML 또는 JSON 애플리케이션 구성 요소 문서를 제공하십시오. 문서에는 이미지에 대한 사용자 지정을 제공하기 위해 정의한 단계에서 실행되는 코드가 포함되어 있습니다. 구성 요소를 참조하는 레시피는 파라미터를 설정하여 런타임에 값을 사용자 지정할 수 있으며, 파라미터가 특정 값으로 설정되지 않은 경우 적용되는 기본값을 사용할 수 있습니다.

입력 파라미터를 사용하여 구성 요소 문서 만들기

이 단원에서는 YAML 구성 요소 문서에서 입력 파라미터를 정의하고 사용하는 방법을 설명합니다.

Image Builder 빌드 또는 테스트 인스턴스에서 파라미터를 사용하고 명령을 실행하는 YAML 애플리케이션 구성 요소 문서를 만들려면, 이미지 운영 체제와 일치하는 단계를 따르십시오.

Linux

YAML 구성 요소 문서 생성

파일 편집 도구를 사용하여 구성 요소 문서 파일을 생성합니다. 설명서 예제에서는 다음 콘텐츠가 포함된 *hello-world-test.yaml*이라는 파일을 사용합니다.

```
# Document Start
#
name: "HelloWorldTestingDocument-Linux"
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
```

```
        inputs:
          commands:
            - echo "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> 이 온라인 YAML 검사기와 같은 도구나 코드 환경의 YAML Lint 확장을 사용하여 YAML이 제대로 구성되어 있는지 확인하십시오.

Windows

YAML 구성 요소 문서 생성

파일 편집 도구를 사용하여 구성 요소 문서 파일을 생성합니다. 설명서 예제에서는 다음 콘텐츠가 포함된 *hello-world-test.yaml*이라는 파일을 사용합니다.

```
# Document Start
#
name: "HelloWorldTestingDocument-Windows"
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: validate
    steps:
      - name: HelloWorldStep
```

```
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> 이 온라인 YAML 검사기와 같은 도구나 코드 환경의 YAML Lint 확장을 사용하여 YAML이 제대로 구성되어 있는지 확인하십시오.

macOS

YAML 구성 요소 문서 생성

파일 편집 도구를 사용하여 구성 요소 문서 파일을 생성합니다. 설명서 예제에서는 다음 콘텐츠가 포함된 *hello-world-test.yaml*이라는 파일을 사용합니다.

```
# Document Start
#
name: "HelloWorldTestingDocument-macOS"
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
```

```
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> 이 온라인 YAML 검사기와 같은 도구나 코드 환경의 YAML Lint 확장을 사용하여 YAML이 제대로 구성되어 있는지 확인하십시오.

AWSTOE 애플리케이션 구성 요소 문서의 단계 및 구문에 대한 자세한 내용은 AWSTOE에서 문서 사용을 참조하십시오. 파라미터 및 해당 요구 사항에 대한 자세한 내용은 AWSTOE에서 변수 정의 및 참조 페이지의 파라미터 섹션을 참조하십시오.

YAML 구성 요소 문서에서 구성 요소 생성

구성 AWSTOE 요소를 생성하는 데 어떤 방법을 사용하든 YAML 애플리케이션 구성 요소 문서는 항상 기준으로 필요합니다.

- Image Builder 콘솔을 사용하여 YAML 문서에서 직접 구성 요소를 만들려면 콘솔에서 사용자 지정 구성 요소 생성 섹션을 참조하십시오.
- Image Builder create-component 명령을 사용하여 명령줄에서 구성 요소를 생성하려면 에서 사용자 지정 구성 요소 생성 AWS CLI 섹션을 참조하세요. 이 예제의 YAML 문서 이름을 Hello World YAML 문서(*hello-world-test.yaml*)의 이름으로 바꾸십시오.

콘솔에서 Image Builder 레시피에 구성 요소 파라미터 설정

구성 요소 파라미터 설정은 이미지 레시피와 컨테이너 레시피에서 동일하게 작동합니다. 새 레시피 또는 새 버전의 레시피를 만들 때는 빌드 구성 요소 및 테스트 구성 요소 목록에서 포함할 구성 요소를 선택합니다. 구성 요소 목록에는 이미지용으로 선택한 기본 운영 체제에 적용할 수 있는 구성 요소가 포함됩니다.

구성 요소를 선택하면 구성 요소 목록 바로 아래의 선택된 구성 요소 섹션에 해당 구성 요소가 표시됩니다. 선택한 각 구성 요소에 대한 구성 옵션이 표시됩니다. 구성 요소에 입력 파라미터가 정의되어 있는 경우 입력 파라미터라는 확장 가능한 섹션으로 표시됩니다.

구성 요소에 정의된 각 파라미터에 대해 다음과 같은 파라미터 설정이 표시됩니다.

- 파라미터 이름(편집 불가) – 파라미터 이름입니다.
- 설명(편집 불가) – 파라미터 설명입니다.
- 유형(편집 불가) – 파라미터 값의 데이터 유형입니다.
- 값 – 파라미터의 값입니다. 이 레시피에서 이 구성 요소를 처음 사용하고 입력 파라미터에 대해 기본값이 정의된 경우 기본값이 값 상자에 회색으로 표시된 텍스트와 함께 나타납니다. 값을 입력하지 않으면 Image Builder에서 기본값을 사용합니다.

레시피에서 기본 이미지 파라미터 사용

이미지 사용자 지정을 위한 레시피를 생성할 때 시작하는 기본 이미지를 식별하는 몇 가지 방법이 있습니다. 기본 이미지의 Amazon Machine Image(AMI) ID를 지정하고 해당 기본 이미지가 업데이트되면 해당 AMI ID가 변경될 수 있으며 레시피를 일치하도록 업데이트해야 합니다.

기본 이미지 ID가 변경될 때마다 레시피를 변경하는 대신 AWS Systems Manager Parameter Store 파라미터(SSM 파라미터)를 정의하여 기본 이미지 AMI ID의 값을 저장한 다음 파라미터를 사용하여 레시피에 기본 이미지를 지정할 수 있습니다. AWS 관리형 AMIs 경우 최신 버전에 퍼블릭 파라미터를 사용할 수 있습니다.

이 자습서에서는 AMI ID 파라미터를 생성하고 이미지 레시피에서 사용하는 프로세스를 안내합니다. 이 자습서의 Image Builder 단계는 콘솔 기반입니다.

내용

- 1단계: 파라미터 스토어 파라미터 찾기 또는 생성
- 2단계: IAM 권한 구성
- 3단계: 파라미터를 사용하는 이미지 레시피 생성

1단계: 파라미터 스토어 파라미터 찾기 또는 생성

이 단계의 프로세스는 기본 이미지에 지정하는 AMI 유형에 따라 다릅니다. AWS 관리형 AMIs 경우 현재 버전을 참조하는 퍼블릭 파라미터를 사용할 수 있습니다. 일부 파라미터는 전혀 사용하지 못할 수 있습니다 AWS 리전.

시작하려면 AMI에 해당하는 탭을 엽니다.

AWS managed AMI

기본 이미지가 AWS 관리형 AMI인 경우 자체 파라미터를 생성하는 대신 퍼블릭 파라미터를 사용하여 AMI ID를 지정할 수 있습니다. AMI의 퍼블릭 파라미터를 찾으려면 AWS Systems Manager 사용 설명서의 퍼블릭 파라미터 검색을 참조하세요.

Custom AMI

AMI ID 파라미터를 생성하려면 콘솔 AWS CLI또는 PowerShell을 사용하여 Systems Manager에서 파라미터 스토어 파라미터 생성 지침을 따릅니다. 파라미터 값이 AMI ID인지 확인하려면 다음 값을 입력합니다.

파라미터 계층: `Standard`

유형: `String`

데이터 유형:를 선택합니다`aws:ec2:image`. 이 유형을 지정하면 시스템은 입력된 값을 검증하여 AMI ID인지 확인합니다.

값: 유효한 AMI ID(예: ***`ami-1234567890abcdef1`***)를 입력합니다.

2단계: IAM 권한 구성

퍼블릭이든 프라이빗이든 Systems Manager Parameter Store 파라미터(SSM 파라미터)를 사용하려면 Image Builder 실행 역할에 다음 Systems Manager Parameter Store 작업을 지정하고 파라미터를 리소스로 나열해야 합니다.

- `ssm:GetParameter` -이 작업을 통해 SSM 파라미터를 사용하여 레시피의 기본 이미지를 지정할 수 있습니다.
- `ssm:PutParameter` -이 작업을 사용하면 배포 중에 출력 AMI ID를 SSM 파라미터에 저장할 수 있습니다. 정책 정의는 동일하게 보이지만이 자습서에서는 예제 정책에 put 작업을 포함하지 않습니다.

사용자 지정 구성 요소에서 SSM 파라미터를 사용하려면 인스턴스 프로파일 역할`ssm:GetParameter`에서를 대신 지정해야 합니다. 자세한 내용은 Systems Manager Parameter Store 파라미터 사용 단원을 참조하십시오.

파이프라인을 생성하거나에서 create-image 명령을 사용하는 경우 Image Builder 실행 역할을 하나만 지정할 AWS CLI수 있습니다. Image Builder 워크플로 실행 역할을 정의한 경우 해당 역할에 파라미터 권한을 추가합니다. 그렇지 않으면 SSM 파라미터에 필요한 권한이 포함된 새 사용자 지정 역할을 생성합니다.

1. 사용자 지정 역할 생성(선택 사항)

   Image Builder 권한에 대해 정의된 사용자 지정 역할이 이미 있는 경우이 단계를 건너뛸 수 있습니다.

   AWS Identity and Access Management 사용 설명서의 역할 생성 프로세스에 따라 AWS 서비스에 권한을 위임합니다.
2. 사용자 지정 역할에 권한 추가

   사용자 지정 역할에 SSM 파라미터 권한을 추가하려면 AWS Identity and Access Management 사용 설명서의 역할 프로세스에 대한 권한 정책 업데이트를 따르세요.

   다음 정책 예제는 계정에 생성된 파라미터가 있는 `ssm:GetParameter` 작업을 보여줍니다.

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

```
    "Action": "ssm:GetParameter",
    "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
   }
  ]
}
```

퍼블릭 파라미터 리소스에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AMI 퍼블릭 파라미터 호출을 참조하세요.

3단계: 파라미터를 사용하는 이미지 레시피 생성

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 이미지 레시피를 선택한 다음 목록 페이지에서 이미지 레시피 생성을 선택합니다.
3. 다음과 같이 기본 이미지 섹션을 작성합니다.
   a. 사용자 지정 AMI 사용 옵션을 선택합니다. 그러면 AMI ID 또는 AMI ID가 포함된 SSM 파라미터를 입력할 수 있는 추가 필드가 표시됩니다.
   b. SSM 파라미터 옵션을 선택합니다.
   c. SSM 파라미터 필드에 1단계에서 생성한 파라미터의 파라미터 이름 또는 Amazon 리소스 이름(ARN)을 입력합니다. 이름을 입력하면 콘솔에 접두사가 없습니다.
4. 필요에 따라 나머지 레시피 구성을 완료합니다.

> **Note**
>
> 와 같은 다른 인터페이스를 통해 상위 이미지를 설정하는 경우 파라미터 이름 AWS CLI에 접두사가 있어야 합니다ssm:(예: `ssm:/ImageBuilder-Tutorial/BaseAMI`.

구성 요소를 사용하여 Image Builder 이미지 사용자 지정

Image Builder는 AWS Task Orchestrator and Executor (AWSTOE) 구성 요소 관리 애플리케이션을 사용하여 복잡한 워크플로를 오케스트레이션합니다. AWSTOE 애플리케이션과 함께 작동하는 빌드 및 테스트 구성 요소는 이미지를 사용자 지정하거나 테스트하기 위한 스크립트를 정의하는 YAML 문서를 기반으로 합니다. AMI 이미지의 경우 Image Builder는 Amazon EC2 빌드 및 테스트 인스턴스에 구성 요소와 AWSTOE 구성 요소 관리 애플리케이션을 설치합니다. 컨테이너 이미지의 경우 구성 요소 및 AWSTOE 구성 요소 관리 애플리케이션은 실행 중인 컨테이너 내부에 설치됩니다.

Image Builder는 AWSTOE 를 사용하여 모든 온인스턴스 활동을 수행합니다. Image Builder 명령을 실행하거나 Image Builder 콘솔을 사용할 AWSTOE 때와 상호 작용하는 데 필요한 추가 설정은 없습니다.

> **Note**
>
> Amazon에서 관리하는 구성 요소의 지원 수명이 다하면 해당 구성 요소는 더 이상 유지 관리되지 않습니다. 이 문제가 발생하기 약 4주 전에 해당 구성 요소를 사용하는 모든 계정은 알림과 AWS Health Dashboard의 해당 계정에서 영향을 받는 레시피의 목록을 수신합니다. 자세한 내용은 AWS Health 사용 설명서를 AWS Health참조하세요.

새 이미지를 빌드하기 위한 워크플로 단계

새 이미지를 빌드하기 위한 Image Builder 워크플로에는 다음과 같은 두 단계가 있습니다.

1. 빌드 단계(스냅샷 이전)-빌드 단계에서 기본 이미지를 실행하는 Amazon EC2 빌드 인스턴스를 변경하여 새 이미지의 베이스라인을 생성합니다. 예를 들어, 레시피에 애플리케이션을 설치하거나 운영 체제 방화벽 설정을 수정하는 구성 요소가 포함될 수 있습니다.

   빌드 단계에서 실행되는 구성 요소 단계는 다음과 같습니다.

   - 빌드
   - 검증

   이 단계가 성공적으로 완료되면 Image Builder는 테스트 단계 및 이후 단계에서 사용할 스냅샷 또는 컨테이너 이미지를 생성합니다.
2. 테스트 단계(스냅샷 이후)-테스트 단계에서 AMI를 생성하는 이미지와 컨테이너 이미지 사이에 약간의 차이가 있습니다. AMI 워크플로의 경우 Image Builder는 빌드 단계의 마지막 단계로 생성한 스냅

샷에서 EC2 인스턴스를 시작합니다. 새 인스턴스에서 테스트를 실행하여 설정을 검증하고 인스턴스가 예상대로 작동하는지 확인합니다. 컨테이너 워크플로의 경우 테스트는 구축에 사용된 것과 동일한 인스턴스에서 실행됩니다.

이미지 빌드 테스트 단계에서 레시피에 포함된 모든 구성 요소에 대해 다음 구성 요소 문서 단계가 실행됩니다.

- 테스트

이 구성 요소 단계는 빌드 및 테스트 구성 요소 유형 모두에 적용됩니다. 이 단계가 성공적으로 완료되면 Image Builder는 스냅샷 또는 컨테이너 이미지에서 최종 이미지를 생성하고 배포할 수 있습니다.

> **Note**
>
> AWSTOE 애플리케이션 프레임워크를 사용하면 구성 요소 문서에서 여러 단계를 정의할 수 있지만 Image Builder에는 실행하는 단계와 실행하는 단계에 대한 엄격한 규칙이 있습니다. 구성 요소를 이미지 빌드 단계에서 실행하려면 구성 요소 문서에서 `build` 또는 `validate` 단계 중 하나 이상을 정의해야 합니다. 구성 요소를 이미지 테스트 단계에서 실행하려면 구성 요소 문서에서 다른 단계가 아닌 `test` 단계를 정의해야 합니다.
> Image Builder는 단계들을 독립적으로 실행하므로 구성 요소 문서의 체인 참조는 단계의 경계를 넘을 수 없습니다. 빌드 단계에서 실행되는 단계에서 테스트 단계에서 실행되는 단계로 값을 체인할 수 없습니다. 하지만 입력 파라미터를 의도한 대상에 정의하고 명령줄을 통해 값을 전달할 수는 있습니다. Image Builder 레시피의 구성 요소 파라미터 설정에 대한 자세한 내용은 자습서: 입력 파라미터를 사용하여 사용자 지정 구성 요소 만들기(을)를 참조합니다.

빌드 또는 테스트 인스턴스의 문제 해결을 지원하기 위해 입력 문서와 로그 파일이 포함된 로그 폴더를 AWSTOE 생성하여 구성 요소가 실행될 때마다 발생하는 상황을 추적합니다. 파이프라인 구성에서 Amazon S3 버킷을 구성한 경우 로그도 여기에 기록됩니다. YAML 문서 및 로그 출력에 대한 자세한 내용은 사용자 지정 AWSTOE 구성 요소에 구성 요소 문서 프레임워크 사용(을)를 참조합니다.

> **Tip**
>
> 추적해야 할 구성 요소가 많을 때 태그를 지정하여 사용하면 지정한 태그에 따라 특정 구성 요소나 버전을 식별할 수 있습니다. 에서 Image Builder 명령을 사용하여 리소스에 태그를 지정하는 방법에 대한 자세한 내용은이 가이드의 리소스 태깅 섹션을 AWS CLI참조하세요.

이 섹션에서는 Image Builder 콘솔 또는 AWS CLI의 명령을 사용하여 구성 요소를 나열하고, 보고, 생성하고, 가져오는 방법을 설명합니다.

주제

- 구성 요소 세부 정보 나열 및 보기
- AWS Marketplace 구성 요소를 사용하여 이미지 사용자 지정
- 관리형 구성 요소를 사용하여 Image Builder 이미지 사용자 지정
- Image Builder 이미지에 대한 사용자 지정 구성 요소 개발
- Image Builder가 AWS Task Orchestrator and Executor 애플리케이션을 사용하여 구성 요소를 관리하는 방법

구성 요소 세부 정보 나열 및 보기

이 섹션에서는 EC2 Image Builder 레시피에서 사용하는 구성 요소에 대한 정보를 찾고 세부 정보를 보는 방법을 설명합니다.

구성 요소 세부 정보

- Image Builder 구성 요소 나열
- 에서 구성 요소 빌드 버전 나열 AWS CLI
- 에서 구성 요소 세부 정보 가져오기 AWS CLI
- 에서 구성 요소 정책 세부 정보 가져오기 AWS CLI

Image Builder 구성 요소 나열

다음 방법 중 하나를 사용하여 Image Builder 구성 요소를 나열하고 필터링할 수 있습니다.

AWS Management Console

에 구성 요소 목록을 표시하려면 다음 단계를 AWS Management Console따릅니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 구성 요소를 선택합니다. 기본적으로 Image Builder는 사용자 계정이 소유한 구성 요소 목록을 표시합니다.
3. 구성 요소 소유권을 기준으로 필터링할 수도 있습니다. 소유하지는 않지만 액세스할 수 있는 구성 요소를 보려면 소유자 유형 드롭다운 목록을 확장하고 값 중 하나를 선택하십시오. 소유자 유형 목록은 검색 표시줄의 검색 텍스트 상자 옆에 있습니다. 다음 값을 사용할 수 있습니다.

- AWS Marketplace - AWS Marketplace 제품 구독과 직접 연결된 구성 요소입니다.
- 빠른 시작(Amazon 관리) - Amazon이 생성하고 유지 관리하는 공개적으로 사용 가능한 구성 요소입니다.
- 본인 소유 - 사용자가 만든 구성 요소입니다. 이는 기본 선택입니다.
- 나와 공유한 구성 요소 - 다른 사용자가 자신의 계정으로 생성하고 사용자와 공유한 구성 요소입니다.
- 타사 관리형 - 사용자가 구독한 타사가 소유한 구성 요소입니다 AWS Marketplace.

AWS CLI

다음 예제는 list-components 명령을 사용하여 계정이 소유한 Image Builder 구성 요소 목록을 반환하는 방법을 보여줍니다.

```
aws imagebuilder list-components
```

구성 요소 소유권을 기준으로 필터링할 수도 있습니다. 사용자 속성은 나열하려는 구성 요소를 소유하는 사람을 정의합니다. 기본적으로 이 요청은 계정이 소유한 구성 요소 목록을 반환합니다. 구성 요소 소유자별로 결과를 필터링하려면, list-components 명령을 실행할 때 `--owner` 매개 변수와 함께 다음 값 중 하나를 지정하십시오.

구성 요소 소유자 값

- `AWSMarketplace`
- `Amazon`
- `Self`
- `Shared`
- `ThirdParty`

다음 예제는 결과를 필터링하기 위한 `--owner` 매개 변수가 포함된 list-components 명령을 보여줍니다.

```
aws imagebuilder list-components --owner Self
{
    "requestId": "012a3456-b789-01cd-e234-fa5678b9012b",
    "componentVersionList": [
        {
```

```
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/sample-
component01/1.0.0",
            "name": "sample-component01",
            "version": "1.0.0",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2020-09-24T16:58:24.444Z"
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/sample-
component01/1.0.1",
            "name": "sample-component01",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2021-07-10T03:38:46.091Z"
        }
    ]
}
```

```
aws imagebuilder list-components --owner Amazon
```

```
aws imagebuilder list-components --owner Shared
```

```
aws imagebuilder list-components --owner ThirdParty
```

에서 구성 요소 빌드 버전 나열 AWS CLI

다음 예시는 list-component-build-versions 명령을 사용하여 특정 의미 체계 버전이 있는 구성 요소 빌드 버전을 나열하는 방법을 보여줍니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

```
aws imagebuilder list-component-build-versions --component-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:component/example-component/1.0.1
{
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "componentSummaryList": [
        {
```

```
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
examplecomponent/1.0.1/1",
            "name": "examplecomponent",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "description": "An example component that builds, validates and tests an
 image",
            "changeDescription": "Updated version.",
            "dateCreated": "2020-02-19T18:53:45.940Z",
            "tags": {
                "KeyName": "KeyValue"
            }
        }
    ]
}
```

에서 구성 요소 세부 정보 가져오기 AWS CLI

다음 예제는 구성 요소의 Amazon 리소스 이름(ARN)을 지정할 때 get-component 명령을 사용하여 구성 요소 세부 정보를 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-component --component-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/example-component/1.0.1/1
   {
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11112",
    "component": {
        "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
examplecomponent/1.0.1/1",
        "name": "examplecomponent",
        "version": "1.0.1",
        "type": "BUILD",
        "platform": "Linux",
        "owner": "123456789012",
        "data": "name: HelloWorldTestingDocument\ndescription: This is hello world
 testing document... etc.\"\n",
        "encrypted": true,
        "dateCreated": "2020-09-24T16:58:24.444Z",
        "tags": {}
    }
}
```

에서 구성 요소 정책 세부 정보 가져오기 AWS CLI

다음 예제는 구성 요소의 ARN을 지정할 때 get-component-policy 명령을 사용하여 구성 요소 정책의 세부 정보를 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-component-policy --component-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/example-component/1.0.1
```

AWS Marketplace 구성 요소를 사용하여 이미지 사용자 지정

는 독립 소프트웨어 공급업체(ISVs)가 만든 다양한 이미지 외에도 자체 Image Builder 이미지를 사용자 지정하는 데 사용할 수 있는 구성 요소를 AWS Marketplace 제공합니다. 이미지 레시피에서 이러한 AWS Marketplace 구성 요소를 사용하여 새 이미지를 빌드하려면 먼저 이러한 구성 요소를 구독해야 합니다.

이미지 레시피에서 AWS Marketplace 구성 요소를 지정하면 Image Builder는 구독을 검증하고 종속성 검사를 수행하여 사용하는 데 필요한 리소스가 있는지 확인합니다. 검증에 성공하면 Image Builder는 이미지 파이프라인 빌드에서 사용할 구성 요소 및 아티팩트에 대한 보안 다운로드를 생성합니다.

AWS Marketplace 구성 요소 검색

다음과 같이 Image Builder 콘솔의 제품 검색 페이지에서 레시피에 사용할 AWS Marketplace 소프트웨어 구성 요소를 검색할 수 있습니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창의 AWS Marketplace 섹션에서 제품 검색을 선택합니다.
3. Components(구성 요소) 탭을 선택합니다. 이 탭에는 연결된 구성 요소가 포함된 전송 옵션을 사용하는 모든 AWS Marketplace 제품이 나열됩니다 AWS Marketplace.
4. 구성 요소가 포함된 특정 소프트웨어 제품을 검색하려면 검색 창에 이름의 일부를 입력하거나 , `Status`, `Operating System` `Publisher`또는 로 필터링할 수 있습니다`Categories`. 검색 창에는 결과에 대한 페이지 매김 컨트롤도 포함되어 있습니다.

결과

각 AWS Marketplace 제품에는 다음 정보가 포함된 자체 세부 정보 패널이 있습니다.

AWS Marketplace 제품 이름 및 로고

소프트웨어 제품 이름은의 제품 세부 정보에 연결됩니다 AWS Marketplace. 링크를 선택하여의 제품에 대해 자세히 알아볼 수 있습니다 AWS Marketplace. 또는 이미 조사를 완료한 경우 구독 옵션 보기 버튼을 사용하여 구독 옵션 요약을 보고 검색 결과에서 직접 구독할 수 있습니다.

버전

여기에는 구성 요소의 기본 버전이 포함됩니다.

운영 체제

구성 요소가 실행되도록 설계된 운영 체제입니다.

게시자

구성 요소의 게시자입니다. 이는의 게시자 세부 정보 페이지에 연결됩니다 AWS Marketplace. 브라우저의 새 탭에서 게시자 세부 정보 페이지가 열립니다.

Categories

구성 요소에 적용되는 하나 이상의 AWS Marketplace 제품 범주입니다.

상태

이 제품을 구독하는지 여부를 표시합니다. 구독하지 않은 경우 구독 옵션 보기를 선택하여 AWS Marketplace 제품의 구독 옵션 요약을 확인하고 선택적으로 Image Builder 콘솔에서 직접 구독할 수 있습니다.

연결된 구성 요소

AWS Marketplace 제품에 구독에 포함된 버전이 하나 이상 있는 경우 연결된 구성 요소 섹션에 표시됩니다. 섹션은 처음에 축소되고 연결된 구성 요소의 수가 표시됩니다. 섹션을 확장하여 자세한 내용을 볼 수 있습니다.

> **Note**
>
> AWS Marketplace 이미지 제품과 연결된 인터넷 보안 센터(CIS) 구성 요소는 제품 검색 결과에 표시되지 않습니다. 이미지 제품을 구독하면 연결된 구성 요소가 구독 페이지에 표시되고 이미지 레시피 생성 대화 상자에 타사 구성 요소로 표시됩니다. 구성 요소에 대한 자세한 내용은 섹션을 참조하세요CIS 하드닝 구성 요소.

AWS Marketplace 구성 요소 구독

레시피에 사용할 구성 요소가 있는 AWS Marketplace 제품을 찾은 후 다음과 같이 Image Builder 콘솔에서 직접 제품을 구독하거나 AWS Marketplace 콘솔에서 구독할 수 있습니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창의 AWS Marketplace 섹션에서 제품 검색을 선택합니다.
3. Components(구성 요소) 탭을 선택합니다. 이 탭에는 연결된 구성 요소가 포함된 전송 옵션을 사용하는 모든 AWS Marketplace 제품이 나열됩니다 AWS Marketplace.
4. 특정 AWS Marketplace 제품을 검색하려면 검색 창에 이름의 일부를 입력합니다. 게시자를 알고 있지만 정확한 제품 이름이나 철자를 모르는 경우를 기준으로 필터링`Publisher`하여 게시자가 사용할 수 있는 제품 목록을 가져올 수도 있습니다.
5. 결과 목록에서 구독하려는 제품을 선택하고 구독 옵션 보기를 선택합니다. 여기에는 AWS Marketplace 제품의 구독 옵션 요약이 표시됩니다.
6. Image Builder 콘솔을 벗어나지 않고 제품을 구독하려면 구독을 선택합니다. 구독이 처리 중이라는 알림을 받습니다. 구독하면 상태가 로 업데이트됩니다`Subscribed`.

현재 구독 중인 AWS Marketplace 제품에 대한 자세한 내용은에 설명된 콘솔 단계를 참조하세요AWS Marketplace Image Builder의 구독.

Image Builder 이미지 레시피에서 AWS Marketplace 구성 요소 사용

다른 유형의 AWS Marketplace 구성 요소를 사용하는 것과 동일한 방식으로 Image Builder 이미지 레시피의 구성 요소를 사용할 수 있습니다. AWS Marketplace 이미지 제품과 연결된 대부분의 구성 요소의 경우 소유권 범주는 입니다`AWS Marketplace`. 예를 들어 구독한 AWS Marketplace 제품의 빌드 구성 요소를 사용하려면 빌드 구성 요소 추가를 선택하고 목록에서 `AWS Marketplace`를 선택합니다. 그러면 구성 요소를 나열 AWS Marketplace 하는 콘솔 인터페이스 오른쪽에 선택 패널이 열립니다.

> **Note**
>
> CIS 강화 구성 요소를 찾는 경우 대신 소유권 목록에서 `Third party managed`를 선택합니다`AWS Marketplace`.

구성 요소의 파라미터를 선택, 정렬 및 구성하는 방법에 대한 자세한 내용은 섹션을 참조하세요이미지 레시피의 새 버전 생성.

관리형 구성 요소를 사용하여 Image Builder 이미지 사용자 지정

관리형 구성 요소는에 의해 생성되며 AWS, 인터넷 보안 센터(CIS)와 같은 타사 조직과 협력하여 생성되기도 합니다. 이미지 또는 컨테이너 레시피에서 관리형 구성 요소를 사용하는 경우 Amazon은 패치 및 기타 업데이트가 적용된 최신 구성 요소 버전을 제공합니다. 구성 요소 목록을 가져오거나 구성 요소 정보를 가져오려면 섹션을 참조하세요구성 요소 세부 정보 나열 및 보기.

다음 주요 AWS 관리형 구성 요소 목록에는를 통해 CIS 강화 AMIs를 구독할 때 사용할 수 있는 구성 요소가 포함되어 있습니다 AWS Marketplace.

주요 구성 요소

- Image Builder Windows 이미지에 대한 Distributor 패키지 관리형 구성 요소 애플리케이션 설치
- CIS 하드닝 구성 요소
- Image Builder용 Amazon 관리형 STIG 하드닝 구성 요소

Image Builder Windows 이미지에 대한 Distributor 패키지 관리형 구성 요소 애플리케이션 설치

AWS Systems Manager Distributor를 사용하면 소프트웨어를 패키징하고 AWS Systems Manager 관리형 노드에 게시할 수 있습니다. 자체 소프트웨어를 패키징하여 게시하거나 Distributor를 사용하여 AWS에서 제공하는 에이전트 소프트웨어 패키지를 찾아 게시할 수 있습니다. Systems Manager Distributor에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Distributor를 참조하세요.

Distributor용 관리형 구성 요소

다음 Image Builder 관리형 구성 요소는 AWS Systems Manager Distributor를 사용하여 Windows 인스턴스에 애플리케이션 패키지를 설치합니다.

- `distributor-package-windows` 관리형 구성 요소는 AWS Systems Manager Distributor를 사용하여 Windows 이미지의 빌드 인스턴스에 지정하는 애플리케이션 패키지를 설치합니다. 레시피에 이 구성 요소를 포함할 때 파라미터를 구성하려면 `distributor-package-windows`(을)를 독립형 구성 요소로 구성 섹션을 참조하세요.
- 구성 `aws-vss-components-windows` 요소는 AWS Systems Manager Distributor를 사용하여 Windows 이미지 빌드 인스턴스에 `AwsVssComponents` 패키지를 설치합니다. 레시피에 이 구성요소를 포함할 때 파라미터를 구성하려면 `aws-vss-components-windows`를 독립형 구성 요소로 구성 섹션을 참조하세요.

Image Builder 레시피에서 관리형 구성 요소를 사용하는 방법에 대한 자세한 내용은 이미지 레시피는 이미지 레시피의 새 버전 생성, 컨테이너 레시피는 컨테이너 레시피의 새 버전 생성(을)를 참조하세요. `AwsVssComponents` 패키지에 대한 자세한 내용은 Amazon EC2 사용 설명서의 VSS 애플리케이션 일치 스냅샷 생성을 참조하십시오.

사전 조건

Systems Manager Distributor를 사용하여 애플리케이션 패키지를 설치하는 Image Builder 구성 요소를 사용하기 전에 먼저 다음 사전 요구 사항이 충족되는지 확인해야 합니다.

- Systems Manager Distributor를 사용하여 인스턴스에 애플리케이션 패키지를 설치하는 Image Builder 구성 요소에는 Systems Manager API를 호출할 수 있는 권한이 필요합니다. Image Builder 레시피의 구성 요소를 사용하기 전에 권한을 부여하는 IAM 정책과 역할을 생성해야 합니다. 권한을 구성하려면 Systems Manager Distributor 권한 구성(을)를 참조하세요.

> **Note**
>
> Image Builder는 현재 인스턴스를 재부팅하는 Systems Manager Distributor 패키지를 지원하지 않습니다. 예를 들어, `AWSNVMe`, `AWSPVDrivers`, 및 `AwsEnaNetworkDriver` Distributor 패키지는 인스턴스를 재부팅하므로 허용되지 않습니다.

Systems Manager Distributor 권한 구성

`distributor-package-windows` 구성 요소 및 이를 사용하는 기타 구성 요소(예`aws-vss-components-windows`:)를 실행하려면 빌드 인스턴스에 대한 추가 권한이 필요합니다. 빌드 인스턴스는 Systems Manager API를 호출하여 Distributor 설치를 시작하고 결과를 폴링할 수 있어야 합니다.

의 다음 절차에 따라 Image Builder 구성 요소가 빌드 인스턴스에서 Systems Manager Distributor 패키지를 설치할 수 있는 권한을 부여하는 사용자 지정 IAM 정책 및 역할을 AWS Management Console 생성합니다.

1단계: 정책 생성

Distributor 권한을 위한 IAM 정책을 생성합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 정책을 선택한 후 정책 생성을 선택합니다.

3. 정책 생성 페이지에서 JSON 탭을 선택한 후 기본 콘텐츠를 다음 JSON 정책으로 바꾸고 필요에 따라 파티션, 리전 및 계정 ID를 대체하거나 와일드카드를 사용합니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowDistributorSendCommand",
   "Effect": "Allow",
   "Action": [
    "ssm:SendCommand"
   ],
   "Resource": [
    "arn:${AWS::Partition}:ssm:${AWS::Region}::document/AWS-ConfigureAWSPackage",
    "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:instance/*"
   ]
  },
  {
   "Sid": "AllowGetCommandInvocation",
   "Effect": "Allow",
   "Action": [
    "ssm:GetCommandInvocation"
   ],
   "Resource": [
    "*"
   ]
  }
 ]
}
```

4. 정책 검토를 선택합니다.
5. 이름에 이 정책을 식별하기 위한 이름(예: *InvokeDistributor*) 또는 선호하는 다른 이름을 입력합니다.
6. (선택 사항)설명에 정책의 목적에 대한 설명을 입력합니다.
7. 정책 생성을 선택합니다.

2단계: 역할 생성

Distributor 권한을 위한 IAM 역할을 생성합니다.

1. IAM 콘솔의 탐색 창에서 역할을 선택하고 역할 생성을 선택합니다.

2. 신뢰할 수 있는 유형의 엔터티 선택에서 AWS 서비스(을)를 선택합니다.
3. 이 역할을 사용할 서비스 선택에서 EC2와 다음: 권한을 차례대로 선택합니다.
4. 사용 사례 선택에서 EC2를 선택한 후 다음: 권한을 선택합니다.
5. 정책 목록에서 AmazonSSMManagedInstanceCore 옆에 있는 확인란을 선택합니다. (목록을 좁혀야 할 경우 검색 상자에 `SSM` 입력)
6. 이 정책 목록에서 EC2InstanceProfileForImageBuilder 옆의 상자를 선택합니다. (목록을 좁혀야 할 경우 검색 상자에 `ImageBuilder` 입력)
7. 다음: 태그를 선택합니다.
8. (선택 사항) 이 역할에 대한 액세스를 구성, 추적 또는 제어할 태그-키 값 페어를 하나 이상 추가한 후 다음: 검토를 선택합니다.
9. 역할 이름에 역할의 이름(예: *`InvokeDistributor`* 또는 자신이 선호하는 다른 이름)을 입력합니다.
10. (선택 사항) 역할 설명에서 기본 텍스트를 이 역할의 목적에 대한 설명으로 바꿉니다.
11. 역할 생성을 선택합니다. 그러면 역할 페이지로 돌아갑니다.

3단계: 역할에 정책 연결

Distributor 권한을 설정하는 마지막 단계는 IAM 역할에 IAM 정책을 연결하는 것입니다.

1. IAM 콘솔의 역할 페이지에서 방금 생성한 역할을 선택합니다. 역할 요약 페이지가 열립니다.
2. 정책 연결을 선택합니다.
3. 이전 절차에서 만든 정책을 검색하고 이름 옆의 확인란을 선택합니다.
4. 정책 연결을 선택합니다.

Systems Manager Distributor를 사용하는 구성 요소가 포함된 모든 이미지의 경우 Image Builder 인프라 구성 리소스에서 이 역할을 사용하십시오. 자세한 내용은 인프라 구성 생성 단원을 참조하십시오.

`distributor-package-windows`(을)를 독립형 구성 요소로 구성

레시피에서 `distributor-package-windows` 구성 요소를 사용하려면 설치할 패키지를 구성하는 다음 파라미터를 설정합니다.

> **Note**
>
> 레시피에 `distributor-package-windows` 구성 요소를 사용하기 전에 모든 사전 조건 요소가 충족되는지 확인해야 합니다.

- 작업(필수) - 패키지 설치 또는 제거 여부를 지정합니다. 유효한 값에는 `Install` 및 `Uninstall`(이)가 있습니다. 기본값은 `Install`입니다.
- Packagename(필수) - 설치 또는 제거할 Distributor 패키지의 이름. 유효한 패키지 이름 목록은 Distributor 패키지 찾기 섹션을 참조하세요.
- PackageVersion(선택 사항) - 설치할 Distributor 패키지의 버전. PackageVersion은 기본적으로 권장 버전으로 설정됩니다.
- AdditionArguments(선택 사항) - 패키지를 설치, 제거 또는 업데이트하기 위해 스크립트에 제공하는 추가 파라미터가 포함된 JSON 문자열. 자세한 내용은 Systems Manager 명령 문서 플러그인 참조 페이지의 aws:configurePackage 입력 섹션의 additionalArguments를 참조하세요.

`aws-vss-components-windows`를 독립형 구성 요소로 구성

레시피에서 `aws-vss-components-windows` 구성 요소를 사용하는 경우 선택적으로 특정 버전의 `AwsVssComponents` 패키지를 사용하도록 `PackageVersion` 파라미터를 설정할 수 있습니다. 이 파라미터를 생략하면 구성 요소는 기본적으로 권장 버전의 `AwsVssComponents` 패키지를 사용합니다.

> **Note**
>
> 레시피에 `aws-vss-components-windows` 구성 요소를 사용하기 전에 모든 사전 조건 요소가 충족되는지 확인해야 합니다.

Distributor 패키지 찾기

Amazon과 타사는 Systems Manager Distributor로 설치할 수 있는 공개 패키지를 제공합니다.

에서 사용 가능한 패키지를 보려면 AWS Systems Manager 콘솔에 AWS Management Console로그인하고 탐색 창에서 Distributor를 선택합니다. Distributor 페이지에는 사용 가능한 모든 패키지가 표시됩니다. 에서 사용 가능한 패키지를 나열하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 패키지 보기(명령줄)를 AWS CLI참조하세요.

또한 사용자 고유의 Systems Manager Distributor 패키지를 만들 수도 있습니다. 자세한 내용은 AWS Systems Manager 사용 설명서에서 패키지 생성을 참조하세요.

CIS 하드닝 구성 요소

CIS(Center for Internet Security)는 커뮤니티 중심의 비영리 단체입니다. 해당 단체의 사이버 보안 전문가들은 서로 협력하여 공공 및 민간 조직을 사이버 위협으로부터 보호하는 IT 보안 지침을 개발합니다. CIS 벤치마크라고 하는 세계적으로 인정받는 모범 사례 세트는 전 세계 IT 조직이 시스템을 안전하게 구성하는 데 도움이 됩니다. 인기 기사, 블로그 게시물, 팟캐스트, 웨비나, 백서를 보려면 Center for Internet Security 웹 사이트의 CIS 인사이트를 참조하세요.

CIS 벤치마크

CIS는 운영 체제, 클라우드 플랫폼, 애플리케이션, 데이터베이스 등을 포함한 특정 기술에 대한 구성 모범 사례를 제공하는 CIS 벤치마크라고 하는 일련의 구성 지침을 만들고 유지 관리합니다. CIS 벤치마크는 PCI DSS, HIPAA, DoD 클라우드 컴퓨팅 SRG, FISMA, DFARS 및 FEDRAMP와 같은 조직 및 표준에서 업계 표준으로 인정받고 있습니다. 자세한 내용은 Center for Internet Security 웹 사이트의 CIS 벤치마크를 참조하세요.

CIS 하드닝 구성 요소

에서 CIS 강화 이미지를 구독하면 스크립트를 실행하는 관련 강화 구성 요소에도 AWS Marketplace액세스하여 구성에 CIS 벤치마크 레벨 1 지침을 적용할 수 있습니다. CIS 조직은 CIS 하드닝 구성 요소가 최신 지침을 반영하도록 소유하고 유지 관리합니다.

> **Note**
>
> CIS 하드닝 구성 요소는 Image Builder 레시피의 표준 구성 요소 순서 지정 규칙을 따르지 않습니다. 출력 이미지에 대해 벤치마크 테스트가 실행되도록 하기 위해 CIS 하드닝 구성 요소는 항상 마지막에 실행됩니다.

Image Builder용 Amazon 관리형 STIG 하드닝 구성 요소

보안 기술 규현 가이드(STIG)는 Defense Information Systems Agency(DISA)에서 생성한 구성 하드닝 표준으로 정보 시스템과 소프트웨어를 보호합니다. 시스템이 STIG 표준 규정을 준수하려면 다양한 보안 설정을 설치, 구성 및 테스트해야 합니다.

Image Builder는 STIG 하드닝 구성 요소를 제공하여 기본 STIG 표준을 준수하는 이미지를 보다 효율적으로 구축할 수 있도록 지원합니다. 이러한 STIG 구성 요소는 잘못된 구성을 검사하고 수정 스크립

트를 실행합니다. STIG 규정을 준수하는 구성 요소를 사용하기 위해 별도의 비용은 필요하지 않습니다.

> **Important**
>
> 극히 일부 예외를 제외하고 STIG 하드닝 구성 요소는 타사 패키지를 설치하지 않습니다. 인스턴스에 타사 패키지가 이미 설치되어 있고 Image Builder에서 해당 패키지에 대해 지원하는 관련 STIG가 있는 경우 하드닝 구성 요소가 해당 패키지를 적용합니다.

이 페이지에는 Image Builder에서 지원하는 모든 STIG가 나열되어 있으며, 새 이미지를 빌드하고 테스트할 때 Image Builder가 시작하는 EC2 인스턴스에 적용됩니다. 이미지에 추가 STIG 설정을 적용하려는 경우 사용자 지정 구성 요소를 생성하여 이미지를 구성할 수 있습니다. 사용자 지정 구성 요소와 생성 방법에 대한 자세한 내용은 구성 요소를 사용하여 Image Builder 이미지 사용자 지정 섹션을 참조하십시오.

이미지를 생성할 때 STIG 하드닝 구성 요소는 지원되는 STIG가 적용되었는지 또는 건너뛰었는지 여부를 기록합니다. Image Builder 로그에서 STIG 하드닝 구성 요소를 사용하는 이미지를 검토하는 것이 좋습니다. Image Builder 로그에 액세스하고 검토하는 방법에 대한 자세한 내용은 파이프라인 빌드 문제 해결 섹션을 참조하십시오.

규정 준수 수준

- 높음(카테고리 I)

  가장 심각한 위험. 기밀성, 가용성 또는 무결성의 손실을 초래할 수 있는 모든 취약성을 포함합니다.
- 보통(카테고리 II)

  기밀성, 가용성 또는 무결성의 손실을 초래할 수 있는 모든 취약성을 포함하지만 위험은 완화될 수 있습니다.
- 낮음(카테고리 III)

  기밀성, 가용성 또는 무결성의 손실을 방지하기 위해 조치를 저하시키는 모든 취약성입니다.

주제

- Windows STIG 하드닝 구성 요소
- Windows용 STIG 버전 이력 로그
- Linux STIG 하드닝 구성 요소

- Linux용 STIG 버전 이력 로그
- SCAP 규정 준수 검증기 구성 요소

Windows STIG 하드닝 구성 요소

AWSTOE Windows STIG 강화 구성 요소는 독립 실행형 서버용으로 설계되었으며 로컬 그룹 정책을 적용합니다. STIG 준수 하드닝 구성 요소는 Windows 인프라에 국방부(DoD)의 InstallRoot를 설치하여 DoD 인증서를 다운로드, 설치 및 업데이트합니다. 또한 STIG 규정 준수를 유지하기 위해 불필요한 인증서를 제거합니다. 현재 STIG 기준은 2012 R2, 2016, 2019, 2022와 같은 Windows Server 버전에서 지원됩니다.

이 섹션에는 각 Windows STIG 하드닝 구성 요소의 현재 설정과 버전 이력 로그가 나열되어 있습니다.

STIG-Build-Windows-Low 버전 2025.1.x

다음 목록에는 하드닝 구성 요소가 인프라에 적용하는 STIG 설정이 나와 있습니다. 지원되는 설정이 인프라에 적용되지 않는 경우 하드닝 구성 요소는 해당 설정을 건너뛰고 계속 진행합니다. 예를 들어 일부 STIG 설정은 독립 실행형 서버에 적용되지 않을 수 있습니다. 조직별 정책도 강화 구성 요소가 적용되는 설정 종류(예: 관리자가 문서 설정을 검토하기 위한 요구 사항)에 영향을 미칠 수 있습니다.

Windows STIG의 전체 목록은 STIG 문서 라이브러리를 참조하세요. 전체 목록을 보는 방법에 대한 자세한 내용은 STIG 보기 도구를 참조하세요.

- Windows Server 2022 STIG 버전 2 릴리스 2

  V-254335, V-254336, V-254337, V-254338, V-254351, V-254357, V-254363 및 V-254481
- Windows Server 2019 STIG 버전 3 릴리스 2

  V-205691, V-205819, V-205858, V-205859, V-205860, V-205870, V-205871, 및 V-205923
- Windows Server 2016 STIG 버전 2 릴리스 9

  V-224916, V-224917, V-224918, V-224919, V-224931, V-224942 및 V-225060
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5

  V-225537, V-225536, V-225526, V-225525, V-225514, V-225511, V-225490, V-225489, V-225488, V-225487, V-225485, V-225484, V-225483, V-225482, V-225481, V-225480, V-225479, V-225476, V-225473, V-225468, V-225462, V-225460, V-225459, V-225412, V-225394, V-225392, V-225376, V-225363, V-225362, V-225360, V-225359, V-225358, V-225357, V-225355, V-225343, V-225342,

V-225336, V-225335, V-225334, V-225333, V-225332, V-225331, V-225330, V-225328, V-225327, V-225324, V-225319, V-225318 및 V-225250

- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2

  Microsoft .NET Framework의 카테고리 III 취약점에 대해서는 STIG 설정이 적용되지 않습니다.

- Windows 방화벽 STIG 버전 2 릴리스 2

  V-241994, V-241995, V-241996, V-241999, V-242000, V-242001, V-242006, V-242007 및 V-242008

- Internet Explorer 11 STIG 버전 2 릴리스 5

  V-223016, V-223056 및 V-223078

- Microsoft Edge STIG 버전 2 릴리스 2(Windows Server 2022만 해당)

  V-235727, V-235731, V-235751, V-235752 및 V-235765

STIG-Build-Windows-Medium 버전 2025.1.x

다음 목록에는 하드닝 구성 요소가 인프라에 적용하는 STIG 설정이 나와 있습니다. 지원되는 설정이 인프라에 적용되지 않는 경우 하드닝 구성 요소는 해당 설정을 건너뛰고 계속 진행합니다. 예를 들어 일부 STIG 설정은 독립 실행형 서버에 적용되지 않을 수 있습니다. 조직별 정책도 강화 구성 요소가 적용되는 설정 종류(예: 관리자가 문서 설정을 검토하기 위한 요구 사항)에 영향을 미칠 수 있습니다.

Windows STIG의 전체 목록은 STIG 문서 라이브러리를 참조하세요. 전체 목록을 보는 방법에 대한 자세한 내용은 STIG 보기 도구를 참조하세요.

> **Note**
>
> STIG-Build-Windows-Medium 강화 구성 요소에는 범주 II 취약성에 대해 특별히 나열된 STIG 설정 외에도 STIG-Build-Windows-Low 강화 구성 요소에 AWSTOE 적용되는 나열된 모든 STIG 설정이 포함됩니다.

- Windows Server 2022 STIG 버전 2 릴리스 2

  하드닝 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-254247, V-254265, V-254269, V-254270, V-254271, V-254272, V-254273, V-254274, V-254276, V-254277, V-254278, V-254285, V-254286, V-254287, V-254288, V-254289, V-254290, V-254291, V-254292, V-254300, V-254301, V-254302, V-254303, V-254304, V-254305, V-254306, V-254307, V-254308, V-254309, V-254310, V-254311, V-254312, V-254313, V-254314, V-254315, V-254316, V-254317, V-254318, V-254319, V-254320, V-254321, V-254322, V-254323, V-254324, V-254325, V-254326, V-254327, V-254328, V-254329, V-254330, V-254331, V-254332, V-254333, V-254334, V-254339, V-254341, V-254342, V-254344, V-254345, V-254346, V-254347, V-254348, V-254349, V-254350, V-254355, V-254356, V-254356, V-254358, V-254359, V-254360, V-254361, V-254362, V-254364, V-254365, V-254366, V-254367, V-254368, V-254369, V-254370, V-254371, V-254372, V-254373, V-254375, V-254376, V-254377, V-254379, V-254380, V-254382, V-254383, V-254384, V-254431, V-254432, V-254433, V-254434, V-254435, V-254436, V-254438, V-254439, V-254442, V-254443, V-254444, V-254445, V-254449, V-254450, V-254451, V-254452, V-254453, V-254454, V-254455, V-254456, V-254459, V-254460, V-254461, V-254462, V-254463, V-254464, V-254468, V-254470, V-254471, V-254472, V-254473, V-254476, V-254477, V-254478, V-254479, V-254480, V-254482, V-254483, V-254484, V-254485, V-254486, V-254487, V-254488, V-254489, V-254490, V-254493, V-254494, V-254495, V-254497, V-254499, V-254501, V-254502, V-254503, V-254504, V-254505, V-254507, V-254508, V-254509, V-254510, V-254511, 및 V-254512

- Windows Server 2019 STIG 버전 3 릴리스 2

  하드닝 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-205625, V-205626, V-205627, V-205629, V-205630, V-205633, V-205634, V-205635, V-205636, V-205637, V-205638, V-205639, V-205643, V-205644, V-205648, V-205649, V-205650, V-205651, V-205652, V-205655, V-205656, V-205659, V-205660, V-205662, V-205671, V-205672, V-205673, V-205675, V-205676, V-205678, V-205679, V-205680, V-205681, V-205682, V-205683, V-205684, V-205685, V-205686, V-205687, V-205688, V-205689, V-205690, V-205692, V-205693, V-205694, V-205697, V-205698, V-205708, V-205709, V-205712, V-205714, V-205716, V-205717, V-205718, V-205719, V-205720, V-205722, V-205729, V-205730, V-205733, V-205747, V-205751, V-205752, V-205754, V-205756, V-205758, V-205759, V-205760, V-205761, V-205762, V-205764, V-205765, V-205766, V-205767, V-205768, V-205769, V-205770, V-205771, V-205772, V-205773, V-205774, V-205775, V-205776, V-205777, V-205778, V-205779, V-205780, V-205781, V-205782, V-205783, V-205784, V-205795, V-205796, V-205797, V-205798, V-205801, V-205808, V-205809, V-205810, V-205811, V-205812, V-205813, V-205814, V-205815, V-205816, V-205817, V-205821, V-205822, V-205823, V-205824, V-205825, V-205826, V-205827, V-205828, V-205830, V-205832, V-205833, V-205834, V-205835, V-205836, V-205837, V-205838, V-205839, V-205840, V-205841, V-205842,

V-205861, V-205863, V-205865, V-205866, V-205867, V-205868, V-205869, V-205872, V-205873, V-205874, V-205911, V-205912, V-205915, V-205916, V-205917, V-205918, V-205920, V-205921, V-205922, V-205924, V-205925, V-236001, 및 V-257503

- Windows Server 2016 STIG 버전 2 릴리스 9

  하드닝 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-224850, V-224852, V-224853, V-224854, V-224855, V-224856, V-224857, V-224858, V-224859, V-224866, V-224867, V-224868, V-224869, V-224870, V-224871, V-224872, V-224873, V-224881, V-224882, V-224883, V-224884, V-224885, V-224886, V-224887, V-224888, V-224889, V-224890, V-224891, V-224892, V-224893, V-224894, V-224895, V-224896, V-224897, V-224898, V-224899, V-224900, V-224901, V-224902, V-224903, V-224904, V-224905, V-224906, V-224907, V-224908, V-224909, V-224910, V-224911, V-224912, V-224913, V-224914, V-224915, V-224920, V-224922, V-224924, V-224925, V-224926, V-224927, V-224928, V-224929, V-224930, V-224935, V-224936, V-224937, V-224938, V-224939, V-224940, V-224941, V-224943, V-224944, V-224945, V-224946, V-224947, V-224948, V-224949, V-224951, V-224952, V-224953, V-224955, V-224956, V-224957, V-224959, V-224960, V-224962, V-224963, V-225010, V-225013, V-225014, V-225015, V-225016, V-225017, V-225018, V-225019, V-225021, V-225022, V-225023, V-225024, V-225028, V-225029, V-225030, V-225031, V-225032, V-225033, V-225034, V-225035, V-225038, V-225039, V-225040, V-225041, V-225042, V-225043, V-225047, V-225049, V-225050, V-225051, V-225052, V-225055, V-225056, V-225057, V-225058, V-225059, V-225061, V-225062, V-225063, V-225064, V-225065, V-225066, V-225067, V-225068, V-225069, V-225072, V-225073, V-225074, V-225076, V-225078, V-225080, V-225081, V-225082, V-225083, V-225084, V-225086, V-225087, V-225088, V-225089, V-225092, V-225093, V-236000, 및 V-257502

- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5

  강화 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-225574, V-225573, V-225572, V-225571, V-225570, V-225569, V-225568, V-225567, V-225566, V-225565, V-225564, V-225563, V-225562, V-225561, V-225560, V-225559, V-225558, V-225557, V-225555, V-225554, V-225553, V-225551, V-225550, V-225549, V-225548, V-225546, V-225545, V-225544, V-225543, V-225542, V-225541, V-225540, V-225539, V-225538, V-225535, V-225534, V-225533, V-225532, V-225531, V-225530, V-225529, V-225528, V-225527, V-225524, V-225523, V-225522, V-225521, V-225520, V-225519, V-225518, V-225517, V-225516, V-225515, V-225513, V-225510, V-225509, V-225508, V-225506, V-225504, V-225503, V-225502, V-225501, V-225500,

V-225494, V-225486, V-225478, V-225477, V-225475, V-225474, V-225472, V-225471, V-225470, V-225469, V-225464, V-225463, V-225461, V-225458, V-225457, V-225456, V-225455, V-225454, V-225453, V-225452, V-225448, V-225443, V-225442, V-225441, V-225415, V-225414, V-225413, V-225411, V-225410, V-225409, V-225408, V-225407, V-225406, V-225405, V-225404, V-225402, V-225401, V-225400, V-225398, V-225397, V-225395, V-225393, V-225391, V-225389, V-225386, V-225385, V-225384, V-225383, V-225382, V-225381, V-225380, V-225379, V-225378, V-225377, V-225375, V-225374, V-225373, V-225372, V-225371, V-225370, V-225369, V-225368, V-225367, V-225356, V-225353, V-225352, V-225351, V-225350, V-225349, V-225348, V-225347, V-225346, V-225345, V-225344, V-225341, V-225340, V-225339, V-225338, V-225337, V-225329, V-225326, V-225325, V-225317, V-225316, V-225315, V-225314, V-225305, V-225304, V-225303, V-225302, V-225301, V-225300, V-225299, V-225298, V-225297, V-225296, V-225295, V-225294, V-225293, V-225292, V-225291, V-225290, V-225289, V-225288, V-225287, V-225286, V-225285, V-225284, V-225283, V-225282, V-225281, V-225280, V-225279, V-225278, V-225277, V-225276, V-225275, V-225273, V-225272, V-225271, V-225270, V-225269, V-225268, V-225267, V-225266, V-225265, V-225264, V-225263, V-225261, V-225260, V-225259 및 V-225239

- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2

  하드닝 구성 요소가 카테고리 III (낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 V-225238 포함

- Windows 방화벽 STIG 버전 2 릴리스 2

  하드닝 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-241989, V-241990, V-241991, V-241993, V-241993, V-241998, V-241998, V-242003 및 V-242003

- Internet Explorer 11 STIG 버전 2 릴리스 5

  하드닝 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-223015, V-223017, V-223018, V-223019, V-223020, V-223021, V-223022, V-223023, V-223024, V-223025, V-223026, V-223027, V-223028, V-223029, V-223030, V-223031, V-223032, V-223033, V-223034, V-223035, V-223036, V-223037, V-223038, V-223039, V-223040, V-223041, V-223042, V-223043, V-223044, V-223045, V-223046, V-223048, V-223049, V-223050, V-223051, V-223052, V-223053, V-223054, V-223055, V-223057, V-223058, V-223059, V-223060, V-223061, V-223062, V-223063, V-223064, V-223065, V-223066, V-223067, V-223068, V-223069, V-223070, V-223071, V-223072, V-223073, V-223074, V-223075, V-223076, V-223077, V-223079, V-223080, V-223081,

V-223082, V-223083, V-223084, V-223085, V-223086, V-223087, V-223088, V-223089, V-223090, V-223091, V-223092, V-223093, V-223094, V-223095, V-223096, V-223097, V-223098, V-223099, V-223100, V-223101, V-223102, V-223103, V-223104, V-223105, V-223106, V-223107, V-223108, V-223109, V-223110, V-223111, V-223112, V-223113, V-223114, V-223115, V-223116, V-223117, V-223118, V-223119, V-223120, V-223121, V-223122, V-223123, V-223124, V-223125, V-223126, V-223127, V-223128, V-223129, V-223130, V-223131, V-223132, V-223133, V-223134, V-223135, V-223136, V-223137, V-223138, V-223139, V-223140, V-223141, V-223142, V-223143, V-223144, V-223145, V-223146, V-223147, V-223148, V-223149, V-250540, 및 V-250541

- Microsoft Edge STIG 버전 2 릴리스 2(Windows Server 2022만 해당)

  강화 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-235720, V-235721, V-235723, V-235724, V-235725, V-235726, V-235728, V-235729, V-235730, V-235732, V-235733, V-235734, V-235735, V-235736, V-235737, V-235738, V-235739, V-235740, V-235741, V-235742, V-235743, V-235744, V-235745, V-235746, V-235747, V-235748, V-235749, V-235750, V-235754, V-235756, V-235760, V-235761, V-235763, V-235764, V-235766, V-235767, V-235768, V-235769, V-235770, V-235771, V-235772, V-235773, V-235774, 및 V-246736

- Microsoft Defender STIG 버전 2 릴리스 4

  하드닝 구성 요소가 카테고리 III(낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-213427, V-213429, V-213430, V-213431, V-213432, V-213433, V-213434, V-213435, V-213436, V-213437, V-213438, V-213439, V-213440, V-213441, V-213442, V-213443, V-213444, V-213445, V-213446, V-213447, V-213448, V-213449, V-213450, V-213451, V-213455, V-213464, V-213465 및 V-213466

STIG-Build-Windows-High 버전 2025.1.x

다음 목록에는 하드닝 구성 요소가 인프라에 적용하는 STIG 설정이 나와 있습니다. 지원되는 설정이 인프라에 적용되지 않는 경우 하드닝 구성 요소는 해당 설정을 건너뛰고 계속 진행합니다. 예를 들어 일부 STIG 설정은 독립 실행형 서버에 적용되지 않을 수 있습니다. 조직별 정책도 강화 구성 요소가 적용되는 설정 종류(예: 관리자가 문서 설정을 검토하기 위한 요구 사항)에 영향을 미칠 수 있습니다.

Windows STIG의 전체 목록은 STIG 문서 라이브러리를 참조하세요. 전체 목록을 보는 방법에 대한 자세한 내용은 STIG 보기 도구를 참조하세요.

> **Note**
>
> STIG-Build-Windows-High 강화 구성 요소에는 범주 I 취약성에 대해 특별히 나열된 STIG 설정 외에도 STIG-Build-Windows-Low 및 STIG-Build-Windows-Medium 강화 구성 요소에 AWSTOE 적용되는 나열된 모든 STIG 설정이 포함됩니다.

- Windows Server 2022 STIG 버전 2 릴리스 2

  하드닝 구성 요소가 카테고리 II 및 III(중간 및 낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-254293, V-254352, V-254353, V-254354, V-254374, V-254378, V-254381, V-254446, V-254465, V-254466, V-254467, V-254469, V-254474, V-254475 및 V-254500

- Windows Server 2019 STIG 버전 3 릴리스 2

  하드닝 구성 요소가 카테고리 II 및 III(중간 및 낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-205653, V-205654, V-205711, V-205713, V-205724, V-205725, V-205757, V-205802, V-205804, V-205805, V-205806, V-205849, V-205908, V-205913, V-205914 및 V-205919

- Windows Server 2016 STIG 버전 2 릴리스 9

  하드닝 구성 요소가 카테고리 II 및 III(중간 및 낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-224874, V-224932, V-224933, V-224934, V-224954, V-224958, V-224961, V-225025, V-225044, V-225045, V-225046, V-225048, V-225053, V-225054 및 V-225079

- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5

  강화 구성 요소가 카테고리 II 및 III(중간 및 낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-225556, V-225552, V-225547, V-225507, V-225505, V-225498, V-225497, V-225496, V-225493, V-225492, V-225491, V-225449, V-225444, V-225399, V-225396, V-225390, V-225366, V-225365, V-225364, V-225354 및 V-225274

- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2

Microsoft .NET 프레임워크의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정이 포함되어 있습니다. 범주 I 취약점에 대해서는 추가 STIG 설정이 적용되지 않습니다.

- Windows 방화벽 STIG 버전 2 릴리스 2

  하드닝 구성 요소가 카테고리 II 및 III(중간 및 낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-241992, V-241997 및 V-242002
- Internet Explorer 11 STIG 버전 2 릴리스 5

  V-252910
- Microsoft Edge STIG 버전 2 릴리스 2(Windows Server 2022만 해당)

  강화 구성 요소가 카테고리 II 및 III(중간 및 낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-235758 및 V-235759
- Microsoft Defender STIG 버전 2 릴리스 4

  하드닝 구성 요소가 카테고리 II 및 III(중간 및 낮음) 취약성에 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

  V-213426, V-213426, V-213452, V-213452, V-213452, V-213453, V-213453 및 V-213453

Windows용 STIG 버전 이력 로그

이 섹션에서는 분기별 STIG 업데이트의 Windows 하드닝 구성 요소 버전 이력을 기록합니다. 한 분기의 변경 사항과 게시된 버전을 보려면 제목을 선택하여 정보를 확장하세요.

2025년 Q1 변경 사항 - 05/04/2025:

2025년 1분기 릴리스의 모든 STIG 구성 요소에 대한 Internet Explorer 11 STIG 버전 2 릴리스 5의 STIGS가 업데이트되었습니다.

- STIG-Build-Windows-Low 버전 2025.1.x
- STIG-Build-Windows-Medium 버전 2025.1.x
- STIG-Build-Windows-High 버전 2025.1.x

2024년 Q4 변경 사항 - 02/04/2025:

STIG 버전을 업데이트하고 다음과 같이 2024년 Q4 릴리스에 STIGS를 적용했습니다.

STIG-Build-Windows-Low 버전 2024.4.0

- Windows Server 2022 STIG 버전 2 릴리스 2
- Windows Server 2019 STIG 버전 3 릴리스 2
- Windows Server 2016 STIG 버전 2 릴리스 9
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2
- Windows 방화벽 STIG 버전 2 릴리스 2
- Internet Explorer 11 STIG 버전 2 릴리스 5
- Microsoft Edge STIG 버전 2 릴리스 2(Windows Server 2022만 해당)

STIG-Build-Windows-Medium 버전 2024.4.0

- Windows Server 2022 STIG 버전 2 릴리스 2
- Windows Server 2019 STIG 버전 3 릴리스 2
- Windows Server 2016 STIG 버전 2 릴리스 9
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2
- Windows 방화벽 STIG 버전 2 릴리스 2
- Internet Explorer 11 STIG 버전 2 릴리스 5
- Microsoft Edge STIG 버전 2 릴리스 2(Windows Server 2022만 해당)
- Defender STIG 버전 2 릴리스 4

STIG-Build-Windows-High 버전 2024.4.0

- Windows Server 2022 STIG 버전 2 릴리스 2
- Windows Server 2019 STIG 버전 3 릴리스 2
- Windows Server 2016 STIG 버전 2 릴리스 9
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5

- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2
- Windows 방화벽 STIG 버전 2 릴리스 2
- Internet Explorer 11 STIG 버전 2 릴리스 5
- Microsoft Edge STIG 버전 2 릴리스 2(Windows Server 2022만 해당)
- Defender STIG 버전 2 릴리스 4

2024년 Q3 변경 사항 - 10/04/2023(변경 없음):

2024년 3분기 릴리스의 Windows 구성 요소 STIGS에는 변경 사항이 없습니다.

2024년 2분기 변경 사항 - 2024년 5월 10일(변경 없음):

2024년 2분기 릴리스의 Windows 구성 요소 STIG에는 변경 사항이 없습니다.

2024년 1분기 변경 사항 - 2024년 2월 6일(변경 없음):

2024년 1분기 릴리스의 Windows 구성 요소 STIG에는 변경 사항이 없습니다.

2023년 4분기 변경 사항 - 2023년 12월 4일(변경 없음):

2023년 4분기 릴리스의 Windows 구성 요소 STIG에는 변경 사항이 없습니다.

2023년 3분기 변경 사항 - 2023년 10월 4일(변경 없음):

2023년 3분기 릴리스의 Windows 구성 요소 STIG에는 변경 사항이 없습니다.

2023년 2분기 변경 사항 - 2023년 5월 3일(변경 없음):

2023년 2분기 릴리스의 Windows 구성 요소 STIG에는 변경 사항이 없습니다.

2023년 1분기 변경 사항 - 2023년 3월 27일(변경 없음):

2023년 1분기 릴리스의 Windows 구성 요소 STIG에는 변경 사항이 없습니다.

2022년 4분기 변경 사항 - 2023년 2월 1일:

2022년 4분기 릴리스에 대한 STIG 버전 업데이트 및 STIG 적용 내용은 다음과 같습니다.

STIG-Build-Windows-Low 버전 2022.4.x

- Windows Server 2022 STIG 버전 1 릴리스 1

- Windows Server 2019 STIG 버전 2 릴리스 5
- Windows Server 2016 STIG 버전 2 릴리스 5
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2
- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 2 릴리스 3
- Microsoft Edge STIG 버전 1 릴리스 6(Windows Server 2022 전용)

STIG-Build-Windows-Medium 버전 2022.4.x

- Windows Server 2022 STIG 버전 1 릴리스 1
- Windows Server 2019 STIG 버전 2 릴리스 5
- Windows Server 2016 STIG 버전 2 릴리스 5
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2
- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 2 릴리스 3
- Microsoft Edge STIG 버전 1 릴리스 6(Windows Server 2022 전용)
- Defender STIG 버전 2 릴리스 4(Windows Server 2022 전용)

STIG-Build-Windows-High 버전 2022.4.x

- Windows Server 2022 STIG 버전 1 릴리스 1
- Windows Server 2019 STIG 버전 2 릴리스 5
- Windows Server 2016 STIG 버전 2 릴리스 5
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 5
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 2
- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 2 릴리스 3
- Microsoft Edge STIG 버전 1 릴리스 6(Windows Server 2022 전용)

- Defender STIG 버전 2 릴리스 4(Windows Server 2022 전용)

2022년 3분기 변경 사항 - 2022년 9월 30일(변경 없음):

2022년 3분기 릴리스의 Windows 구성 요소 STIG에는 변경 사항이 없습니다.

2022년 2분기 변경 사항 - 2022년 8월 2일:

2022년 2분기 릴리스에 STIG 버전을 업데이트하고 STIG를 적용했습니다.

STIG-Build-Windows-Low 버전 1.5.x

- Windows Server 2019 STIG 버전 2 릴리스 4
- Windows Server 2016 STIG 버전 2 릴리스 4
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 3
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1
- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 1 릴리스 19

STIG-Build-Windows-Medium 버전 1.5.x

- Windows Server 2019 STIG 버전 2 릴리스 4
- Windows Server 2016 STIG 버전 2 릴리스 4
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 3
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1
- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 1 릴리스 19

STIG-Build-Windows-High 버전 1.5.x

- Windows Server 2019 STIG 버전 2 릴리스 4
- Windows Server 2016 STIG 버전 2 릴리스 4
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 3
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1

- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 1 릴리스 19

2022년 1분기 변경 사항 - 2022년 8월 2일(변경 없음):

2022년 1분기 릴리스의 Windows 구성 요소 STIGS에는 변경 사항이 없습니다.

2021년 4분기 변경 사항 - 2021년 12월 20일:

2021년 4분기 릴리스에 STIG 버전을 업데이트하고 STIG를 적용했습니다.

STIG-Build-Windows-Low 버전 1.5.x

- Windows Server 2019 STIG 버전 2 릴리스 3
- Windows Server 2016 STIG 버전 2 릴리스 3
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 3
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1
- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 1 릴리스 19

STIG-Build-Windows-Medium 버전 1.5.x

- Windows Server 2019 STIG 버전 2 릴리스 3
- Windows Server 2016 STIG 버전 2 릴리스 3
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 3
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1
- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 1 릴리스 19

STIG-Build-Windows-High 버전 1.5.x

- Windows Server 2019 STIG 버전 2 릴리스 3
- Windows Server 2016 STIG 버전 2 릴리스 3
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 3
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1

- Windows 방화벽 STIG 버전 2 릴리스 1
- Internet Explorer 11 STIG 버전 1 릴리스 19

2021년 3분기 변경 사항 - 2021년 9월 30일:

2021년 3분기 릴리스에 STIG 버전을 업데이트하고 STIG를 적용했습니다.

STIG-Build-Windows-Low 버전 1.4.x

- Windows Server 2019 STIG 버전 2 릴리스 2
- Windows Server 2016 STIG 버전 2 릴리스 2
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 2
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1
- Windows 방화벽 STIG 버전 1 릴리스 7
- Internet Explorer 11 STIG 버전 1 릴리스 19

STIG-Build-Windows-Medium 버전 1.4.x

- Windows Server 2019 STIG 버전 2 릴리스 2
- Windows Server 2016 STIG 버전 2 릴리스 2
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 2
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1
- Windows 방화벽 STIG 버전 1 릴리스 7
- Internet Explorer 11 STIG 버전 1 릴리스 19

STIG-Build-Windows-High 버전 1.4.x

- Windows Server 2019 STIG 버전 2 릴리스 2
- Windows Server 2016 STIG 버전 2 릴리스 2
- Windows Server 2012 R2 MS STIG 버전 3 릴리스 2
- Microsoft .NET Framework 4.0 STIG 버전 2 릴리스 1
- Windows 방화벽 STIG 버전 1 릴리스 7
- Internet Explorer 11 STIG 버전 1 릴리스 19

Linux STIG 하드닝 구성 요소

이 섹션에는 Linux STIG 하드닝 구성 요소에 대한 정보와 버전 이력 로그가 수록되어 있습니다. Linux 배포판에 자체 STIG 설정이 없는 경우 하드닝 구성 요소가 RHEL 설정을 적용합니다.

Linux 구성 요소에는 Linux 인스턴스에 대해 다음 동작을 사용자 지정하는 데 도움이 되는 선택적 입력 파라미터가 있습니다.

- InstallPackages(문자열) 값이 인 경우 No구성 요소는 추가 소프트웨어 패키지를 설치하지 않습니다. 값이 인 경우 Yes구성 요소는 규정 준수를 극대화하는 데 필요한 추가 소프트웨어 패키지를 설치합니다. 기본값은 No입니다.
- SetDoDConsentBanner(문자열) 값이 인 경우 STIG Linux 구성 요소 중 하나가 설치된 인스턴스에 연결할 때 No DoD 동의 배너가 표시되지 않습니다. 값이 인 경우 STIG Linux 구성 요소 중 하나가 설치된 인스턴스에 연결할 때 로그인하기 전에 Yes DoD 동의 배너가 표시됩니다. 로그인하려면 먼저 배너를 승인해야 합니다. 기본값은 No입니다.

  동의 배너의 예는 DLA Document Services 웹 사이트에 액세스할 때 표시되는 면책 조항 국방부 개인 정보 보호 및 동의 공지를 참조하세요.

하드닝 구성 요소는 지원되는 STIG 설정을 다음과 같이 Linux 배포를 기반으로 하는 인프라에 적용합니다.

Red Hat Enterprise Linux(RHEL) 7 STIG 설정

- RHEL 7
- CentOS 7
- Amazon Linux 2(AL2)

RHEL 8 STIG 설정

- RHEL 8
- CentOS 8
- Amazon Linux 2023(AL 2023)

RHEL 9 STIG 설정

- RHEL 9

- CentOS Stream 9

STIG-Build-Linux-Low 버전 2025.1.x

다음 목록에는 하드닝 구성 요소가 인프라에 적용하는 STIG 설정이 나와 있습니다. 지원되는 설정이 인프라에 적용되지 않는 경우 하드닝 구성 요소는 해당 설정을 건너뛰고 계속 진행합니다. 예를 들어 일부 STIG 설정은 독립 실행형 서버에 적용되지 않을 수 있습니다. 조직별 정책도 하드닝 구성 요소가 적용되는 설정 종류(예: 관리자가 문서 설정을 검토하기 위한 요구 사항)에 영향을 미칠 수 있습니다.

전체 목록은 STIG 문서 라이브러리를 참조하십시오. 전체 목록을 보는 방법에 대한 자세한 내용은 STIG 보기 도구를 참조하세요.

RHEL 7 STIG 버전 3 릴리스 15

- RHEL 7/CentOS 7/AL2

  V-204452, V-204576 및 V-204605

RHEL 8 STIG 버전 2 릴리스 2

- RHEL 8/CentOS 8/AL 2023

  V-230241, V-230281, V-244527, V-230269, V-230270, V-230285, V-230346, V-230381, V-230395, V-230491, V-230468, V-230469, V-230485, V-230486, V-230494, V-230495, V-230496, V-230497, V-230498, V-230499 및 V-230253

RHEL 9 STIG 버전 2 릴리스 3

- RHEL 9/CentOS Stream 9

  V-257782, V-257824, V-258138, V-258037, V-257880, V-257795, V-257796, V-258173, V-258069, V-258076, V-258067, V-257946, V-257947

Ubuntu 18.04 STIG 버전 2 릴리스 15

V-219163, V-219164, V-219165, V-219172, V-219173, V-219174, V-219175, V-219178, V-219180, V-219210, V-219301, V-219327, V-219332 및 V-219333

Ubuntu 20.04 STIG 버전 2 릴리스 2

V-238203, V-238202, V-238234, V-238235, V-238237, V-238323, V-238373, V-238221, V-238222, V-238223, V-238224, V-238226, V-238362, V-238357 및 V-238308

Ubuntu 22.04 STIG 버전 2 릴리스 3

V-260479, V-260480, V-260481, V-260521, V-260520, V-260476, V-260472,V-260549, V-260550, V-260551, V-260552, V-260581 및 V-260596

Ubuntu 24.04 STIG 버전 1 릴리스 1

V-270645, V-270646, V-270664, V-270820, V-270677, V-270690, V-270706, V-270710, V-270695, V-270749, V-270752 및 V-270818

STIG-Build-Linux-Medium 버전 2025.1.x

다음 목록에는 하드닝 구성 요소가 인프라에 적용하는 STIG 설정이 나와 있습니다. 지원되는 설정이 인프라에 적용되지 않는 경우 하드닝 구성 요소는 해당 설정을 건너뛰고 계속 진행합니다. 예를 들어 일부 STIG 설정은 독립 실행형 서버에 적용되지 않을 수 있습니다. 조직별 정책도 하드닝 구성 요소가 적용되는 설정 종류(예: 관리자가 문서 설정을 검토하기 위한 요구 사항)에 영향을 미칠 수 있습니다.

전체 목록은 STIG 문서 라이브러리를 참조하십시오. 전체 목록을 보는 방법에 대한 자세한 내용은 STIG 보기 도구를 참조하세요.

> **Note**
>
> STIG-Build-Linux-Medium 강화 구성 요소에는 범주 II 취약성에 대해 특별히 나열된 STIG 설정 외에도 STIG-Build-Linux-Low 강화 구성 요소에 AWSTOE 적용되는 나열된 모든 STIG 설정이 포함됩니다.

RHEL 7 STIG 버전 3 릴리스 15

이 Linux 배포판의 카테고리 III(낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

- RHEL 7/CentOS 7/AL2

  V-204405, V-204406, V-204407, V-204408, V-204409, V-204410, V-204411, V-204412, V-204413, V-204414, V-204415, V-204416, V-204417, V-204418, V-204422, V-204423, V-204426, V-204427, V-204431, V-204434, V-204435, V-204437, V-204449, V-204450, V-204451, V-204457, V-204466, V-204490, V-204491, V-204503, V-204507, V-204508, V-204510, V-204511, V-204512, V-204514,

V-204515, V-204516, V-204517, V-204521, V-204524, V-204531, V-204536, V-204537, V-204538, V-204539, V-204540, V-204541, V-204542, V-204543, V-204544, V-204545, V-204546, V-204547, V-204548, V-204549, V-204550, V-204551, V-204552, V-204553, V-204554, V-204555, V-204556, V-204557, V-204558, V-204559, V-204560, V-204562, V-204563, V-204564, V-204565, V-204566, V-204567, V-204568, V-204572, V-204578, V-204579, V-204584, V-204585, V-204587, V-204588, V-204589, V-204590, V-204591, V-204592, V-204593, V-204595, V-204596, V-204597, V-204598, V-204599, V-204600, V-204601, V-204602, V-204609, V-204610, V-204611, V-204612, V-204613, V-204614, V-204615, V-204616, V-204617, V-204619, V-204622, V-204625, V-204630, V-204631, V-204633, V-233307, V-237634, V-237635, V-251703, V-255925, V-255927, V-255928, 및 V-256970

RHEL 8 STIG 버전 2 릴리스 2

이 Linux 배포판의 카테고리 III(낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

- RHEL 8/CentOS 8/AL 2023

V-230238, V-230239, V-230257, V-230258, V-230259, V-230262, V-230273, V-230275, V-230316, V-230325, V-230366, V-230478, V-230488, V-230489, V-230505, V-230523, V-230550, V-230559, V-230560, V-230561, V-237640, V-244547, V-256974, V-230525, V-244544, V-230248, V-230249, V-230250, V-230245, V-230246, V-230247, V-230397, V-230399, V-230400, V-230401, V-244543, V-230228, V-230298, V-230387, V-230482, V-230231, V-230233, V-230324, V-230365, V-230370, V-230373, V-230378, V-230383, V-230236, V-230314, V-230315, V-244523, V-257258, V-230266, V-230267, V-230268, V-230280, V-230310, V-230311, V-230312, V-230502, V-230532, V-230535, V-230536, V-230537, V-230538, V-230539, V-230540, V-230541, V-230542, V-230543, V-230544, V-230545, V-230546, V-230547, V-230548, V-230549, V-244550, V-244551, V-244552, V-244553, V-244554, V-250317, V-251718, V-230237, V-230313, V-230356, V-230357, V-230358, V-230359, V-230360, V-230361, V-230362, V-230363, V-230368, V-230369, V-230375, V-230376, V-230377, V-244524, V-244533, V-251713, V-251717, V-251714, V-251715, V-251716, V-230332, V-230333, V-230335, V-230337, V-230339, V-230341, V-230343, V-230345, V-230240, V-230282, V-250315, V-250316, V-230255, V-230277, V-230278, V-230348, V-230353, V-230386, V-230390, V-230392, V-230394, V-230396, V-230393, V-230398, V-230402, V-230403, V-230404, V-230405, V-230406, V-230407, V-230408, V-230409, V-230410, V-230411, V-230412, V-230413, V-230418, V-230419, V-230421, V-230422, V-230423, V-230424, V-230425, V-230426, V-230427, V-230428, V-230429, V-230430, V-230431, V-230432, V-230433, V-230434, V-230435, V-230436, V-230437, V-230438, V-230439, V-230444, V-230446, V-230447, V-230448, V-230449, V-230455, V-230456, V-230462,

V-230463, V-230464, V-230465, V-230466, V-230467, V-230471, V-230472, V-230473, V-230474, V-230480, V-230483, V-244542, V-230503, V-230507, V-244525, V-230244, V-230286, V-230287, V-230288, V-230290, V-230291, V-230296, V-230330, V-230382, V-230526, V-230527, V-230555, V-230556, V-244526, V-244528, V-237642, V-237643, 및 V-251711

RHEL 9 STIG 버전 2 릴리스 3

이 Linux 배포판의 카테고리 III(낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

- RHEL 9/CentOS Stream 9

V-257981, V-257780, V-257825, V-257827, V-257828, V-257829, V-257830, V-257831, V-257832, V-257833, V-257834, V-257836, V-257838, V-257839, V-257840, V-257841, V-257842, V-257882, V-257883, V-257884, V-257885, V-257889, V-257890, V-257891, V-257892, V-257893, V-257894, V-257895, V-257896, V-257897, V-257898, V-257899, V-257900, V-257901, V-257902, V-257903, V-257904, V-257905, V-257906, V-257907, V-257908, V-257909, V-257910, V-257911, V-257912, V-257913, V-257914, V-257915, V-257918, V-257919, V-257920, V-257921, V-257922, V-257923, V-257928, V-257929, V-257930, V-257934, V-257935, V-257943, V-257948, V-257954, V-257978, V-257980, V-258035, V-258041, V-258046, V-258052, V-258063, V-258068, V-258081, V-258082, V-258083, V-258089, V-258120, V-258124, V-258126, V-258140, V-258141, V-258151, V-258234, V-257886, V-257916, V-257917, V-257952, V-257936, V-257939, V-257940, V-258036, V-258038, V-257887, V-257924, V-257925, V-258145, V-258152, V-258153, V-258154, V-258156, V-258157, V-258158, V-258159, V-258160, V-258161, V-258162, V-258163, V-258164, V-258165, V-258166, V-258167, V-258168, V-258169, V-258170, V-258171, V-258172, V-258176, V-258177, V-258178, V-258179, V-258180, V-258181, V-258182, V-258183, V-258184, V-258185, V-258186, V-258187, V-258188, V-258189, V-258190, V-258191, V-258192, V-258193, V-258194, V-258195, V-258196, V-258197, V-258198, V-258199, V-258200, V-258201, V-258202, V-258203, V-258204, V-258205, V-258206, V-258207, V-258208, V-258209, V-258210, V-258211, V-258212, V-258213, V-258214, V-258215, V-258216, V-258217, V-258218, V-258219, V-258220, V-258221, V-258222, V-258223, V-258224, V-258225, V-258226, V-258227, V-258228, V-258229, V-257781, V-257783, V-257786, V-257797, V-257798, V-257799, V-257800, V-257801, V-257802, V-257803, V-257809, V-257810, V-257811, V-257812, V-257813, V-257815, V-257816, V-257942, V-257957, V-257958, V-257959, V-257960, V-257961, V-257962, V-257963, V-257964, V-257965, V-257966, V-257967, V-257968, V-257969, V-257970, V-257971, V-257972, V-257973, V-257974, V-257975, V-257976, V-257977, V-258077, V-258128, V-258129, V-258043, V-258049, V-258071, V-258072, V-258073, V-258074, V-258075, V-258104, V-258105, V-258107, V-258108, V-258109, V-258110, V-258111, V-258117,

V-258119, V-257979, V-257982, V-257983, V-257985, V-257987, V-257988, V-257992, V-257993, V-257994, V-257995, V-257996, V-257997, V-257998, V-257999, V-258000, V-258001, V-258002, V-258003, V-258004, V-258005, V-258006, V-258007, V-258008, V-258009, V-258010, V-258011, V-258130, V-257814, V-258054, V-258055, V-258056, V-258057, V-258060, V-258070, V-258080, V-258088, V-258091, V-258092, V-258093, V-258095, V-258097, V-258098, V-258099, V-258100, V-258101, V-258102, V-258103, V-258112, V-258113, V-258114, V-258115, V-258116, V-258118, V-258233, V-257951, V-257953, V-258142, V-258144, V-258146, V-258147, V-258148, V-258150, V-258064, V-258065, V-258066, V-258232, V-258237, V-258239, V-258240, V-257788, V-257790, V-257791, V-257792, V-257793, V-257794, V-257817, V-258175, V-257804, V-257805, V-257806, V-257807, V-257808, V-258034, V-258039, V-257888, V-257926, V-257927, V-257933, V-258084, V-258085, V-257949, V-258040, V-257818, V-257849, V-258122, V-258123, V-258133, V-257944, V-258028, V-258079, V-258125, 및 V-258137

Ubuntu 18.04 STIG 버전 2 릴리스 15

이 Linux 배포판의 카테고리 III(낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-219149, V-219155, V-219156, V-219160, V-219166, V-219168, V-219176, V-219181, V-219184, V-219186, V-219188, V-219189, V-219190, V-219191, V-219192, V-219193, V-219194, V-219195, V-219196, V-219197, V-219198, V-219199, V-219200, V-219201, V-219202, V-219203, V-219204, V-219205, V-219206, V-219207, V-219208, V-219209, V-219213, V-219214, V-219215, V-219216, V-219217, V-219218, V-219219, V-219220, V-219221, V-219222, V-219223, V-219224, V-219225, V-219226, V-219227, V-219228, V-219229, V-219230, V-219231, V-219232, V-219233, V-219234, V-219235, V-219236, V-219238, V-219239, V-219240, V-219241, V-219242, V-219243, V-219244, V-219250, V-219254, V-219257, V-219263, V-219264, V-219265, V-219266, V-219267, V-219268, V-219269, V-219270, V-219271, V-219272, V-219273, V-219274, V-219275, V-219276, V-219277, V-219279, V-219281, V-219287, V-219291, V-219297, V-219298, V-219299, V-219300, V-219303, V-219304, V-219306, V-219309, V-219310, V-219311, V-219312, V-219315, V-219318, V-219319, V-219323, V-219326, V-219328, V-219330, V-219331, V-219335, V-219336, V-219337, V-219338, V-219339, V-219342, V-219344, V-233779, V-233780, 및 V-255906

Ubuntu 20.04 STIG 버전 2 릴리스 2

이 Linux 배포판의 카테고리 III(낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-238200, V-238205, V-238207, V-238231, V-238329, V-238337, V-238339, V-238340, V-238344, V-238345, V-238346, V-238347, V-238348, V-238349, V-238350, V-238351, V-238352, V-238371, V-238376, V-238377, V-238378, V-238209, V-238325, V-238330, V-238333, V-238369, V-238230, V-238338, V-238341, V-238342, V-238343, V-238324, V-238353, V-238228, V-238225, V-238227, V-238299, V-238238, V-238239, V-238240, V-238241, V-238242, V-238244, V-238245, V-238246, V-238247, V-238248, V-238249, V-238250, V-238251, V-238252, V-238253, V-238254, V-238255, V-238256, V-238257, V-238258, V-238264, V-238268, V-238271, V-238277, V-238278, V-238279, V-238280, V-238281, V-238282, V-238283, V-238284, V-238285, V-238286, V-238287, V-238288, V-238289, V-238290, V-238291, V-238292, V-238293, V-238294, V-238295, V-238297, V-238300, V-238301, V-238302, V-238304, V-238309, V-238310, V-238315, V-238316, V-238317, V-238318, V-238319, V-238320, V-251505, V-238360, V-238210, V-238211, V-238212, V-238213, V-238220, V-255912, V-238355, V-238236, V-238303, V-238356, V-238359, V-238370, V-238334, 및 V-238232

Ubuntu 22.04 STIG 버전 2 릴리스 3

이 Linux 배포판의 카테고리 III(낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-260546, V-260473, V-260474, V-260478, V-260485, V-260486, V-260487, V-260488, V-260489, V-260490, V-260495, V-260496, V-260497, V-260498, V-260499, V-260500, V-260508, V-260513, V-260522, V-260540, V-260542, V-260543, V-260547, V-260554, V-260572, V-260590, V-260475, V-260491, V-260510, V-260511, V-260492, V-260493, V-260494, V-260507, V-260591, V-260594, V-260597, V-260598, V-260599, V-260600, V-260601, V-260602, V-260603, V-260604, V-260605, V-260606, V-260607, V-260608, V-260609, V-260610, V-260611, V-260612, V-260613, V-260614, V-260615, V-260616, V-260617, V-260618, V-260619, V-260620, V-260621, V-260622, V-260623, V-260624, V-260625, V-260626, V-260627, V-260628, V-260629, V-260630, V-260631, V-260632, V-260633, V-260634, V-260635, V-260636, V-260637, V-260638, V-260639, V-260640, V-260641, V-260642, V-260643, V-260644, V-260645, V-260646, V-260647, V-260648, V-260649, V-260471, V-260514, V-260553, V-260573, V-260576, V-260574, V-260560, V-260561, V-260562, V-260563, V-260564, V-260565, V-260566, V-260567, V-260569, V-260527, V-260528, V-260530, V-260533, V-260534, V-260575, V-260505, V-260506, V-260512, V-260582, V-260584, V-260585, V-260586, V-260477, V-260545, V-260555, V-260556, V-260557, V-260509, V-260588, V-260589, V-260537, V-260538, 및 V-260535

Ubuntu 24.04 STIG 버전 1 릴리스 1

이 Linux 배포판의 카테고리 III(낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-270649, V-270651, V-270652, V-270653, V-270654, V-270656, V-270657, V-270659, V-270660, V-270661, V-270662, V-270663, V-270669, V-270672, V-270673, V-270674, V-270676, V-270678, V-270679, V-270680, V-270681, V-270683, V-270684, V-270685, V-270686, V-270687, V-270688, V-270689, V-270692, V-270693, V-270696, V-270697, V-270698, V-270699, V-270700, V-270701, V-270702, V-270703, V-270704, V-270705, V-270709, V-270715, V-270716, V-270718, V-270720, V-270721, V-270722, V-270723, V-270724, V-270725, V-270726, V-270727, V-270728, V-270729, V-270730, V-270731, V-270732, V-270733, V-270737, V-270739, V-270740, V-270741, V-270742, V-270743, V-270746, V-270750, V-270753, V-270755, V-270756, V-270757, V-270758, V-270759, V-270760, V-270765, V-270766, V-270767, V-270768, V-270769, V-270770, V-270771, V-270772, V-270773, V-270775, V-270776, V-270777, V-270778, V-270779, V-270780, V-270781, V-270782, V-270783, V-270784, V-270785, V-270786, V-270787, V-270788, V-270789, V-270790, V-270791, V-270792, V-270793, V-270794, V-270795, V-270796, V-270797, V-270798, V-270799, V-270800, V-270801, V-270802, V-270803, V-270804, V-270805, V-270806, V-270807, V-270808, V-270809, V-270810, V-270811, V-270812, V-270813, V-270814, V-270815, V-270821, V-270822, V-270823, V-270824, V-270825, V-270826, V-270827, V-270828, V-270829, V-270830, V-270831, 및 V-270832

STIG-Build-Linux-High 버전 2025.1.x

다음 목록에는 하드닝 구성 요소가 인프라에 적용하는 STIG 설정이 나와 있습니다. 지원되는 설정이 인프라에 적용되지 않는 경우 하드닝 구성 요소는 해당 설정을 건너뛰고 계속 진행합니다. 예를 들어 일부 STIG 설정은 독립 실행형 서버에 적용되지 않을 수 있습니다. 조직별 정책도 하드닝 구성 요소가 적용되는 설정 종류(예: 관리자가 문서 설정을 검토하기 위한 요구 사항)에 영향을 미칠 수 있습니다.

전체 목록은 STIG 문서 라이브러리를 참조하십시오. 전체 목록을 보는 방법에 대한 자세한 내용은 STIG 보기 도구를 참조하세요.

> **Note**
>
> STIG-Build-Linux-High 강화 구성 요소에는 STIG-Build-Linux-Low 및 STIG-Build-Linux-Medium 강화 구성 요소에 AWSTOE 적용되는 나열된 모든 STIG 설정과 범주 I 취약성에 특별히 적용되는 나열된 STIG 설정이 포함됩니다.

RHEL 7 STIG 버전 3 릴리스 15

이 Linux 배포판의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

- RHEL 7/CentOS 7/AL2

V-204424, V-204425, V-204442, V-204443, V-204447, V-204448, V-204455, V-204497, V-204502, V-204594, V-204620 및 V-204621

RHEL 8 STIG 버전 2 릴리스 2

이 Linux 배포판의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

- RHEL 8/CentOS 8/AL 2023

  V-230223, V-230265, V-230529, V-230531, V-230264, V-230487, V-230492, V-230533, V-230558 및 V-244540

RHEL 9 STIG 버전 2 릴리스 3

이 Linux 배포판의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

- RHEL 9/CentOS Stream 9

  V-257820, V-257821, V-257826, V-257835, V-257955, V-257956, V-258059, V-258230, V-258238, V-257784, V-257785, V-257984, V-257986, V-258078, V-258094 및 V-258235

Ubuntu 18.04 STIG 버전 2 릴리스 15

이 Linux 배포판의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-219157, V-219158, V-219177, V-219212, V-219308, V-219314, V-219316, V-251507 및 V-264388

Ubuntu 20.04 STIG 버전 2 릴리스 2

이 Linux 배포판의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-238218, V-238219, V-238201, V-251504, V-238326, V-238327 및 V-238380

Ubuntu 22.04 STIG 버전 2 릴리스 3

이 Linux 배포판의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-260482, V-260483, V-260523, V-260524, V-260469, V-260526, V-260529, V-260570, V-260571, V-260579 및 V-260539

Ubuntu 24.04 STIG 버전 1 릴리스 1

이 Linux 배포판의 카테고리 II 및 III(중간 및 낮음) 취약성에 대해 하드닝 구성 요소가 적용하는 지원되는 모든 STIG 설정과 함께 다음이 포함됩니다.

V-270647, V-270648, V-270665, V-270666, V-270708, V-270717, V-270711, V-270712, V-270713 및 V-270714

Linux용 STIG 버전 이력 로그

이 섹션에서는 Linux 구성 요소 버전 이력을 기록합니다. 한 분기의 변경 사항 및 게시된 버전을 보려면 제목을 선택하여 정보를 확장하십시오.

2025년 Q1 변경 사항 - 04/11/2025:

다음 STIG 버전을 업데이트하고, 2025년 1분기 릴리스에 STIGS를 적용하고, Ubuntu 24.04에 대한 지원을 추가했습니다.

STIG-Build-Linux-Low 버전 2025.1.x

- RHEL 7 STIG 버전 3 릴리스 15
- RHEL 8 STIG 버전 2 릴리스 2
- RHEL 9 STIG 버전 2 릴리스 3
- Ubuntu 18.04 STIG 버전 2 릴리스 15
- Ubuntu 20.04 STIG 버전 2 릴리스 2
- Ubuntu 22.04 STIG 버전 2 릴리스 3
- Ubuntu 24.04 STIG 버전 1 릴리스 1

STIG-Build-Linux-Medium 버전 2025.1.x

- RHEL 7 STIG 버전 3 릴리스 15
- RHEL 8 STIG 버전 2 릴리스 2
- RHEL 9 STIG 버전 2 릴리스 3

- Ubuntu 18.04 STIG 버전 2 릴리스 15
- Ubuntu 20.04 STIG 버전 2 릴리스 2
- Ubuntu 22.04 STIG 버전 2 릴리스 3
- Ubuntu 24.04 STIG 버전 1 릴리스 1

STIG-Build-Linux-High 버전 2025.1.x

- RHEL 7 STIG 버전 3 릴리스 15
- RHEL 8 STIG 버전 2 릴리스 2
- RHEL 9 STIG 버전 2 릴리스 3
- Ubuntu 18.04 STIG 버전 2 릴리스 15
- Ubuntu 20.04 STIG 버전 2 릴리스 2
- Ubuntu 22.04 STIG 버전 2 릴리스 3
- Ubuntu 24.04 STIG 버전 1 릴리스 1

2024년 Q4 변경 사항 - 12/10/2024:

다음 STIG 버전을 업데이트하고, 2024년 4분기 릴리스에 STIGS를 적용하고, Linux 구성 요소의 두 가지 새로운 입력 파라미터에 대한 정보를 추가했습니다.

STIG-Build-Linux-Low 버전 2024.4.x

- RHEL 7 STIG 버전 3 릴리스 15
- RHEL 8 STIG 버전 2 릴리스 1
- RHEL 9 STIG 버전 2 릴리스 2
- Ubuntu 18.04 STIG 버전 2 릴리스 15
- Ubuntu 20.04 STIG 버전 2 릴리스 1
- Ubuntu 22.04 STIG 버전 2 릴리스 2

STIG-Build-Linux-Medium 버전 2024.4.x

- RHEL 7 STIG 버전 3 릴리스 15
- RHEL 8 STIG 버전 2 릴리스 1
- RHEL 9 STIG 버전 2 릴리스 2

- Ubuntu 18.04 STIG 버전 2 릴리스 15
- Ubuntu 20.04 STIG 버전 2 릴리스 1
- Ubuntu 22.04 STIG 버전 2 릴리스 2

STIG-Build-Linux-High 버전 2024.4.x

- RHEL 7 STIG 버전 3 릴리스 15
- RHEL 8 STIG 버전 2 릴리스 1
- RHEL 9 STIG 버전 2 릴리스 2
- Ubuntu 18.04 STIG 버전 2 릴리스 15
- Ubuntu 20.04 STIG 버전 2 릴리스 1
- Ubuntu 22.04 STIG 버전 2 릴리스 2

2024년 Q3 변경 사항 - 10/04/2024(변경 없음):

2024년 3분기 릴리스의 Linux 구성 요소 STIGS에는 변경 사항이 없습니다.

2024년 2분기 변경 사항 - 2024년 5월 10일:

STIG 버전을 업데이트하고 2024년 2분기 릴리스에 STIG를 적용했습니다. 또한 다음과 같이 RHEL 9, CentOS Stream 9 및 Ubuntu 22.04에 대한 지원이 추가되었습니다.

STIG-Build-Linux-Low 버전 2024.2.x

- RHEL 7 STIG 버전 3 릴리스 14
- RHEL 8 STIG 버전 1 릴리스 14
- RHEL 9 STIG 버전 1 릴리스 3
- Ubuntu 18.04 STIG 버전 2 릴리스 14
- Ubuntu 20.04 STIG 버전 1 릴리스 12
- Ubuntu 22.04 STIG 버전 1 릴리스 1

STIG-Build-Linux-Medium 버전 2024.2.x

- RHEL 7 STIG 버전 3 릴리스 14
- RHEL 8 STIG 버전 1 릴리스 14

- RHEL 9 STIG 버전 1 릴리스 3
- Ubuntu 18.04 STIG 버전 2 릴리스 14
- Ubuntu 20.04 STIG 버전 1 릴리스 12
- Ubuntu 22.04 STIG 버전 1 릴리스 1

STIG-Build-Linux-High 버전 2024.2.x

- RHEL 7 STIG 버전 3 릴리스 14
- RHEL 8 STIG 버전 1 릴리스 14
- RHEL 9 STIG 버전 1 릴리스 3
- Ubuntu 18.04 STIG 버전 2 릴리스 14
- Ubuntu 20.04 STIG 버전 1 릴리스 12
- Ubuntu 22.04 STIG 버전 1 릴리스 1

2024년 1분기 변경 사항 -2024년 2월 6일:

STIG 버전을 업데이트하고 2024년 1분기 릴리스에 다음과 같이 STIG를 적용했습니다.

STIG-Build-Linux-Low 버전 2024.1.x

- RHEL 7 STIG 버전 3 릴리스 14
- RHEL 8 STIG 버전 1 릴리스 13
- Ubuntu 18.04 STIG 버전 2 릴리스 13
- Ubuntu 20.04 STIG 버전 1 릴리스 11

STIG-Build-Linux-Medium 버전 2024.1.x

- RHEL 7 STIG 버전 3 릴리스 14
- RHEL 8 STIG 버전 1 릴리스 13
- Ubuntu 18.04 STIG 버전 2 릴리스 13
- Ubuntu 20.04 STIG 버전 1 릴리스 11

STIG-Build-Linux-High 버전 2024.1.x

- RHEL 7 STIG 버전 3 릴리스 14

- RHEL 8 STIG 버전 1 릴리스 13
- Ubuntu 18.04 STIG 버전 2 릴리스 13
- Ubuntu 20.04 STIG 버전 1 릴리스 11

2023년 4분기 변경 사항 - 2023년 12월 7일:

STIG 버전을 업데이트하고 2023년 4분기 릴리스에 다음과 같이 STIG를 적용했습니다.

STIG-Build-Linux-Low 버전 2023.4.x

- RHEL 7 STIG 버전 3 릴리스 13
- RHEL 8 STIG 버전 1 릴리스 12
- Ubuntu 18.04 STIG 버전 2 릴리스 12
- Ubuntu 20.04 STIG 버전 1 릴리스 10

STIG-Build-Linux-Medium 버전 2023.4.x

- RHEL 7 STIG 버전 3 릴리스 13
- RHEL 8 STIG 버전 1 릴리스 12
- Ubuntu 18.04 STIG 버전 2 릴리스 12
- Ubuntu 20.04 STIG 버전 1 릴리스 10

STIG-Build-Linux-High 버전 2023.4.x

- RHEL 7 STIG 버전 3 릴리스 13
- RHEL 8 STIG 버전 1 릴리스 12
- Ubuntu 18.04 STIG 버전 2 릴리스 12
- Ubuntu 20.04 STIG 버전 1 릴리스 10

2023년 3분기 변경 사항 - 2023년 10월 4일:

2023년 3분기 릴리스에 대한 STIG 버전 업데이트 및 STIG 적용 내용은 다음과 같습니다.

STIG-Build-Linux-Low 버전 2023.3.x

- RHEL 7 STIG 버전 3 릴리스 12

- RHEL 8 STIG 버전 1 릴리스 11
- Ubuntu 18.04 STIG 버전 2 릴리스 11
- Ubuntu 20.04 STIG 버전 1 릴리스 9

STIG-Build-Linux-Medium 버전 2023.3.x

- RHEL 7 STIG 버전 3 릴리스 12
- RHEL 8 STIG 버전 1 릴리스 11
- Ubuntu 18.04 STIG 버전 2 릴리스 11
- Ubuntu 20.04 STIG 버전 1 릴리스 9

STIG-Build-Linux-High 버전 2023.3.x

- RHEL 7 STIG 버전 3 릴리스 12
- RHEL 8 STIG 버전 1 릴리스 11
- Ubuntu 18.04 STIG 버전 2 릴리스 11
- Ubuntu 20.04 STIG 버전 1 릴리스 9

2023년 2분기 변경 사항 - 2023년 5월 3일:

2023년 2분기 릴리스에 대한 STIG 버전 업데이트 및 STIG 적용 내용은 다음과 같습니다.

STIG-Build-Linux-Low 버전 2023.2.x

- RHEL 7 STIG 버전 3 릴리스 11
- RHEL 8 STIG 버전 1 릴리스 10
- Ubuntu 18.04 STIG 버전 2 릴리스 11
- Ubuntu 20.04 STIG 버전 1 릴리스 8

STIG-Build-Linux-Medium 버전 2023.2.x

- RHEL 7 STIG 버전 3 릴리스 11
- RHEL 8 STIG 버전 1 릴리스 10
- Ubuntu 18.04 STIG 버전 2 릴리스 11
- Ubuntu 20.04 STIG 버전 1 릴리스 8

STIG-Build-Linux-High 버전 2023.2.x

- RHEL 7 STIG 버전 3 릴리스 11
- RHEL 8 STIG 버전 1 릴리스 10
- Ubuntu 18.04 STIG 버전 2 릴리스 11
- Ubuntu 20.04 STIG 버전 1 릴리스 8

2023년 1분기 변경 사항 - 2023년 3월 27일:

2023년 1분기 릴리스에 대한 STIG 버전 업데이트 및 STIG 적용 내용은 다음과 같습니다.

STIG-Build-Linux-Low 버전 2023.1.x

- RHEL 7 STIG 버전 3 릴리스 10
- RHEL 8 STIG 버전 1 릴리스 9
- Ubuntu 18.04 STIG 버전 2 릴리스 10
- Ubuntu 20.04 STIG 버전 1 릴리스 7

STIG-Build-Linux-Medium 버전 2023.1.x

- RHEL 7 STIG 버전 3 릴리스 10
- RHEL 8 STIG 버전 1 릴리스 9
- Ubuntu 18.04 STIG 버전 2 릴리스 10
- Ubuntu 20.04 STIG 버전 1 릴리스 7

STIG-Build-Linux-High 버전 2023.1.x

- RHEL 7 STIG 버전 3 릴리스 10
- RHEL 8 STIG 버전 1 릴리스 9
- Ubuntu 18.04 STIG 버전 2 릴리스 10
- Ubuntu 20.04 STIG 버전 1 릴리스 7

2022년 4분기 변경 사항 - 2023년 2월 1일:

2022년 4분기 릴리스에 대한 STIG 버전 업데이트 및 STIG 적용 내용은 다음과 같습니다.

STIG-Build-Linux-Low 버전 2022.4.x

- RHEL 7 STIG 버전 3 릴리스 9
- RHEL 8 STIG 버전 1 릴리스 8
- Ubuntu 18.04 STIG 버전 2 릴리스 9
- Ubuntu 20.04 STIG 버전 1 릴리스 6

STIG-Build-Linux-Medium 버전 2022.4.x

- RHEL 7 STIG 버전 3 릴리스 9
- RHEL 8 STIG 버전 1 릴리스 8
- Ubuntu 18.04 STIG 버전 2 릴리스 9
- Ubuntu 20.04 STIG 버전 1 릴리스 6

STIG-Build-Linux-High 버전 2022.4.x

- RHEL 7 STIG 버전 3 릴리스 9
- RHEL 8 STIG 버전 1 릴리스 8
- Ubuntu 18.04 STIG 버전 2 릴리스 9
- Ubuntu 20.04 STIG 버전 1 릴리스 6

2022년 3분기 변경 사항 - 2022년 9월 30일(변경 없음):

2022년 3분기 릴리스의 Linux 구성 요소 STIG에는 변경 사항이 없습니다.

2022년 2분기 변경 사항 - 2022년 8월 2일:

다음과 같이 Ubuntu 지원을 도입하고, STIG 버전을 업데이트하고, 2022년 2분기 릴리스에 STIG를 적용했습니다.

STIG-Build-Linux-Low 버전 2022.x

- RHEL 7 STIG 버전 3 릴리스 7
- RHEL 8 STIG 버전 1 릴리스 6
- Ubuntu 18.04 STIG 버전 2 릴리스 6(신규)

- Ubuntu 20.04 STIG 버전 1 릴리스 4(신규)

STIG-Build-Linux-Medium 버전 2022.2.x

- RHEL 7 STIG 버전 3 릴리스 7
- RHEL 8 STIG 버전 1 릴리스 6
- Ubuntu 18.04 STIG 버전 2 릴리스 6(신규)
- Ubuntu 20.04 STIG 버전 1 릴리스 4(신규)

STIG-Build-Linux-High 버전 2022.x

- RHEL 7 STIG 버전 3 릴리스 7
- RHEL 8 STIG 버전 1 릴리스 6
- Ubuntu 18.04 STIG 버전 2 릴리스 6(신규)
- Ubuntu 20.04 STIG 버전 1 릴리스 4(신규)

2022년 1분기 변경 사항 - 2022년 4월 26일:

리팩터링하여 컨테이너에 대해 더 나은 지원을 포함했습니다. 이전 AL2 스크립트를 RHEL 7과 결합했습니다. 다음과 같이 STIG 버전을 업데이트하고 2022년 1분기 릴리스에 STIG를 적용했습니다.

STIG-Build-Linux-Low 버전 3.6.x

- RHEL 7 STIG 버전 3 릴리스 6
- RHEL 8 STIG 버전 1 릴리스 5

STIG-Build-Linux-Medium 버전 3.6.x

- RHEL 7 STIG 버전 3 릴리스 6
- RHEL 8 STIG 버전 1 릴리스 5

STIG-Build-Linux-High 버전 3.6.x

- RHEL 7 STIG 버전 3 릴리스 6
- RHEL 8 STIG 버전 1 릴리스 5

2021년 4분기 변경 사항 - 2021년 12월 20일:

STIG 버전이 업데이트되고 2021년 4분기 릴리스에 STIG가 다음과 같이 적용되었습니다.

STIG-Build-Linux-Low 버전 3.5.x

- RHEL 7 STIG 버전 3 릴리스 5
- RHEL 8 STIG 버전 1 릴리스 4

STIG-Build-Linux-Medium 버전 3.5.x

- RHEL 7 STIG 버전 3 릴리스 5
- RHEL 8 STIG 버전 1 릴리스 4

STIG-Build-Linux-High 버전 3.5.x

- RHEL 7 STIG 버전 3 릴리스 5
- RHEL 8 STIG 버전 1 릴리스 4

2021년 3분기 변경 사항 - 2021년 9월 30일:

STIG 버전이 업데이트되고 2021년 3분기 릴리스에 STIG가 다음과 같이 적용되었습니다.

STIG-Build-Linux-Low 버전 3.4.x

- RHEL 7 STIG 버전 3 릴리스 4
- RHEL 8 STIG 버전 1 릴리스 3

STIG-Build-Linux-Medium 버전 3.4.x

- RHEL 7 STIG 버전 3 릴리스 4
- RHEL 8 STIG 버전 1 릴리스 3

STIG-Build-Linux-High 버전 3.4.x

- RHEL 7 STIG 버전 3 릴리스 4
- RHEL 8 STIG 버전 1 릴리스 3

SCAP 규정 준수 검증기 구성 요소

보안 콘텐츠 자동화 프로토콜(SCAP)은 IT 전문가가 규정 준수를 위한 애플리케이션 보안 취약성을 식별하는 데 사용할 수 있는 일련의 표준입니다. SCAP 규정 준수 검사기(SCC)는 미국 해군 정보 전쟁 센터(NIWC) Atlantic에서 출시한 SCAP 검증을 거친 검사 도구입니다. 자세한 내용은 NIWC Atlantic 웹 사이트의 보안 콘텐츠 자동화 프로토콜(SCAP) 규정 준수 검사기(SCC)를 참조하십시오.

AWSTOE `scap-compliance-checker-windows` 및 `scap-compliance-checker-linux` 구성 요소는 파이프라인 빌드 및 테스트 인스턴스에 SCC 스캐너를 다운로드하여 설치합니다. 스캐너가 실행되면 DISA SCAP 벤치마크를 사용하여 인증된 구성 스캔을 수행하고 다음 정보가 포함된 보고서를 제공합니다. AWSTOE 또한는 애플리케이션 로그에 정보를 기록합니다.

- 인스턴스에 적용되는 STIG 설정.
- 인스턴스의 전체 규정 준수 점수.

정확한 규정 준수 검증 결과를 보고하려면 빌드 프로세스의 마지막 단계로 SCAP 검증을 실행하는 것이 좋습니다.

> **Note**
>
> STIG 보기 도구 중 하나를 사용하여 보고서를 검토할 수 있습니다. 이러한 도구는 DoD 사이버 거래소를 통해 온라인으로 제공됩니다.

다음 섹션에서는 SCAP 검증 구성 요소에 포함된 벤치마크에 대해 설명합니다.

scap-compliance-checker-windows 버전 2024.03.0

구성 `scap-compliance-checker-windows` 요소는 Image Builder가 image. AWSTOE logs를 빌드하고 테스트하기 위해 생성하는 EC2 인스턴스에서 실행되며 SCC 애플리케이션이 생성하는 보고서와 점수를 모두 기록합니다.

구성 요소는 다음 워크플로우 단계를 수행합니다.

1. SCC 애플리케이션을 다운로드하고 설치합니다.
2. 규정 준수 벤치마크를 가져옵니다.
3. SCC 애플리케이션을 사용하여 검증을 실행합니다.
4. 규정 준수 보고서와 점수를 빌드 인스턴스 데스크톱에 로컬로 저장합니다.
5. 로컬 보고서의 규정 준수 점수를 AWSTOE 애플리케이션 로그 파일에 로깅합니다.

> **Note**
>
> AWSTOE 는 현재 Windows Server 2012 R2 MS, 2016, 2019 및 2022에 대한 SCAP 규정 준수 검증을 지원합니다.

Windows용 SCAP 규정 준수 검사기 구성 요소에는 다음과 같은 벤치마크가 포함됩니다.

SCC 버전: 5.10

2023년 Q4 벤치마크:

- U_MS_Defender_Antivirus_V2R5_STIG_SCAP_1-2_Benchmark
- U_MS_DotNet_Framework_4-0_V2R2_STIG_SCAP_1-2_Benchmark
- U_MS_IE11_V2R6_STIG_SCAP_1-2_벤치마크
- U_MS_Windows_2012_and_2012_R2_DC_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_2012_and_2012_R2_MS_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Defender_Firewall_V2R3_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2016_V2R7_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2019_V3R2_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2022_V2R2_STIG_SCAP_1-2_Benchmark
- U_CAN_Ubuntu_20-04_LTS_V1R10_STIG_SCAP_1-2_Benchmark
- U_RHEL_7_V3R15_STIG_SCAP_1-3_벤치마크
- U_RHEL_8_V1R13_STIG_SCAP_1-3_벤치마크
- U_RHEL_9_V2R1_STIG_SCAP_1-3_벤치마크

scap-compliance-checker-linux 버전 2021.04.0

구성 `scap-compliance-checker-linux` 요소는 Image Builder가 image. AWSTOE logs를 빌드하고 테스트하기 위해 생성하는 EC2 인스턴스에서 실행되며 SCC 애플리케이션이 생성하는 보고서와 점수를 모두 기록합니다.

구성 요소는 다음 워크플로우 단계를 수행합니다.

1. SCC 애플리케이션을 다운로드하고 설치합니다.
2. 규정 준수 벤치마크를 가져옵니다.

3. SCC 애플리케이션을 사용하여 검증을 실행합니다.
4. 규정 준수 보고서와 점수를 빌드 인스턴스의 다음 `/opt/scc/SCCResults` 위치에 로컬로 저장합니다.
5. 로컬 보고서의 규정 준수 점수를 AWSTOE 애플리케이션 로그 파일에 로깅합니다.

> **Note**
>
> AWSTOE 는 현재 RHEL 7/8 및 Ubuntu 18.04/20.04에 대한 SCAP 규정 준수 검증을 지원합니다. SCC 애플리케이션은 현재 검증을 위한 x86 아키텍처를 지원합니다.

Linux용 SCAP 규정 준수 검사기 구성 요소에는 다음과 같은 벤치마크가 포함됩니다.

SCC 버전: 5.10

2023년 Q4 벤치마크:

- U_CAN_Ubuntu_20-04_LTS_V1R10_STIG_SCAP_1-2_Benchmark
- U_RHEL_7_V3R15_STIG_SCAP_1-3_벤치마크
- U_RHEL_8_V1R13_STIG_SCAP_1-3_벤치마크
- U_RHEL_9_V2R1_STIG_SCAP_1-3_벤치마크
- U_MS_Defender_Antivirus_V2R5_STIG_SCAP_1-2_Benchmark
- U_MS_DotNet_Framework_4-0_V2R2_STIG_SCAP_1-2_Benchmark
- U_MS_IE11_V2R6_STIG_SCAP_1-2_벤치마크
- U_MS_Windows_2012_and_2012_R2_DC_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_2012_and_2012_R2_MS_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Defender_Firewall_V2R3_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2016_V2R7_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2019_V3R2_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2022_V2R2_STIG_SCAP_1-2_Benchmark

SCAP 버전 이력

다음 표에서는 이 설명서에서 기술한 SCAP 환경 및 설정의 중요 변경사항에 대해 설명합니다.

변경 사항	설명	날짜
2025년 Q1 SCAP 업데이트	• scap-compliance-checker-windows 버전 2024.03.0 (SCC 버전: 5.10) • scap-compliance-checker-windows 버전 2025.01.0 (SCC 버전: 5.10	2025년 4월 11일
2023년 Q4 SCAP 업데이트	• scap-compliance-checker-windows 버전 2023.04.0 (SCC 버전: 5.8) • scap-compliance-checker-windows 버전 2023.04.0 (SCC 버전: 5.8)	2021년 12월 20일
2023년 Q3 SCAP 업데이트	• scap-compliance-checker-windows 버전 2023.03.0 (SCC 버전: 5.7.2) • scap-compliance-checker-linux 버전 2023.03.0(SCC 버전: 5.7.2	2023년 11월 13일
SCAP 구성 요소 추가	다음과 같은 SCAP 구성 요소를 도입했습니다. • scap-compliance-checker-linux 버전 2021.04.0 생성 (SCC 버전: 5.4.2) • scap-compliance-checker-linux 버전 2021.04.0 생성 (SCC 버전: 5.4.2)	2021년 12월 20일

Image Builder 이미지에 대한 사용자 지정 구성 요소 개발

고유한 구성 요소를 생성하여 정확한 사양에 따라 Image Builder 이미지를 사용자 지정할 수 있습니다. 다음 단계를 사용하여 Image Builder 이미지 또는 컨테이너 레시피에 대한 사용자 지정 구성 요소를 개발합니다.

1. 구성 요소 문서를 개발하고 로컬에서 검증하려면 AWS Task Orchestrator and Executor (AWSTOE) 애플리케이션을 설치하고 로컬 시스템에 설정할 수 있습니다. 자세한 내용은 를 사용하여 사용자 지정 구성 요소를 개발하도록 수동 설정 AWSTOE 단원을 참조하십시오.
2. 구성 요소 문서 프레임워크를 사용하는 AWSTOE 구성 요소 문서를 생성합니다. 문서 프레임워크에 대한 자세한 내용은 사용자 지정 AWSTOE 구성 요소에 구성 요소 문서 프레임워크 사용 섹션을 참조하세요.
3. 사용자 지정 구성 요소를 생성할 때 구성 요소 문서를 지정합니다. 자세한 내용은 Image Builder로 사용자 지정 구성 요소 생성 단원을 참조하십시오.

주제

- Image Builder에서 사용자 지정 구성 요소에 대한 YAML 구성 요소 문서 생성
- Image Builder로 사용자 지정 구성 요소 생성

Image Builder에서 사용자 지정 구성 요소에 대한 YAML 구성 요소 문서 생성

구성 요소를 빌드하려면, YAML 또는 JSON 애플리케이션 구성 요소 문서를 제공하십시오. 문서에는 이미지에 대한 사용자 지정을 제공하기 위해 정의한 단계에서 실행되는 코드가 포함되어 있습니다.

이 섹션의 일부 예제에서는 구성 요소 관리 애플리케이션에서 `UpdateOS` 작업 모듈을 호출하는 빌드 AWSTOE 구성 요소를 생성합니다. 이 모듈은 운영 체제를 업데이트합니다. `UpdateOS` 작업 모듈에 대한 자세한 내용은 UpdateOS 섹션을 참조하십시오.

macOS 운영 체제 예제에서는 `ExecuteBash` 작업 모듈을 사용하여 `wget` 유틸리티를 설치하고 확인합니다. `UpdateOS` 작업 모듈은 macOS를 지원하지 않습니다. `ExecuteBash` 작업 모듈에 대한 자세한 내용은 ExecuteBash 섹션을 참조하십시오. AWSTOE 애플리케이션 구성 요소 문서의 단계 및 구문에 대한 자세한 내용은 AWSTOE에서 문서 사용을 참조하십시오.

> **Note**
>
> Image Builder는 구성 요소 문서에 정의된 단계를 기반으로 다음과 같이 구성 요소 유형을 결정합니다.

- 빌드 - 기본 구성 요소 유형입니다. 테스트 구성 요소로 분류되지 않은 모든 것은 빌드 구성 요소입니다. 이 유형의 구성 요소는 이미지 빌드 단계에서 실행됩니다. 이 빌드 구성 요소에 `test` 단계가 정의되어 있는 경우 해당 단계는 테스트 단계에서 실행됩니다.
- 테스트 - 테스트 구성 요소로 적합하려면 구성 요소 문서에 `test`라는 이름의한 단계만 포함되어야 합니다. 빌드 구성 요소 구성과 관련된 테스트의 경우 독립형 테스트 구성 요소를 사용하지 않는 것이 좋습니다. 그보다는 연결된 빌드 구성 요소의 `test` 단계를 사용하십시오.

Image Builder에서 단계를 사용하여 빌드 프로세스에서 구성 요소 워크플로를 관리하는 방법에 대한 자세한 내용은 구성 요소를 사용하여 Image Builder 이미지 사용자 지정(을)를 참조하십시오.

샘플 애플리케이션에 대해 YAML 애플리케이션 구성 요소 문서를 만들려면 이미지 운영 체제에 맞는 탭의 단계를 따르십시오.

Linux

YAML 구성 요소 파일 생성

파일 편집 도구를 사용하여 구성 요소 문서를 생성합니다. 설명서 예제에서는 *update-linux-os.yaml*라는 파일을 다음 내용과 함께 사용합니다.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

```
name: update-linux-os
description: Updates Linux with the latest security updates.
schemaVersion: 1
phases:
  - name: build
    steps:
    - name: UpdateOS
      action: UpdateOS
# Document End
```

> **Tip**
>
> 이 온라인 YAML 검사기와 같은 도구나 코드 환경의 YAML Lint 확장을 사용하여 YAML이 제대로 구성되어 있는지 확인하십시오.

Windows

YAML 구성 요소 파일 생성

파일 편집 도구를 사용하여 구성 요소 문서를 생성합니다. 설명서 예제에서는 *update-windows-os.yaml*라는 파일을 다음 내용과 함께 사용합니다.

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
name: update-windows-os
```

```
description: Updates Windows with the latest security updates.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: UpdateOS
        action: UpdateOS
# Document End
```

> **Tip**
>
> 이 온라인 YAML 검사기와 같은 도구나 코드 환경의 YAML Lint 확장을 사용하여 YAML이 제대로 구성되어 있는지 확인하십시오.

macOS

YAML 구성 요소 파일 생성

파일 편집 도구를 사용하여 구성 요소 문서를 생성합니다. 설명서 예제에서는 *wget-macos.yaml*라는 파일을 다음 내용과 함께 사용합니다.

```
name: WgetInstallDocument
description: This is wget installation document.
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: WgetBuildStep
        action: ExecuteBash
        inputs:
          commands:
            - |
              PATH=/usr/local/bin:$PATH
              sudo -u ec2-user brew install wget


  - name: validate
    steps:
      - name: WgetValidateStep
        action: ExecuteBash
```

```
        inputs:
          commands:
            - |
              function error_exit {
                echo $1
                echo "{\"failureMessage\":\"$2\"}"
                exit 1
              }

              type wget
              if [ $? -ne 0 ]; then
                error_exit "$stderr" "Wget installation failed!"
              fi

  - name: test
    steps:
      - name: WgetTestStep
        action: ExecuteBash
        inputs:
          commands:
            - wget -h
```

> **Tip**
>
> 이 온라인 YAML 검사기와 같은 도구나 코드 환경의 YAML Lint 확장을 사용하여 YAML이 제대로 구성되어 있는지 확인하십시오.

Image Builder로 사용자 지정 구성 요소 생성

구성 요소 문서를 완료한 후 이를 사용하여 Image Builder 레시피가 사용할 수 있는 사용자 지정 구성 요소를 생성할 수 있습니다. Image Builder 콘솔, API 또는 SDK 또는 명령줄에서 사용자 지정 구성 요소를 생성할 수 있습니다. 입력 파라미터를 사용하여 사용자 지정 구성 요소를 생성하고 레시피에 사용하는 방법에 대한 자세한 내용은 자습서: 입력 파라미터를 사용하여 사용자 지정 구성 요소 만들기 섹션을 참조하세요.

다음 섹션에서는 콘솔 또는 AWS CLI에서 구성 요소를 생성하는 방법을 보여줍니다.

내용

- 콘솔에서 사용자 지정 구성 요소 생성
- 에서 사용자 지정 구성 요소 생성 AWS CLI

- 스크립트를 가져와에서 구성 요소를 생성합니다. AWS CLI

콘솔에서 사용자 지정 구성 요소 생성

Image Builder 콘솔에서 AWSTOE 애플리케이션 구성 요소를 생성하려면 다음 단계를 따릅니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 구성 요소를 선택합니다. 그런 다음 구성 요소 생성하기를 선택합니다.
3. 구성 요소 생성하기 페이지의 구성 요소 세부 정보 아래에서 다음을 입력합니다.

   a. 이미지 운영 체제(OS). 구성 요소와 호환되는 운영 체제를 지정합니다.
   b. 구성 요소 범주. 드롭다운에서 생성 중인 빌드 또는 테스트 구성 요소의 유형을 선택합니다.
   c. 구성 요소 이름. 구성 요소의 이름을 입력합니다.
   d. 구성 요소 버전. 구성 요소의 버전에 대한 번호를 입력합니다.
   e. 설명. 구성 요소를 식별할 수 있도록 선택적 설명을 제공합니다.
   f. 설명 변경하기. 이 버전의 구성 요소에 적용된 변경 사항을 이해하는 데 도움이 되는 선택적 설명을 제공합니다.
4. 정의 문서 섹션의 기본 옵션은 문서 콘텐츠 정의하기입니다. 구성 요소 문서는 Image Builder가 빌드 및 테스트 인스턴스에서 이미지를 생성하기 위해 수행하는 작업을 정의합니다.

   콘텐츠 상자에서 YAML 구성 요소 문서 콘텐츠를 입력합니다. Linux용 Hello World 예제로 시작하려면 예제 사용하기 옵션을 선택합니다. YAML 구성 요소 문서를 생성하거나 해당 페이지에서 UpdateOS 예제를 복사하여 붙여넣는 방법에 대해 자세히 알아보려면 Image Builder에서 사용자 지정 구성 요소에 대한 YAML 구성 요소 문서 생성(을)를 참조합니다.
5. 구성 요소 세부 정보를 입력한 후 구성 요소 생성하기를 선택합니다.

   > **Note**
   >
   > 레시피를 생성하거나 업데이트할 때 새 구성 요소를 보려면 내 소유 필터를 빌드 또는 테스트 구성 요소 목록에 적용합니다. 필터는 검색 상자 옆의 구성 요소 목록의 상단에 있습니다.
6. 구성 요소를 삭제하려면 구성 요소 페이지에서 삭제할 구성 요소 옆에 있는 확인란을 선택합니다. 작업 드롭다운에서 구성 요소 삭제하기를 선택합니다.

구성 요소 업데이트

새 구성 요소 버전을 생성하려면 다음 단계를 따릅니다.

1. 시작 위치에 따라:
   - 구성 요소 목록 페이지에서 구성 요소 이름 옆의 확인란을 선택한 다음 작업 메뉴에서 새 버전 생성하기를 선택합니다.
   - 구성 요소 세부 정보 페이지에서 제목의 오른쪽 상단에 있는 새 버전 생성하기 버튼을 선택합니다.
2. 구성 요소 생성하기 페이지가 표시되면 구성 요소 정보가 이미 현재 값으로 채워져 있습니다. 구성 요소 생성하기 단계에 따라 구성 요소를 업데이트합니다. 이렇게 하면 구성 요소 버전에 고유한 시맨틱 버전을 입력할 수 있습니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

에서 사용자 지정 구성 요소 생성 AWS CLI

이 섹션에서는 다음과 같이에서 Image Builder 명령을 설정하고 사용하여 AWSTOE 애플리케이션 구성 요소를 AWS CLI 생성하는 방법을 알아봅니다.

- YAML 구성 요소 문서를 명령줄에서 참조할 수 있는 S3 버킷에 업로드합니다.
- create-component 명령을 사용하여 AWSTOE 애플리케이션 구성 요소를 생성합니다.
- list-components 명령과 이름 필터를 사용하여 구성 요소 버전을 나열하여 이미 존재하는 버전을 확인합니다. 출력을 사용하여 업데이트에 사용할 다음 버전을 결정할 수 있습니다.

입력 YAML 문서에서 AWSTOE 애플리케이션 구성 요소를 생성하려면 이미지 운영 체제 플랫폼과 일치하는 단계를 따릅니다.

Linux

Amazon S3에 애플리케이션 구성 요소 문서 저장

S3 버킷을 AWSTOE 애플리케이션 구성 요소 소스 문서의 리포지토리로 사용할 수 있습니다. 구성 요소 문서를 저장하려면 다음 단계를 수행합니다.

- Amazon S3에 문서 업로드

  문서가 64KB 미만인 경우 이 단계를 건너뛸 수 있습니다. 크기가 64KB 이상인 문서는 Amazon S3에 저장해야 합니다.

```
aws s3 cp update-linux-os.yaml s3://amzn-s3-demo-destination-bucket/my-
path/update-linux-os.yaml
```

YAML 문서에서 구성 요소 생성

에서 사용하는 create-component 명령을 간소화하려면 명령에 전달하려는 모든 구성 요소 파라미터가 포함된 JSON 파일을 AWS CLI생성합니다. 이전에 만든 *update-linux-os.yaml* 문서의 위치를 포함하십시오. `uri` 키-값 페어에는 파일 참조가 포함됩니다.

> **Note**
>
> JSON 파일의 데이터 값에 대한 이름 지정 규칙은 Image Builder API 작업 요청 파라미터에 지정된 패턴을 따릅니다. API 명령 요청 파라미터를 검토하려면 EC2 Image Builder API 참조의 CreateComponent 명령을 참조하십시오.
> 데이터 값을 명령줄 파라미터로 제공하려면 AWS CLI 명령 참조에 지정된 파라미터 이름을 참조하십시오.

1. CLI 입력 JSON 파일 생성

   파일 편집 도구를 사용하여 *create-update-linux-os-component.json*(이)라는 이름의 파일을 생성합니다. 다음 콘텐츠를 포함합니다.

```
{
 "name": "update-linux-os",
 "semanticVersion": "1.1.2",
 "description": "An example component that updates the Linux operating system",
 "changeDescription": "Initial version.",
 "platform": "Linux",
 "uri": "s3://amzn-s3-demo-destination-bucket/my-path/update-linux-os.yaml",
 "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
b123-456b-7f89-0123456f789c",
 "tags": {
  "MyTagKey-purpose": "security-updates"
 }
}
```

2. 구성 요소 생성

   이전 단계에서 만든 JSON 파일의 파일 이름을 참조하여 다음 명령을 사용하여 구성 요소를 생성합니다.

   ```
   aws imagebuilder create-component --cli-input-json file://create-update-linux-os-component.json
   ```

   > **Note**
   >
   > - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
   > - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

Windows

Amazon S3에 애플리케이션 구성 요소 문서 저장

S3 버킷을 AWSTOE 애플리케이션 구성 요소 소스 문서의 리포지토리로 사용할 수 있습니다. 구성 요소 문서를 저장하려면 다음 단계를 수행합니다.

- Amazon S3에 문서 업로드

  문서가 64KB 미만인 경우 이 단계를 건너뛸 수 있습니다. 크기가 64KB 이상인 문서는 Amazon S3에 저장해야 합니다.

  ```
  aws s3 cp update-windows-os.yaml s3://amzn-s3-demo-destination-bucket/my-path/update-windows-os.yaml
  ```

YAML 문서에서 구성 요소 생성

에서 사용하는 create-component 명령을 간소화하려면 명령에 전달하려는 모든 구성 요소 파라미터가 포함된 JSON 파일을 AWS CLI생성합니다. 이전에 만든 *update-windows-os.yaml* 문서의 위치를 포함하십시오. `uri` 키-값 페어에는 파일 참조가 포함됩니다.

> **Note**
>
> JSON 파일의 데이터 값에 대한 이름 지정 규칙은 Image Builder API 작업 요청 파라미터에 지정된 패턴을 따릅니다. API 명령 요청 파라미터를 검토하려면 EC2 Image Builder API 참조의 CreateComponent 명령을 참조하십시오.
> 데이터 값을 명령줄 파라미터로 제공하려면 AWS CLI 명령 참조에 지정된 파라미터 이름을 참조하십시오.

1. CLI 입력 JSON 파일 생성

   파일 편집 도구를 사용하여 *create-update-windows-os-component.json*(이)라는 이름의 파일을 생성합니다. 다음 콘텐츠를 포함합니다.

```
{
 "name": "update-windows-os",
 "semanticVersion": "1.1.2",
 "description": "An example component that updates the Windows operating
 system.",
 "changeDescription": "Initial version.",
 "platform": "Windows",
 "uri": "s3://amzn-s3-demo-destination-bucket/my-path/update-windows-os.yaml",
 "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
b123-456b-7f89-0123456f789c",
 "tags": {
  "MyTagKey-purpose": "security-updates"
 }
}
```

   > **Note**
   >
   > - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
   > - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

2. 구성 요소 생성

   이전 단계에서 만든 JSON 파일의 파일 이름을 참조하여 다음 명령을 사용하여 구성 요소를 생성합니다.

   ```
   aws imagebuilder create-component --cli-input-json file://create-update-windows-os-component.json
   ```

   > **Note**
   >
   > - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
   > - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

macOS

Amazon S3에 애플리케이션 구성 요소 문서 저장

S3 버킷을 AWSTOE 애플리케이션 구성 요소 소스 문서의 리포지토리로 사용할 수 있습니다. 구성 요소 문서를 저장하려면 다음 단계를 수행합니다.

- Amazon S3에 문서 업로드

  문서가 64KB 미만인 경우 이 단계를 건너뛸 수 있습니다. 크기가 64KB 이상인 문서는 Amazon S3에 저장해야 합니다.

  ```
  aws s3 cp wget-macos.yaml s3://amzn-s3-demo-destination-bucket/my-path/wget-macos.yaml
  ```

YAML 문서에서 구성 요소 생성

에서 사용하는 create-component 명령을 간소화하려면 명령에 전달하려는 모든 구성 요소 파라미터가 포함된 JSON 파일을 AWS CLI생성합니다. 이전에 만든 *wget-macos.yaml* 문서의 위치를 포함하십시오. `uri` 키-값 페어에는 파일 참조가 포함됩니다.

> **Note**
>
> JSON 파일의 데이터 값에 대한 이름 지정 규칙은 Image Builder API 작업 요청 파라미터에 지정된 패턴을 따릅니다. API 명령 요청 파라미터를 검토하려면 EC2 Image Builder API 참조의 CreateComponent 명령을 참조하십시오.
> 데이터 값을 명령줄 파라미터로 제공하려면 AWS CLI 명령 참조에 지정된 파라미터 이름을 참조하십시오.

1. CLI 입력 JSON 파일 생성

   파일 편집 도구를 사용하여 *install-wget-macos-component.json*(이)라는 이름의 파일을 생성합니다. 다음 콘텐츠를 포함합니다.

```
{
 "name": "install install-wget-macos-component",
 "semanticVersion": "1.1.2",
 "description": "An example component that installs and verifies the wget
 utility on macOS.",
 "changeDescription": "Initial version.",
 "platform": "macOS",
 "uri": "s3://amzn-s3-demo-destination-bucket/my-path/wget-macos.yaml",
 "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
b123-456b-7f89-0123456f789c",
 "tags": {
  "MyTagKey-purpose": "install-software"
 }
}
```

2. 구성 요소 생성

   이전 단계에서 만든 JSON 파일의 파일 이름을 참조하여 다음 명령을 사용하여 구성 요소를 생성합니다.

```
aws imagebuilder create-component --cli-input-json file://install-wget-macos-
component.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.

- JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

AWSTOE 의 업데이트를 위한 구성 요소 버전 관리 AWS CLI

AWSTOE 구성 요소 이름과 버전은 구성 요소 접두사 뒤에 구성 요소의 Amazon 리소스 이름(ARN)에 포함됩니다. 구성 요소의 새 버전마다 고유한 ARN이 있습니다. 새 버전을 생성하는 단계는 새 구성 요소를 만드는 단계와 완전히 동일합니다. 단, 시맨틱 버전이 해당 구성 요소 이름에 고유해야 합니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

다음 논리적 버전을 할당하려면 먼저 변경하려는 구성 요소의 기존 버전 목록을 가져오십시오. list-components 명령을 AWS CLI와 함께 사용하고 이름을 필터링합니다.

이 예제에서는 이전 Linux 예제에서 만든 구성 요소의 이름을 기준으로 필터링합니다. 생성한 구성 요소를 나열하려면 create-component 명령에서 사용한 JSON 파일의 name 파라미터 값을 사용합니다.

```
aws imagebuilder list-components --filters name="name",values="update-linux-os"
{
    "requestId": "123a4567-b890-123c-45d6-ef789ab0cd1e",
    "componentVersionList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:1234560087789012:component/update-
linux-os/1.0.0",
            "name": "update-linux-os",
            "version": "1.0.0",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2020-09-24T16:58:24.444Z"
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:1234560087789012:component/update-
linux-os/1.0.1",
            "name": "update-linux-os",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
```

```
            "dateCreated": "2021-07-10T03:38:46.091Z"
        }
    ]
}
```

결과를 바탕으로 다음 버전을 결정할 수 있습니다.

스크립트를 가져와에서 구성 요소를 생성합니다. AWS CLI

일부 시나리오에서는 기존 스크립트로 시작하는 것이 더 쉬울 수 있습니다. 이 시나리오의 경우 다음 예제를 사용할 수 있습니다.

이 예제에서는 *import-component.json*(그림 참조)이라는 파일이 있다고 가정합니다. 파일은 *amzn-s3-demo-source-bucket*에 이미 업로드되어 있는 AdminConfig.ps1이라는 PowerShell 스크립트를 직접 참조한다는 점에 유의합니다. 현재 SHELL(은)는 구성 요소 format에 의해 지원됩니다.

```
{
"name": "MyImportedComponent",
"semanticVersion": "1.0.0",
"description": "An example of how to import a component",
"changeDescription": "First commit message.",
"format": "SHELL",
"platform": "Windows",
"type": "BUILD",
"uri": "s3://amzn-s3-demo-source-bucket/AdminConfig.ps1",
"kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/60763706-
b131-418b-8f85-3420912f020c"
}
```

가져온 스크립트에서 구성 요소를 생성하려면 다음 명령을 실행합니다.

```
aws imagebuilder import-component --cli-input-json file://import-component.json
```

> **Note**
>
> 예상치 못한 요금이 부과되지 않도록 하려면 이 가이드의 예제에서 만든 리소스와 파이프라인을 정리하세요. Image Builder에서 리소스 삭제에 대한 자세한 내용은 오래되거나 사용되지 않는 Image Builder 리소스 삭제(을)를 참조하세요.

Image Builder가 AWS Task Orchestrator and Executor 애플리케이션을 사용하여 구성 요소를 관리하는 방법

EC2 Image Builder는 AWS Task Orchestrator and Executor (AWSTOE) 애플리케이션을 사용하여 복잡한 워크플로를 오케스트레이션하고, 시스템 구성을 수정하고, 추가 개발 스크립트 또는 코드 없이 이미지를 테스트합니다. 이 애플리케이션은 선언적 문서 스키마를 구현하는 구성 요소를 관리하고 실행합니다.

AWSTOE 는 이미지를 생성할 때 Image Builder가 빌드 및 테스트 인스턴스에 설치하는 독립 실행형 애플리케이션입니다. EC2 인스턴스에 수동으로 설치하여 사용자 지정 구성 요소를 생성할 수도 있습니다. 추가 설정이 필요하지 않으며 온프레미스에서 실행할 수도 있습니다.

내용

- AWSTOE 다운로드
- 지원되는 리전
- AWSTOE 명령 참조
- 를 사용하여 사용자 지정 구성 요소를 개발하도록 수동 설정 AWSTOE
- 사용자 지정 AWSTOE 구성 요소에 구성 요소 문서 프레임워크 사용
- AWSTOE 구성 요소 관리자가 지원하는 작업 모듈
- AWSTOE run 명령에 대한 입력 구성

AWSTOE 다운로드

설치하려면 아키텍처 및 플랫폼의 다운로드 링크를 AWSTOE선택합니다. 서비스의 VPC 엔드포인트(예: Image Builder)에 연결하는 경우 AWSTOE 다운로드를 위해 S3 버킷에 대한 액세스를 포함하는 사용자 지정 엔드포인트 정책이 연결되어 있어야 합니다. 그렇지 않으면 빌드 및 테스트 인스턴스가 부트스트랩 스크립트(`bootstrap.sh`)를 다운로드하고 AWSTOE 애플리케이션을 설치할 수 없습니다. 자세한 내용은 Image Builder에 대한 VPC 엔드포인트 정책 생성하기을 참조하세요.

> **Important**
>
> AWS 는 TLS 버전 1.0 및 1.1에 대한 지원을 단계적으로 중단합니다. AWSTOE 다운로드를 위해 S3 버킷에 액세스하려면 클라이언트 소프트웨어가 TLS 버전 1.2 이상을 사용해야 합니다. 자세한 내용은 AWS 보안 블로그 게시물을 참조하세요.

아키텍처	플랫폼	다운로드 링크	예제
386	AL 2 및 2023 RHEL 7, 8 및 9 Ubuntu 16.04, 18.04, 20.04, 22.04 및 24.04 CentOS 7 및 8 SUSE 12 및 15	`https://awstoe- <region>.s3.com/latest/linux/386/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/386/awstoe
AMD64	AL 2 및 2023 RHEL 7, 8 및 9 Ubuntu 16.04, 18.04, 20.04, 22.04 및 24.04 CentOS 7 및 8 CentOS Stream 8 SUSE 12 및 15	`https://awstoe- <region>.s3.com/latest/linux/amd64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/awstoe
AMD64	macOS 10.14.x(Mojave), 10.15.x(Catalina), 11.x(Big Sur), 12.x(Monterey)	`https://awstoe- region.s3.com/latest/darwin/amd64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/darwin/amd64/awstoe
AMD64	Windows Server 2012 R2, 2016, 2019 및 2022	`https://awstoe- <region>.s3.com/latest/windows/amd64/awstoe.exe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/windows/amd64/awstoe.exe
ARM64	AL 2 및 2023 RHEL 7, 8 및 9	`https://awstoe- <region>.s3`	https://awstoe-us-east-1.s3.us-east

아키텍처	플랫폼	다운로드 링크	예제
	Ubuntu 16.04, 18.04, 20.04, 22.04 및 24.04 CentOS 7 및 8 CentOS Stream 8 SUSE 12 및 15	`s.com/latest/linux/arm64/awstoe`	-1.amazonaws.com/latest/linux/arm64/awstoe

지원되는 리전

AWSTOE 는 다음 리전에서 독립 실행형 애플리케이션으로 지원됩니다.

AWS 리전 이름	AWS 리전
미국 동부(오하이오)	us-east-2
미국 동부(버지니아 북부)	us-east-1
AWS GovCloud(미국 동부)	us-gov-east-1
AWS GovCloud(미국 서부)	us-gov-west-1
미국 서부(캘리포니아 북부)	us-west-1
미국 서부(오리건)	us-west-2
아프리카(케이프타운)	af-south-1
아시아 태평양(홍콩)	ap-east-1
아시아 태평양(오사카)	ap-northeast-3
아시아 태평양(서울)	ap-northeast-2
아시아 태평양(뭄바이)	ap-south-1
아시아 태평양(하이데라바드)	ap-south-2

AWS 리전 이름	AWS 리전
아시아 태평양(싱가포르)	ap-southeast-1
아시아 태평양(시드니)	ap-southeast-2
아시아 태평양(자카르타)	ap-southeast-3
아시아 태평양(도쿄)	ap-northeast-1
캐나다(중부)	ca-central-1
유럽(프랑크푸르트)	eu-central-1
유럽(취리히)	eu-central-2
유럽(스톡홀름)	eu-north-1
유럽(밀라노)	eu-south-1
유럽(스페인)	eu-south-2
유럽(아일랜드)	eu-west-1
유럽(런던)	eu-west-2
유럽(파리)	eu-west-3
이스라엘(텔아비브)	il-central-1
중동(UAE)	me-central-1
중동(바레인)	me-south-1
남아메리카(상파울루)	sa-east-1
중국(베이징)	cn-north-1
중국(닝샤)	cn-northwest-1

AWSTOE 명령 참조

AWSTOE 는 Amazon EC2 인스턴스에서 실행되는 명령줄 구성 요소 관리 애플리케이션입니다. Image Builder는 EC2 빌드 또는 테스트 인스턴스를 시작할 때 인스턴스 AWSTOE 에를 설치합니다. 그런 다음에서 AWSTOE 명령을 실행 AWS CLI 하여 이미지 또는 컨테이너 레시피에 지정된 구성 요소를 설치하거나 검증합니다.

> **Note**
>
> 일부 AWSTOE 작업 모듈을 Linux 서버에서 실행하려면 승격된 권한이 필요합니다. 상승된 권한을 사용하려면 명령 구문에 접두사로 sudo을(를) 붙이거나 아래 링크된 sudo su 명령을 실행하기 전에 로그인할 때 명령을 한 번 실행하세요. AWSTOE 작업 모듈에 대한 자세한 내용은 섹션을 참조하세요AWSTOE 구성 요소 관리자가 지원하는 작업 모듈.

run

run 명령을 사용하여 하나 이상의 구성 요소 문서에 대해 YAML 문서 스크립트를 실행합니다.

검증

validate 명령을 실행하여 하나 이상의 구성 요소 문서에 대해 YAML 문서 의미 체계를 검증합니다.

awstoe 실행 명령

이 명령은 `--documents` 매개 변수로 지정된 구성 파일 또는 `--config` 매개 변수로 지정된 구성 요소 문서 목록에 포함된 순서대로 YAML 구성 요소 문서 스크립트를 실행합니다.

> **Note**
>
> 다음 매개변수 중 하나를 정확히 지정해야 합니다.
> --config
> --documents

구문

```
awstoe run [--config <file path>] [--cw-ignore-failures <?>]
      [--cw-log-group <?>] [--cw-log-region us-west-2] [--cw-log-stream <?>]
```

```
    [--document-s3-bucket-owner <owner>] [--documents <file path,file path,...>]
    [--execution-id <?>] [--log-directory <file path>]
    [--log-s3-bucket-name <name>] [--log-s3-bucket-owner <owner>]
    [--log-s3-key-prefix <?>] [--parameters name1=value1,name2=value2...]
    [--phases <phase name>] [--state-directory <directory path>] [--version <?>]
    [--help] [--trace]
```

매개 변수 및 옵션

매개 변수

--config ***./config-example.json***

간략한 형식: -c *./config-example.json*

구성 파일(조건부) 이 매개 변수는 이 명령이 실행 중인 구성 요소의 구성 설정이 포함된 JSON 파일의 파일 위치를 포함합니다. 구성 파일에 run 명령 설정을 지정하는 경우, `--documents` 매개 변수를 지정하면 안 됩니다. 입력 구성에 대한 자세한 정보는 AWSTOE run 명령에 대한 입력 구성 섹션을 참조하세요.

유효한 위치에는 다음이 포함됩니다.

- 로컬 파일 경로(*./config-example.json*)
- S3 URI(`s3://`*bucket/key*)

--cw-ignore-failures

간략한 형식: 해당 사항 없음

CloudWatch Logs의 로깅 실패는 무시하세요.

--cw-log-group

간략한 형식: 해당 사항 없음

CloudWatch Logs의 `LogGroup` 이름입니다.

--cw-log-region

간략한 형식: 해당 사항 없음

CloudWatch Logs에 적용되는 AWS 리전입니다.

--cw-log-stream

간략한 형식: 해당 사항 없음

console.log 파일을 스트리밍할 AWSTOE 위치를 지정하는 CloudWatch Logs의 LogStream 이름입니다.

--document-s3-bucket-owner

간략한 형식: 해당 사항 없음

S3 URI 기반 문서에 대한 버킷 소유자의 계정 ID입니다.

--documents *./doc-1.yaml,./doc-n.yaml*

Short form: -d *./doc-1.yaml*,*./doc-n*

구성 요소 문서(조건부) 이 매개 변수에는 실행할 YAML 구성 요소 문서의 쉼표로 구분된 파일 위치 목록이 포함됩니다. --config 매개 변수를 사용하여 run 명령에 대한 YAML 문서를 지정하는 경우, --documents 매개 변수를 지정하지 않아야 합니다.

유효한 위치에는 다음이 포함됩니다.

- 로컬 파일 경로(*./component-doc-example.yaml*).
- S3 URI(s3://*bucket/key*).
- Image Builder 컴포넌트 빌드 버전 ARN(arn:aws:imagebuilder:us-west-*2:123456789012*:component/*my-example-component*/2021.12.02/1).

> **Note**
>
> 목록의 항목 사이에는 공백이 없고 쉼표만 있습니다.

--execution-id

간략한 형식: -i

현재 run 명령 실행에 적용되는 고유 ID입니다. 이 ID는 해당 파일을 고유하게 식별하고 현재 명령 실행에 연결하기 위해 출력 및 로그 파일 이름에 포함됩니다. 이 설정을 생략하면가 GUID를 AWSTOE 생성합니다.

--log-directory

간략한 형식: -l

가이 명령 실행의 모든 로그 파일을 AWSTOE 저장하는 대상 디렉터리입니다. 기본적으로 이 파일은 TOE_<DATETIME>_<EXECUTIONID> 디렉터리에 위치합니다. 로그 디렉터리를 지정하지 않으면는 현재 작업 디렉터리()를 AWSTOE 사용합니다..

--log-s3-bucket-name

간략한 형식: -b

구성 요소 로그가 Amazon S3에 저장되는 경우(권장)는 구성 요소 애플리케이션 로그를이 파라미터에 이름이 지정된 S3 버킷에 AWSTOE 업로드합니다.

--log-s3-bucket-owner

간략한 형식: 해당 사항 없음

구성 요소 로그가 Amazon S3에 저장되는 경우(권장),가 로그 파일을 AWSTOE 작성하는 버킷의 소유자 계정 ID입니다.

--log-s3-key-prefix

간략한 형식: -k

구성 요소 로그가 Amazon S3에 저장되어 있는 경우(권장), 이 접두사는 버킷 내 로그 위치의 S3 객체 키 접두사입니다.

--parameters ***name1***=***value1***,***name2***=***value2***...

간략한 형식: 해당 사항 없음

매개변수는 구성 요소 문서에 정의된 변경 가능한 변수로, 호출 애플케이션이 런타임에 제공할 수 있는 설정을 포함합니다.

--phases

간략한 형식: -p

YAML 구성 요소 문서에서 실행할 단계를 지정하는 쉼표로 구분된 목록입니다. 구성 요소 문서에 추가 단계가 포함된 경우, 해당 단계는 실행되지 않습니다.

--state-directory

간략한 형식: -s

상태 추적 파일이 저장되는 파일 경로입니다.

--version

간략한 형식: -v

구성 요소 애플케이션 버전을 지정합니다.

옵션

--help

간략한 형식: -h

구성 요소 관리 애플케이션 옵션 사용에 대한 도움말 설명서를 표시합니다.

--trace

간략한 형식: -t

콘솔에 대한 자세한 로깅을 활성화합니다.

awstoe 검증 명령

이 명령을 실행하면 `--documents` 매개 변수로 지정된 각 구성 요소 문서의 YAML 문서 구문을 검증합니다.

구문

```
awstoe validate [--document-s3-bucket-owner <owner>]
      --documents <file path,file path,...> [--help] [--trace]
```

매개 변수 및 옵션

매개 변수

--document-s3-bucket-owner

간략한 형식: 해당 사항 없음

제공된 S3 URI 기반 문서의 소스 계정 ID

--documents ***./doc-1.yaml,./doc-n.yaml***

Short form: -d *./doc-1.yaml*,*./doc-n*

구성 요소 문서(필수) 이 매개 변수에는 실행할 YAML 구성 요소 문서의 쉼표로 구분된 파일 위치 목록이 포함됩니다. 유효한 위치에는 다음이 포함됩니다.

- 로컬 파일 경로(*./component-doc-example.yaml*)
- S3 URI(s3://*bucket/key*)

- Image Builder 컴포넌트 빌드 버전 ARN(arn:aws:imagebuilder:us-west-*2:123456789012*:component/*my-example-component*/2021.12.02/1)

> **Note**
>
> 목록의 항목 사이에는 공백이 없고 쉼표만 있습니다.

옵션

--help

간략한 형식: -h

구성 요소 관리 애플케이션 옵션 사용에 대한 도움말 설명서를 표시합니다.

--trace

간략한 형식: -t

콘솔에 대한 자세한 로깅을 활성화합니다.

를 사용하여 사용자 지정 구성 요소를 개발하도록 수동 설정 AWSTOE

AWS Task Orchestrator and Executor (AWSTOE) 애플리케이션은 구성 요소 정의 프레임워크 내에서 명령을 생성, 검증 및 실행하는 독립 실행형 애플리케이션입니다. AWS 서비스는 AWSTOE 를 사용하여 워크플로를 오케스트레이션하고, 소프트웨어를 설치하고, 시스템 구성을 수정하고, 이미지 빌드를 테스트할 수 있습니다.

다음 단계에 따라 AWSTOE 애플리케이션을 수동으로 설치하고 이를 독립형 애플리케이션으로 사용하여 사용자 지정 구성 요소를 개발합니다. Image Builder 콘솔 또는 AWS CLI 명령을 사용하여 사용자 지정 구성 요소를 생성하는 경우 Image Builder가 이러한 단계를 자동으로 처리합니다. 자세한 내용은 Image Builder로 사용자 지정 구성 요소 생성 단원을 참조하십시오.

AWSTOE 설치 다운로드의 서명 확인

이 섹션에서는 AWSTOE Linux, macOS 및 Windows 기반 운영 체제에서에 대한 설치 다운로드의 유효성을 확인하기 위한 권장 프로세스를 설명합니다.

주제

- Linux 또는 macOS에서 AWSTOE 설치 다운로드용 서명 확인

- Windows에서 AWSTOE 설치 다운로드에 대한 서명 확인

Linux 또는 macOS에서 AWSTOE 설치 다운로드용 서명 확인

이 주제에서는 Linux 기반 또는 macOS 운영 체제에서에 대한 설치 다운로드의 유효성을 확인하기 위한 권장 프로세스에 AWSTOE 대해 설명합니다.

인터넷에서 애플리케이션을 다운로드할 때마다 소프트웨어 게시자의 자격 증명을 인증하는 것이 좋습니다. 또한 애플리케이션이 게시된 이후 변경되거나 손상되지 않았는지 확인하세요. 이를 통해 바이러스나 기타 악성 코드가 포함된 애플리케이션 버전을 설치하는 것을 방지할 수 있습니다.

이 단원의 절차를 수행한 후에 AWSTOE 애플리케이션의 소프트웨어가 변경되거나 손상된 것을 발견한 경우 설치 파일을 실행하지 마십시오. 대신에 문의하십시오. 지원 옵션에 대한 지원 자세한 내용은 섹션을 참조하세요지원.

AWSTOE Linux 기반 및 macOS 운영 체제용 파일은 안전한 디지털 서명을 위한 Pretty Good Privacy(OpenPGP) 표준의 오픈 소스 구현GnuPG인를 사용하여 서명됩니다. GnuPG (라고도 함GPG)는 디지털 서명을 통해 인증 및 무결성 검사를 제공합니다. Amazon EC2는 다운로드한 Amazon EC2 CLI 도구를 확인하는 데 사용할 수 있는 퍼블릭 키 및 서명을 게시합니다. PGP 및 GnuPG(GPG)에 대한 자세한 내용은 http://www.gnupg.org를 참조하세요.

첫 번째 단계는 소프트웨어 게시자와 신뢰를 구축하는 것입니다. 소프트웨어 게시자의 퍼블릭 키를 다운로드하고, 퍼블릭 키의 소유자가 정당한 소유자인지 확인한 다음, 퍼블릭 키를 키링에 추가합니다. 키링은 알려진 퍼블릭 키의 모음입니다. 퍼블릭 키의 신뢰성을 설정한 후, 이를 사용하여 애플리케이션의 서명을 확인할 수 있습니다.

주제

- GPG 도구 설치
- 퍼블릭 키 인증 및 가져오기
- 패키지의 서명 확인

GPG 도구 설치

Linux, Unix 또는 macOS 운영 체제를 사용하는 경우 일반적으로 GPG 도구가 이미 설치되어 있습니다. 시스템에 도구가 설치되어 있는지 테스트하려면 명령 프롬프트에 gpg(을)를 입력합니다. GPG 도구가 설치되어 있는 경우, GPG 명령 프롬프트가 표시됩니다. GPG 도구가 설치되어 있지 않은 경우, 명령을 찾을 수 없다는 오류 메시지가 표시됩니다. 리포지토리에서 GnuPG 패키지를 설치할 수 있습니다.

GPG 도구를 설치하려면 인스턴스와 일치하는 운영 체제를 선택합니다.

Debian-based Linux

터미널에서 다음 명령을 실행합니다.

```
apt-get install gnupg
```

Red Hat–based Linux

터미널에서 다음 명령을 실행합니다.

```
yum install gnupg
```

macOS

터미널에서 다음 명령을 실행합니다.

```
brew install gnupg
```

퍼블릭 키 인증 및 가져오기

프로세스의 다음 단계는 AWSTOE 퍼블릭 키를 인증하고 GPG 키링에 신뢰할 수 있는 키로 추가하는 것입니다.

AWSTOE 퍼블릭 키를 인증하고 가져오려면

1. 다음 중 하나를 수행하여 퍼블릭 GPG 빌드 키 사본을 가져옵니다.

   - 키를

     https://awstoe-**<region>**.s3.**<region>**.amazonaws.com/assets/awstoe.gpg에서 다운로드하세요. 예를 들어 https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/assets/awstoe.gpg입니다.
   - 다음 텍스트에서 키를 복사하여 `awstoe.gpg`라는 파일에 붙여 넣습니다. 다음의 모든 항목을 포함해야 합니다.

     ```
     -----BEGIN PGP PUBLIC KEY BLOCK-----
     Version: GnuPG v2
     ```

```
mQENBF8UqwsBCACdiRF2bkZYaFSDPFC+LIkWLwFvtUCRwAHtD8KIwTJ6LVn3fHAU
GhuK0ZH9mRrqRT2bq/xJjGsnF9VqTj2AJqndGJdDjz75YCZYM+ocZ+r5HSJaeW9i
S5dykHj7Txti2zHe0G5+W0v7v5bPi2sPHsN7XWQ7+G2AMEPTz8PjxY//I0DvMQns
Sle3l9hz6wCClz1l9LbBzTyHfSm5ucTXvNe88XX5Gmt37OCDM7vfli0Ctv8WFoLN
6jbxuA/sV71yIkPm9IYp3+GvaKeT870+sn8/JOOKE/U4sJV1ppbqmuUzDfhrZUaw
8eW8IN9A1FTIuWiZED/5L83UZuQs1S7s2PjlABEBAAG0GkFXU1RPRSA8YXdzdG9l
QGFtYXpvbi5jb20+iQE5BBMBCAAjBQJfFKsLAhsDBwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQ3r3BVvWuvFJGiwf9EVmrBR77+Qe/DUeXZJYoaFr7If/fVDZl
6V3TC6p0J0Veme7uXleRUTFOjzbh+7e5sDX19HrnPquzCnzfMiqbp4lSoeUuNdOf
FcpuTCQH+M+sIEIgPno4PLl0Uj2uE1o++mxmonBl/Krk+hly8hB2L/9n/vW3L7BN
OMb1Ll9PmgGPbWipcT8KRdz4SUex9TXGYzjlWb3jU3uXetdaQY1M3kVKE1siRsRN
YYDtpcjmwbhjpu4xm19aFqNoAHCDctEsXJA/mkU3erwIRocPyjAZE2dnlkL9ZkFZ
z9DQkcIarbCnybDM5lemBbdhXJ6hezJE/b17VA0t1fY04MoEkn6oJg==
=oyze
-----END PGP PUBLIC KEY BLOCK-----
```

2. awstoe.gpg를 저장한 디렉터리의 명령 프롬프트에서 다음 명령을 사용하여 AWSTOE 퍼블릭 키를 키링으로 가져옵니다.

```
gpg --import awstoe.gpg
```

이 명령은 다음과 같은 결과를 반환합니다.

```
gpg: key F5AEBC52: public key "AWSTOE <awstoe@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

다음 단계에서 필요하므로 키 값을 적어 둡니다. 이전 예제에서 키 값은 F5AEBC52입니다.

3. 키-값을 이전 단계의 값으로 대체하고 다음 명령을 실행하여 지문을 확인합니다.

```
gpg --fingerprint key-value
```

이 명령에서 다음과 비슷한 결과를 반환합니다.

```
pub   2048R/F5AEBC52 2020-07-19
      Key fingerprint = F6DD E01C 869F D639 15E5  5742 DEBD C156 F5AE BC52
uid       [ unknown] AWSTOE <awstoe@amazon.com>
```

또한 지문 문자열은 이전 예제에 표시된 F6DD E01C 869F D639 15E5 5742 DEBD C156 F5AE BC52(와)과 동일해야 합니다. 반환된 키 지문을 이 페이지에 게시된 지문과 비교합니다. 두

지문이 일치해야 합니다. 일치하지 않는 경우 AWSTOE 설치 스크립트를 설치하지 말고에 문의하세요 지원.

패키지의 서명 확인

GPG 도구를 설치하고, AWSTOE 퍼블릭 키를 인증 및 가져오고, 퍼블릭 키가 신뢰할 수 있는지 확인하면 설치 스크립트의 서명을 확인할 준비가 된 것입니다.

설치 스크립트 서명을 확인하려면

1. 명령 프롬프트에서 다음 명령을 실행하여 애플리케이션 바이너리를 다운로드합니다.

   ```
   curl -O https://awstoe-<region>.s3.<region>.amazonaws.com/latest/
   linux/<architecture>/awstoe
   ```

   예시:

   ```
   curl -O https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/
   awstoe
   ```

   architecture의 지원되는 값은 amd64, 386, arm64일 수 있습니다.

2. 명령 프롬프트에서 다음 명령을 실행하여 동일한 S3 키 접두사 경로에서 해당 애플리케이션 바이너리용 서명 파일을 다운로드합니다.

   ```
   curl -O https://awstoe-<region>.s3.<region>.amazonaws.com/latest/
   linux/<architecture>/awstoe.sig
   ```

   예시:

   ```
   curl -O https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/
   awstoe.sig
   ```

   architecture의 지원되는 값은 amd64, 386, arm64일 수 있습니다.

3. 저장한 디렉터리의 명령 프롬프트awstoe.sig와 AWSTOE 설치 파일에서 다음 명령을 실행하여 서명을 확인합니다. 두 파일이 모두 있어야 합니다.

   ```
   gpg --verify ./awstoe.sig ~/awstoe
   ```

출력은 다음과 같아야 합니다.

```
gpg: Signature made Mon 20 Jul 2020 08:54:55 AM IST using RSA key ID F5AEBC52
gpg: Good signature from "AWSTOE awstoe@amazon.com" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F6DD E01C 869F D639 15E5 5742 DEBD C156 F5AE BC52
```

출력에 `Good signature from "AWSTOE <awstoe@amazon.com>"` 문구가 포함된 경우 서명을 확인했고, AWSTOE 설치 스크립트 실행을 계속할 수 있음을 의미합니다.

출력에 `BAD signature` 문구가 포함된 경우, 절차를 올바르게 수행했는지 확인합니다. 계속해서 이 응답을 받게 되면, 이전에 다운로드한 설치 파일을 실행하지 마시고 지원에 문의하세요.

다음은 표시될 수 있는 경고에 대한 세부 정보입니다.

- 경고: 이 키는 신뢰할 수 있는 서명으로 인증되지 않았습니다. 서명이 소유자에게 속한다는 표시가 없습니다. AWS 사무실을 방문하여 직접 키를 받는 것이 가장 좋습니다. 그러나 대부분의 경우 웹 사이트에서 다운로드합니다. 이 경우 웹 사이트는 AWS 웹 사이트입니다.
- gpg: 궁극적으로 신뢰할 수 있는 키를 찾을 수 없습니다. 이는 사용자 또는 사용자가 신뢰하는 다른 사용자가 특정 키를 '궁극적으로 신뢰'하지 않음을 뜻합니다.

자세한 내용은 http://www.gnupg.org를 참조하세요.

Windows에서 AWSTOE 설치 다운로드에 대한 서명 확인

이 주제에서는 Windows 기반 운영 체제에서 AWS Task Orchestrator and Executor 애플리케이션에 대한 설치 파일의 유효성을 확인하기 위한 권장 프로세스에 대해 설명합니다.

인터넷에서 애플리케이션을 다운로드할 때마다 소프트웨어 게시자의 자격 증명을 인증하고 애플리케이션이 게시된 이후 변경되거나 손상되지 않았는지 확인하는 것이 좋습니다. 이를 통해 바이러스나 기타 악성 코드가 포함된 애플리케이션 버전을 설치하는 것을 방지할 수 있습니다.

이 단원의 절차를 수행한 후에 AWSTOE 애플리케이션의 소프트웨어가 변경되거나 손상된 것을 발견한 경우 설치 파일을 실행하지 마십시오. 대신에 문의하세요 지원.

Windows 기반 운영 체제에서 다운로드된 awstoe 바이너리의 유효성을 확인하려면 Amazon Services LLC 서명자 인증서의 지문이 다음 값과 동일한지 확인해야 합니다.

BA 81 25 EE AC 64 2E A9 F3 C5 93 CA 6D 3E B7 93 7D 68 75 74

> **Note**
>
> 새 바이너리의 롤아웃 기간 동안 서명자 인증서가 새 지문과 일치하지 않을 수 있습니다. 서명자 인증서가 일치하지 않는 경우 지문 값이 다음과 같은지 확인하세요.
> F8 83 11 EE F0 4A A2 91 E3 79 21 BA 6B FC AF F8 19 92 12 D7

이 값을 확인하려면 다음 절차를 수행합니다.

1. 다운로드한 `awstoe.exe`(을)를 마우스 오른쪽 버튼으로 클릭하고 속성(Properties) 창을 엽니다.
2. 디지털 서명(Digital Signatures) 탭을 선택합니다.
3. 서명 목록(Signature List)에서 Amazon Services LLC를 선택한 후 세부 정보(Details)를 선택합니다.
4. 일반 탭이 선택되어 있지 않으면 이 탭을 선택한 후 인증서 보기(View Certificate)를 선택합니다.
5. 세부 정보 탭을 선택한 다음, 선택되어 있지 않은 경우 표시 드롭다운 목록에서 모두(All)를 선택합니다.
6. 지문(Thumbprint) 필드가 보일 때까지 아래로 스크롤한 후 지문(Thumbprint)을 선택합니다. 그러면 아래 창에 전체 지문 값이 표시됩니다.

   - 아래 창의 지문 값이 다음과 값과 동일한지 확인합니다.

     BA 81 25 EE AC 64 2E A9 F3 C5 93 CA 6D 3E B7 93 7D 68 75 74

     그러면 다운로드한 AWSTOE 바이너리가 정품이며 안전하게 설치할 수 있습니다.
   - 아래 세부 정보 창의 지문 값이 이전 값과 동일하지 않을 경우 `awstoe.exe`(을)를 실행하지 마십시오.

시작 절차

- 1단계: 설치 AWSTOE
- 2단계: AWS 자격 증명 설정
- 3단계: 구성 요소 문서를 로컬로 개발
- 4단계: AWSTOE 구성 요소 검증
- 5단계: AWSTOE 구성 요소 실행

1단계: 설치 AWSTOE

로컬에서 구성 요소를 개발하려면 AWSTOE 애플리케이션을 다운로드하여 설치합니다.

1. AWSTOE 애플리케이션 다운로드

   설치하려면 아키텍처 및 플랫폼에 적합한 다운로드 링크를 AWSTOE선택합니다. 애플리케이션 다운로드 링크의 전체 목록은 AWSTOE 다운로드 섹션을 참조하세요.

   > **Important**
   >
   > AWS 는 TLS 버전 1.0 및 1.1에 대한 지원을 단계적으로 중단합니다. AWSTOE 다운로드를 위해 S3 버킷에 액세스하려면 클라이언트 소프트웨어가 TLS 버전 1.2 이상을 사용해야 합니다. 자세한 내용은 AWS 보안 블로그 게시물을 참조하세요.

2. 서명 확인

   다운로드를 확인하는 단계는 AWSTOE 애플리케이션을 설치한 후 실행하는 서버 플랫폼에 따라 다릅니다. Linux 서버에서 다운로드를 확인하려면 Linux 또는 macOS에서 서명 확인 섹션을 참조하세요. Windows 서버에서 다운로드를 확인하려면 윈도우에서 서명 확인 섹션을 참조하세요.

> **Note**
>
> AWSTOE 는 다운로드 위치에서 직접 호출됩니다. 별도의 설치 단계는 필요 없습니다. 또한는 로컬 환경을 변경할 AWSTOE 수 있습니다.
> 구성 요소 개발 중에 변경 사항을 격리하려면 EC2 인스턴스를 사용하여 AWSTOE 구성 요소를 개발하고 테스트하는 것이 좋습니다.

2단계: AWS 자격 증명 설정

AWSTOE 는 다음과 AWS 서비스같은 작업을 실행할 때 Amazon S3 및 Amazon CloudWatch와 같은 다른에 연결하기 위한 AWS 자격 증명이 필요합니다.

- 사용자가 제공한 Amazon S3 경로에서 AWSTOE 문서 다운로드.
- `S3Download` 또는 `S3Upload` 작업 모듈을 실행합니다.
- 활성화된 경우 CloudWatch로 로그를 스트리밍합니다.

EC2 인스턴스 AWSTOE 에서를 실행하는 경우 실행은 EC2 인스턴스에 연결된 IAM 역할과 동일한 권한을 AWSTOE 사용합니다.

EC2용 IAM 역할에 대한 자세한 내용은 Amazon EC2의 IAM 역할을 참조하세요.

다음 예제에서는 AWS_ACCESS_KEY_ID 및 AWS_SECRET_ACCESS_KEY 환경 변수를 사용하여 AWS 자격 증명을 설정하는 방법을 보여줍니다.

Linux, macOS 또는 Unix에서 이러한 변수를 설정하려면 export를 사용합니다.

```
export AWS_ACCESS_KEY_ID=your_access_key_id
```

```
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

PowerShell을 사용하여 Windows에서 이러한 변수를 설정하려면 $env(을)를 사용합니다.

```
$env:AWS_ACCESS_KEY_ID=your_access_key_id
```

```
$env:AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

명령 프롬프트를 사용하여 Windows에서 이러한 변수를 설정하려면 set(을)를 사용합니다.

```
set AWS_ACCESS_KEY_ID=your_access_key_id
```

```
set AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

3단계: 구성 요소 문서를 로컬로 개발

구성 요소는 일반 텍스트 YAML 문서를 사용하여 작성됩니다. 문서 구문에 대한 자세한 내용은 사용자 지정 AWSTOE 구성 요소에 구성 요소 문서 프레임워크 사용 섹션을 참조하세요.

다음은 시작하는 데 도움이 되는 Hello World 구성 요소 문서의 예입니다.

Linux

이 안내서의 Linux 구성 요소 예제 중 일부는 hello-world-linux.yml이라는 구성 요소 문서 파일을 참조합니다. 다음 문서를 사용하여 이러한 예제를 시작할 수 있습니다.

```
name: Hello World
```

```
description: This is hello world testing document for Linux.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the test phase.'
```

Windows

이 안내서의 Windows 구성 요소 예제 중 일부는 `hello-world-windows.yml`이라는 구성 요소 문서 파일을 참조합니다. 다음 문서를 사용하여 이러한 예제를 시작할 수 있습니다.

```
name: Hello World
description: This is Hello World testing document for Windows.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the build phase.'
  - name: validate
    steps:
```

```
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the test phase.'
```

macOS

이 안내서의 일부 macOS 구성 요소 예제는 `hello-world-macos.yml`이라는 구성 요소 문서 파일을 참조합니다. 다음 문서를 사용하여 이러한 예제를 시작할 수 있습니다.

```
name: Hello World
description: This is hello world testing document for macOS.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the test phase.'
```

4단계: AWSTOE 구성 요소 검증

애플리케이션을 사용하여 로컬에서 AWSTOE 구성 요소의 구문을 AWSTOE 검증할 수 있습니다. 다음 예제에서는 구성 요소를 실행하지 않고 구문을 검증하는 AWSTOE 애플리케이션 `validate` 명령을 보여줍니다.

> **Note**
>
> AWSTOE 애플리케이션은 현재 운영 체제의 구성 요소 구문만 검증할 수 있습니다. 예를 들어 Windows에서 `awstoe.exe`(을)를 실행하는 경우, `ExecuteBash` 작업 모듈을 사용하는 Linux 문서의 구문을 검증할 수 없습니다.

Linux 또는 macOS

```
awstoe validate --documents /home/user/hello-world.yml
```

Windows

```
awstoe.exe validate --documents C:\Users\user\Documents\hello-world.yml
```

5단계: AWSTOE 구성 요소 실행

AWSTOE 애플리케이션은 `--phases` 명령줄 인수를 사용하여 지정된 문서의 단계를 하나 이상 실행할 수 있습니다. `--phases`의 지원되는 값은 `build`, `validate` 및 `test`입니다. 여러 단계 값을 쉼표로 구분된 값으로 입력할 수 있습니다.

단계 목록을 제공하면 AWSTOE 애플리케이션이 각 문서의 지정된 단계를 순차적으로 실행합니다. 예를 들어,는의 `build` 및 `validate` 단계를 `document1.yaml`실행한 다음의 `build` 및 `validate` 단계를 AWSTOE 실행합니다`document2.yaml`.

로그를 안전하게 저장하고 문제 해결을 위해 보관하려면 Amazon S3에 로그 스토리지를 구성하는 것이 좋습니다. Image Builder에서 로그를 게시하기 위한 Amazon S3 위치는 인프라 구성에 지정됩니다. 이 인프라 구성에 대한 자세한 내용은 Image Builder 인프라 구성 관리 섹션을 참조하세요.

단계 목록이 제공되지 않은 경우 AWSTOE 애플리케이션은 YAML 문서에 나열된 순서대로 모든 단계를 실행합니다.

단일 또는 여러 문서에서 특정 단계를 실행하려면, 다음 명령을 사용합니다.

단일 단계

```
awstoe run --documents hello-world.yml --phases build
```

여러 단계

```
awstoe run --documents hello-world.yml --phases build,test
```

문서 실행

단일 문서에서 모든 단계를 실행합니다.

```
awstoe run --documents documentName.yaml
```

여러 문서의 모든 단계 실행

```
awstoe run --documents documentName1.yaml,documentName2.yaml
```

Amazon S3 정보를 입력하여 사용자 정의 로컬 경로에서 AWSTOE 로그 업로드(권장)

```
awstoe run --documents documentName.yaml --log-s3-bucket-name amzn-s3-demo-destination-
bucket --log-s3-key-prefix S3KeyPrefix --log-s3-bucket-owner S3BucketOwner --log-
directory local_path
```

단일 문서에서 모든 단계를 실행하고, 콘솔에 모든 로그를 표시합니다.

```
awstoe run --documents documentName.yaml --trace
```

명령 예

```
awstoe run --documents s3://bucket/key/doc.yaml --phases build,validate
```

고유 ID로 문서 실행

```
awstoe run --documents documentName.yaml --execution-id user-provided-id --
phases build,test
```

에 대한 도움말 보기 AWSTOE

```
awstoe --help
```

사용자 지정 AWSTOE 구성 요소에 구성 요소 문서 프레임워크 사용

AWS Task Orchestrator and Executor (AWSTOE) 구성 요소 프레임워크를 사용하여 구성 요소를 빌드하려면 생성한 구성 요소에 적용되는 단계와 단계를 나타내는 YAML 기반 문서를 제공해야 합니다. 새 Amazon Machine Image(AMI) 또는 컨테이너 이미지를 생성할 때 구성 요소를 AWS 서비스 사용합니다.

주제

- 구성 요소 문서 워크플로우
- 구성 요소 로깅
- 입력 및 출력 체인화
- 문서 스키마 및 정의
- 문서 예제
- 사용자 지정 구성 요소 문서에서 변수 사용
- AWSTOE에서 조건부 구문 사용
- AWSTOE 구성 요소 문서에서 비교 연산자 사용
- AWSTOE 구성 요소 문서에서 논리 연산자 사용
- AWSTOE에서 루프 구문 사용

구성 요소 문서 워크플로우

AWSTOE 구성 요소 문서는 단계와 단계를 사용하여 관련 작업을 그룹화하고 해당 작업을 구성 요소의 논리적 워크플로로 구성합니다.

> **Tip**
>
> 구성 요소를 사용하여 이미지를 빌드하는 서비스는 빌드 프로세스에 사용할 단계와 해당 단계의 실행 허용 시기에 대한 규칙을 구현할 수 있습니다. 이는 구성 요소를 설계할 때 고려해야 할 중요한 사항입니다.

단계

단계는 이미지 빌드 프로세스를 통한 워크플로우의 진행 상황을 나타냅니다. 예를 들어 Image Builder 서비스는 생성한 이미지에 대해 빌드 단계에서 `build` 및 `validate` 단계를 사용합니다. 테스트 단계에서 `test` 및 `container-host-test` 단계를 사용하여 최종 AMI를 생성하거나 컨테이너 이미지를 배포하기 전에 이미지 스냅샷 또는 컨테이너 이미지가 예상 결과를 생성하는지 확인합니다.

구성 요소가 실행되면 각 단계의 관련 명령이 구성 요소 문서에 표시된 순서대로 적용됩니다.

단계 규칙

- 각 단계 이름은 문서 내에서 고유해야 합니다.
- 문서에 여러 단계를 정의할 수 있습니다.
- 문서에 다음 단계 중 하나 이상을 포함해야 합니다.
  - 빌드 — Image Builder의 경우 이 단계는 일반적으로 빌드 단계에서 사용됩니다.
  - 검증 — Image Builder의 경우 이 단계는 일반적으로 빌드 단계에서 사용됩니다.
  - 테스트 — Image Builder의 경우 이 단계는 일반적으로 테스트 단계에서 사용됩니다.
- 단계는 항상 문서에 정의된 순서대로 실행됩니다. 의 AWSTOE 명령 AWS CLI 에 지정된 순서는 영향을 미치지 않습니다.

단계(Steps)

단계는 각 단계 내의 워크플로우를 정의하는 개별 작업 단위입니다. 단계는 순차적으로 실행됩니다. 하지만 한 단계의 입력 또는 출력이 다음 단계에 입력으로 제공될 수도 있습니다. 이를 '체인화'라고 합니다.

단계 규칙

- 단계 이름은 해당 단계에 대해 고유해야 합니다.
- 단계는 종료 코드를 반환하는 지원되는 작업(작업 모듈)을 사용해야 합니다.

  지원되는 작업 모듈의 전체 목록, 작동 방식, 입력/출력 값 및 예제는 AWSTOE 구성 요소 관리자가 지원하는 작업 모듈(을)를 참조하세요.

구성 요소 로깅

AWSTOE 는 구성 요소가 실행될 때마다 새 이미지를 빌드하고 테스트하는 데 사용되는 새 로그 폴더를 EC2 인스턴스에 생성합니다. 컨테이너 이미지의 경우 로그 폴더가 컨테이너에 저장됩니다.

이미지 생성 프로세스 중에 문제가 발생하는 경우 문제를 해결하는 데 도움이 되도록 구성 요소를 실행하는 동안 AWSTOE 생성되는 모든 출력 파일과 입력 문서가 로그 폴더에 저장됩니다.

로그 폴더 이름은 다음과 같은 부분으로 구성됩니다.

1. 로그 디렉터리 - 서비스가 AWSTOE 구성 요소를 실행하면 명령에 대한 다른 설정과 함께 로그 디렉터리를 전달합니다. 다음 예제에서는 Image Builder에서 사용하는 로그 파일 형식을 보여 줍니다.
   - Linux 및 macOS: `/var/lib/amazon/toe/`
   - Windows: `$env:ProgramFiles\Amazon\TaskOrchestratorAndExecutor\`
2. 파일 접두사 - 모든 구성 요소에 사용되는 표준 접두사입니다: 'TOE_'.
3. 런타임 — YYYY-MM-DD_HH-MM-SS_UTC-0 형식의 타임스탬프입니다.
4. 실행 ID -가 하나 이상의 구성 요소를 AWSTOE 실행할 때 할당되는 GUID입니다.

예시: *`/var/lib/amazon/toe/`*`TOE`*`_2021-07-01_12-34-56_UTC-0_a1bcd2e3-45f6-789a-bcde-0fa1b2c3def4`*

AWSTOE 는 로그 폴더에 다음 코어 파일을 저장합니다.

입력 파일

- document.yaml — 명령의 입력으로 사용되는 문서. 구성 요소가 실행된 후 이 파일은 아티팩트로 저장됩니다.

출력 파일

- application.log - 애플리케이션 로그에는 구성 요소가 실행될 때 발생하는 상황에 대한 AWSTOE 의 타임스탬프가 있는 디버그 수준 정보가 들어 있습니다.
- detailedoutput.json — 이 JSON 파일에는 구성 요소 실행 시 적용되는 모든 문서, 단계에 대한 실행 상태, 입력, 출력 및 실패에 대한 자세한 정보가 들어 있습니다.
- console.log - 콘솔 로그에는 구성 요소가 실행되는 동안 콘솔에 AWSTOE 쓰는 모든 표준 출력(stdout) 및 표준 오류(stderr) 정보가 포함됩니다.
- chaining.json –이 JSON 파일은 체인 표현식을 해석하는 데 AWSTOE 적용된 최적화를 나타냅니다.

> **Note**
>
> 로그 폴더에는 여기에서 다루지 않은 다른 임시 파일도 포함될 수 있습니다.

입력 및 출력 체인화

AWSTOE 구성 관리 애플리케이션은 다음 형식으로 참조를 작성하여 입력과 출력을 연결하는 기능을 제공합니다.

```
{{ phase_name.step_name.inputs/outputs.variable }}
```

or

```
{{ phase_name.step_name.inputs/outputs[index].variable }}
```

체인화 기능을 사용하면 코드를 재활용하고 문서의 유지 관리 용이성을 개선할 수 있습니다.

체인화 규칙

- 체인화 표현식은 각 단계의 입력 섹션에서만 사용할 수 있습니다.
- 체인화 표현식이 포함된 명령문은 따옴표로 묶어야 합니다. 예시:
  - 잘못된 표현식: `echo {{ phase.step.inputs.variable }}`
  - 유효한 표현식: `"echo {{ phase.step.inputs.variable }}"`
  - 유효한 표현식: `'echo {{ phase.step.inputs.variable }}'`
- 체인화 표현식은 동일한 문서 내 다른 단계의 변수를 참조할 수 있습니다. 하지만 호출 서비스에는 체인화 표현식이 단일 단계 컨텍스트 내에서만 작동하도록 요구하는 규칙이 있을 수 있습니다. 예를 들어 Image Builder는 각 단계를 독립적으로 실행하므로 빌드 단계에서 테스트 단계로의 체인화를 지원하지 않습니다.
- 체인화 표현식의 인덱스는 0부터 시작하는 인덱싱을 따릅니다. 첫 번째 요소를 참조하는 인덱스는 0으로 시작합니다.

예시

다음 예제 단계의 두 번째 항목에서 소스 변수를 참조하기 위한 체인화 패턴은 다음과 같습니다.`{{ build.SampleS3Download.inputs[1].source }}`

```
phases:
  - name: 'build'
    steps:
      - name: SampleS3Download
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
```

```
        inputs:
          - source: 's3://sample-bucket/sample1.ps1'
            destination: 'C:\sample1.ps1'
          - source: 's3://sample-bucket/sample2.ps1'
            destination: 'C:\sample2.ps1'
```

다음 예제 단계의 출력 변수('Hello'와 동등)를 참조하기 위한 체인화 패턴은 다음과 같습니다.{{ build.*SamplePowerShellStep*.outputs.stdout }}

```
phases:
  - name: 'build'
    steps:
      - name: SamplePowerShellStep
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          commands:
            - 'Write-Host "Hello"'
```

문서 스키마 및 정의

다음은 문서의 YAML 스키마입니다.

```
name: (optional)
description: (optional)
schemaVersion: "string"

phases:
  - name: "string"
    steps:
      - name: "string"
        action: "string"
        timeoutSeconds: integer
        onFailure: "Abort|Continue|Ignore"
        maxAttempts: integer
        inputs:
```

문서의 스키마 정의는 다음과 같습니다.

필드	설명	형식	필수
name	문서의 이름.	String	No
설명	문서에 대한 설명.	String	No
schemaVersion	문서의 스키마 버전, 현재 1.0입니다.	String	예
단계	단계를 포함한 단계 목록.	나열	예

단계의 스키마 정의는 다음과 같습니다.

필드	설명	형식	필수
이름	단계 이름.	String	예
단계(steps)	단계의 단계 목록.	나열	예

단계의 스키마 정의는 다음과 같습니다.

필드	설명	형식	필수	기본값
name	단계에 대한 사용자 정의 이름.	String		
작업	단계를 실행하는 모듈과 관련된 키워드.	String		
timeoutSeconds	실패하거나 재시도하기 전에 단계가 실행되는 시간(초)	Integer	아니요	7,200초(120분)

필드	설명	형식	필수	기본값
	또한 무한 타임아웃을 나타내는 -1 값을 지원합니다. 0 및 기타 음수 값은 허용되지 않습니다.			

필드	설명	형식	필수	기본값
onFailure	장애 발생 시 단계에서 취해야 할 조치를 지정합니다. 유효한 값은 다음과 같습니다. • 중단 - 최대 시도 횟수가 지나면 단계가 실패하고 실행이 중지됩니다. 단계 및 문서의 상태를 Failed(으)로 설정합니다. • 계속 - 최대 시도 횟수가 지나면 단계가 실패하고 나머지 단계를 계속 실행합니다. 단계 및 문서의 상태를 Failed(으)로 설정합니다. • 무시 - 최대 시도 실패 횟수가 지난 후 단계를 IgnoredFailure 로 설정하고 나머지 단계를 계속 실행합니다. 단계 및 문서의 상태	String	No	중단

필드	설명	형식	필수	기본값
	를 SuccessWithIgnoredFailure (으)로 설정합니다.			
maxAttempts	단계가 실패하기 전까지 허용되는 최대 시도 횟수.	Integer	아니요	1
입력	작업 모듈이 단계를 실행하는 데 필요한 파라미터를 포함.	Dict	예	

문서 예제

다음 예제는 대상 운영 체제에 대한 작업을 수행하는 AWSTOE 구성 요소 문서를 보여줍니다.

Linux

예제 1: 사용자 지정 바이너리 파일 실행

다음은 Linux 인스턴스에서 사용자 지정 바이너리 파일을 다운로드하고 실행하는 예제 문서입니다.

```
name: LinuxBin
description: Download and run a custom Linux binary file.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://<replaceable>amzn-s3-demo-source-bucket</replaceable>/
<replaceable>myapplication</replaceable>
            destination: /tmp/<replaceable>myapplication</replaceable>
      - name: Enable
        action: ExecuteBash
        onFailure: Continue
```

```
        inputs:
          commands:
            - 'chmod u+x {{ build.Download.inputs[0].destination }}'
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: '{{ build.Download.inputs[0].destination }}'
          arguments:
            - '--install'
      - name: Delete
        action: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

Windows

예제 1: Windows 업데이트 설치

다음은 사용 가능한 모든 Windows 업데이트를 설치하고, 구성 스크립트를 실행하고, AMI를 생성하기 전에 변경 사항을 검증하고, AMI가 생성된 후 변경 사항을 테스트하는 예제 문서입니다.

```
name: RunConfig_UpdateWindows
description: 'This document will install all available Windows updates and run a
 config script. It will then validate the changes before an AMI is created. Then
 after AMI creation, it will test all the changes.'
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: DownloadConfigScript
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - source: 's3://customer-bucket/config.ps1'
            destination: 'C:\config.ps1'

      - name: RunConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
```

```
        inputs:
          file: '{{build.DownloadConfigScript.inputs[0].destination}}'

      - name: Cleanup
        action: DeleteFile
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - path: '{{build.DownloadConfigScript.inputs[0].destination}}'

      - name: RebootAfterConfigApplied
        action: Reboot
        inputs:
          delaySeconds: 60

      - name: InstallWindowsUpdates
        action: UpdateOS

  - name: validate
    steps:
      - name: DownloadTestConfigScript
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - source: 's3://customer-bucket/testConfig.ps1'
            destination: 'C:\testConfig.ps1'

      - name: ValidateConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          file: '{{validate.DownloadTestConfigScript.inputs[0].destination}}'

      - name: Cleanup
        action: DeleteFile
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - path: '{{validate.DownloadTestConfigScript.inputs[0].destination}}'
```

```
  - name: test
    steps:
      - name: DownloadTestConfigScript
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - source: 's3://customer-bucket/testConfig.ps1'
            destination: 'C:\testConfig.ps1'

      - name: ValidateConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          file: '{{test.DownloadTestConfigScript.inputs[0].destination}}'
```

예제 2: Windows 인스턴스 AWS CLI 에 설치

다음은 설정 파일을 사용하여 Windows 인스턴스 AWS CLI 에를 설치하는 예제 문서입니다.

```
name: InstallCLISetUp
description: Install &CLI; using the setup file
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://aws-cli/AWSCLISetup.exe
            destination: C:\Windows\temp\AWSCLISetup.exe
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: '{{ build.Download.inputs[0].destination }}'
          arguments:
            - '/install'
            - '/quiet'
            - '/norestart'
      - name: Delete
```

```
        action: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

예제 3: MSI 설치 관리자를 AWS CLI 사용하여 설치

다음은 MSI 설치 관리자를 AWS CLI 사용하여를 설치하는 예제 문서입니다.

```
name: InstallCLIMSI
description: Install &CLI; using the MSI installer
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://aws-cli/AWSCLI64PY3.msi
            destination: C:\Windows\temp\AWSCLI64PY3.msi
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: 'C:\Windows\System32\msiexec.exe'
          arguments:
            - '/i'
            - '{{ build.Download.inputs[0].destination }}'
            - '/quiet'
            - '/norestart'
      - name: Delete
        action: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

macOS

예제 1: 사용자 지정 macOS 바이너리 파일 실행

다음은 macOS 인스턴스에서 사용자 지정 바이너리 파일을 다운로드하고 실행하는 예제 문서입니다.

```
name: macOSBin
```

```
description: Download and run a binary file on macOS.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://<replaceable>amzn-s3-demo-source-bucket</replaceable>/
<replaceable>myapplication</replaceable>
            destination: /tmp/<replaceable>myapplication</replaceable>
      - name: Enable
        action: ExecuteBash
        onFailure: Continue
        inputs:
          commands:
            - 'chmod u+x {{ build.Download.inputs[0].destination }}'
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: '{{ build.Download.inputs[0].destination }}'
          arguments:
            - '--install'
      - name: Delete
        action: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

사용자 지정 구성 요소 문서에서 변수 사용

변수를 사용하면 애플리케이션 전체에서 사용할 수 있는 의미 있는 이름으로 데이터에 레이블을 지정할 수 있습니다. 복잡한 워크플로에 대해 간단하고 읽기 쉬운 형식으로 사용자 지정 변수를 정의하고 AWSTOE 구성 요소의 YAML 애플리케이션 구성 요소 문서에서 참조할 수 있습니다.

이 섹션에서는 구문, 이름 제약 조건 및 예제를 포함하여 AWSTOE YAML 애플리케이션 구성 요소 문서에서 구성 요소의 변수를 정의하는 데 도움이 되는 정보를 제공합니다.

상수

상수는 한 번 정의하면 수정하거나 재정의할 수 없는 변경 불가능한 변수입니다. 상수는 AWSTOE 문서의 `constants` 섹션에 있는 값을 사용하여 정의할 수 있습니다.

상수 이름 지정 규칙

- 이름은 3~128자 이내로 작성해야 합니다.
- 이름에는 영숫자(A-Z, a-z, 0-9), 하이픈(-) 또는 밑줄(_)만 포함될 수 있습니다.
- 단, 문서 내에서 고유 이름을 갖도록 합니다.
- 이름은 YAML 문자열로 지정되어야 합니다.

구문

```
constants:
  - <name>:
      type: <constant type>
      value: <constant value>
```

키 이름	필수	설명
name	예	상수의 이름. 문서에 대해 고유해야 합니다(다른 파라미터 이름 또는 상수와 같지 않아야 함).
value	예	상수의 값.
type	예	상수의 유형. 지원되는 string 유형

문서 내 참조 상수 값

다음과 같이 YAML 문서 내의 단계 또는 루프 입력에서 상수를 참조할 수 있습니다.

- 상수 참조는 대/소문자를 구분하며 이름이 정확하게 일치해야 합니다.
- 이름은 이중 중괄호 {{ *MyConstant* }}로 묶어야 합니다.
- 중괄호 내에는 공백이 허용되며 자동으로 잘립니다. 예를 들어, 다음 참조는 모두 유효합니다.

  {{ *MyConstant* }}, {{ *MyConstant*}}, {{*MyConstant* }}, {{*MyConstant*}}
- YAML 문서의 참조는 문자열(작은따옴표 또는 큰따옴표로 묶음)로 지정해야 합니다.

예: - {{ *MyConstant* }}(은)는 문자열로 식별되지 않으므로 유효하지 않습니다.

그러나 - '{{ *MyConstant* }}' 및 - "{{ *MyConstant* }}" 참조는 모두 유효합니다.

예시

단계 입력에서 참조되는 상수

```
name: Download AWS CLI version 2
schemaVersion: 1.0
constants:
  - Source:
      type: string
      value: https://awscli.amazonaws.com/AWSCLIV2.msi
phases:
  - name: build
    steps:
      - name: Download
        action: WebDownload
        inputs:
          - source: '{{ Source }}'
            destination: 'C:\Windows\Temp\AWSCLIV2.msi'
```

루프 입력에서 참조되는 상수

```
name: PingHosts
schemaVersion: 1.0
constants:
  - Hosts:
      type: string
      value: 127.0.0.1,amazon.com
phases:
  - name: build
    steps:
      - name: Ping
        action: ExecuteBash
        loop:
          forEach:
            list: '{{ Hosts }}'
            delimiter: ','
        inputs:
```

```
        commands:
          - ping -c 4 {{ loop.value }}
```

파라미터

파라미터는 호출 애플리케이션이 런타임에 제공할 수 있는 설정을 포함하는 변경 가능한 변수입니다. YAML 문서의 Parameters 섹션에서 파라미터를 정의할 수 있습니다.

파라미터 이름 규칙

- 이름은 3~128자 이내로 작성해야 합니다.
- 이름에는 영숫자(A-Z, a-z, 0-9), 하이픈(-) 또는 밑줄(_)만 포함될 수 있습니다.
- 단, 문서 내에서 고유 이름을 갖도록 합니다.
- 이름은 YAML 문자열로 지정되어야 합니다.

구문

```
parameters:
  - <name>:
      type: <parameter type>
      default: <parameter value>
      description: <parameter description>
```

키 이름	필수	설명
name	예	파라미터의 이름입니다. 문서에 대해 고유해야 합니다(다른 파라미터 이름 또는 상수와 같지 않아야 함).
type	예	파라미터의 데이터 유형입니다. 지원되는 유형에는 string(이)가 있습니다.
default	아니요	파라미터의 기본값입니다.
description	아니요	파라미터에 대해 설명합니다.

문서의 참조 파라미터 값

다음과 같이 YAML 문서 내의 단계 또는 루프 입력에서 파라미터를 참조할 수 있습니다.

- 파라미터 참조는 대/소문자를 구분하며 이름이 정확하게 일치해야 합니다.
- 이름은 {{ *MyParameter* }}라는 이중 중괄호로 묶어야 합니다.
- 중괄호 내에는 공백이 허용되며 자동으로 잘립니다. 예를 들어, 다음 참조는 모두 유효합니다.

  {{ *MyParameter* }}, {{ *MyParameter*}}, {{*MyParameter* }}, {{*MyParameter*}}
- YAML 문서의 참조는 문자열(작은따옴표 또는 큰따옴표로 묶음)로 지정해야 합니다.

  예: - {{ *MyParameter* }}(은)는 문자열로 식별되지 않으므로 유효하지 않습니다.

  그러나 - '{{ *MyParameter* }}' 및 - "{{ *MyParameter* }}" 참조는 모두 유효합니다.

예시

다음 예제는 YAML 문서에서 파라미터를 사용하는 방법을 보여 줍니다.

- 단계별 입력의 파라미터를 참조하세요.

```
name: Download AWS CLI version 2
schemaVersion: 1.0
parameters:
  - Source:
      type: string
      default: 'https://awscli.amazonaws.com/AWSCLIV2.msi'
      description: The AWS CLI installer source URL.
phases:
  - name: build
    steps:
      - name: Download
        action: WebDownload
        inputs:
          - source: '{{ Source }}'
            destination: 'C:\Windows\Temp\AWSCLIV2.msi'
```

- 루프 입력의 파라미터를 참조하세요.

```
name: PingHosts
schemaVersion: 1.0
```

```
parameters:
  - Hosts:
      type: string
      default: 127.0.0.1,amazon.com
      description: A comma separated list of hosts to ping.
phases:
  - name: build
    steps:
      - name: Ping
        action: ExecuteBash
        loop:
          forEach:
            list: '{{ Hosts }}'
            delimiter: ','
        inputs:
          commands:
            - ping -c 4 {{ loop.value }}
```

런타임 시 파라미터 재정의

키-값 페어와 AWS CLI 함께의 `--parameters` 옵션을 사용하여 런타임 시 파라미터 값을 설정할 수 있습니다.

- 파라미터 키-값 쌍을 등호(<name>=<value>)로 구분하여 이름과 값으로 지정합니다.
- 여러 파라미터는 쉼표로 구분해야 합니다.
- YAML 구성 요소 문서에 없는 파라미터 이름은 무시됩니다.
- 파라미터 이름과 값이 모두 필요합니다.

> **Important**
>
> 구성 요소 파라미터는 일반 텍스트 값이며 AWS CloudTrail에 기록됩니다. AWS Secrets Manager 또는 AWS Systems Manager 파라미터 스토어를 사용하여 보안 암호를 저장하는 것이 좋습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager란 무엇입니까?를 참조하십시오. AWS Systems Manager Parameter Store에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Parameter Store 섹션을 참조하십시오.

구문

```
--parameters name1=value1,name2=value2...
```

CLI 옵션	필수	설명
--parameters *name*=*value*,...	아니요	이 옵션은 파라미터 이름을 키로 사용하여 키-값 페어의 목록을 가져옵니다.

예시

다음 예제는 YAML 문서에서 파라미터를 사용하는 방법을 보여 줍니다.

- 이 `--parameter` 옵션에 지정된 파라미터 키-값 쌍은 유효하지 않습니다.

  ```
  --parameters ntp-server=
  ```

- AWS CLI에서 `--parameter` 옵션을 사용하여 하나의 파라미터 키-값 쌍을 설정합니다.

  ```
  --parameters ntp-server=ntp-server-windows-qe.us-east1.amazon.com
  ```

- AWS CLI에서 `--parameter` 옵션을 사용하여 여러 파라미터 키-값 쌍을 설정합니다.

  ```
  --parameters ntp-server=ntp-server.amazon.com,http-url=https://internal-us-
  east1.amazon.com
  ```

Systems Manager Parameter Store 파라미터 사용

로 변수 접두사를 지정하여 구성 요소 문서에서 AWS Systems Manager Parameter Store 파라미터(SSM 파라미터)를 참조할 수 있습니다`aws:ssm`. 예:

`{{ aws:ssm:/my/param }}`는 SSM 파라미터의 값으로 해석됩니다`/my/param`.

이 기능은 다음 SSM 파라미터 유형을 지원합니다.

- 문자열 - AWSTOE 문자열 유형에 매핑됩니다.
- StringList - 유형에 매핑됩니다 AWSTOE `stringList`.

- SecureString - AWSTOE 문자열 유형에 매핑됩니다.

파라미터 스토어에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager 파라미터 스토어를 참조하세요.

SSM 파라미터를 사용하여 AWS Secrets Manager 보안 암호를 참조할 수도 있습니다SecureString. 예를 들어 `{{ aws:ssm:/aws/reference/secretsmanager/test/test-secret }}`입니다. 자세한 내용은 파라미터 스토어 파라미터에서 보안 암호 참조를 AWS Secrets Manager 참조하세요.

> **Important**
>
> Image Builder는 로그에서 `SecureString` 파라미터 해상도를 제외합니다. 그러나 구성 요소 문서에서 실행된 명령을 통해 민감한 정보가 기록되지 않도록 할 책임도 있습니다. 예를 들어 보안 문자열과 함께 `echo` 명령을 사용하는 경우 명령은 로그에 일반 텍스트 값을 씁니다.

필수 IAM 권한

구성 요소에서 Systems Manager 파라미터를 사용하려면 인스턴스 역할에 파라미터 리소스 ARN에 대한 `ssm:GetParameter` 권한이 있어야 합니다. 예시:

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "ssm:GetParameter",
   "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
  }
 ]
}
```

암호화된 값에 액세스하려면 다음 권한도 필요합니다.

- 고객 관리형으로 암호화된 `SecureString` 파라미터 또는 AWS Secrets Manager 값에 `kms:Decrypt`를 추가합니다 AWS KMS key.
- Secrets Manager 보안 암호를 참조하는 `secretsmanager:GetSecretValue` 경우를 추가합니다.

구성 요소 문서에서 SSM 파라미터 참조

다음 예제에서는 구성 요소에서 Systems Manager 파라미터의 Systems Manager 파라미터 스토어 파라미터를 참조하는 방법을 보여줍니다.

```
name: UseSSMParameterVariable
description: This is a sample component document that prints out the value of an SSM
 Parameter. Never do this for a SecureString parameter.
schemaVersion: 1.0

phases:
  - name: verify
    steps:
      - name: EchoParameterValue
        action: ExecuteBash
        inputs:
          commands:
            - echo "Log SSM parameter name: /my/test/param, value {{ aws:ssm:/my/test/
param }}."
```

SSM 파라미터에 대한 동적 런타임 변수 확인

AWSTOE는 변수 참조 내에서 런타임 시 값을 조작하거나 변환하는 데 사용할 수 있는 다음과 같은 내장 함수를 제공합니다.

해석 함수

`resolve` 함수는 다른 변수 참조 내부의 변수 참조를 해석하여 동적 변수 이름 참조를 허용합니다. 이는 파라미터 경로의 일부가 가변적이고 문서 파라미터로 전달될 수 있는 SSM 파라미터로 작업할 때 유용합니다.

`resolve` 함수는 SSM 파라미터의 이름 부분에 대한 동적 확인만 지원합니다.

구문

다음 예제`dynamic_variable`의는 SSM 파라미터의 이름을 나타내며 다음 중 하나여야 합니다.

- SSM 파라미터 참조(예: `aws:ssm:/my/param`)
- 구성 요소 문서 파라미터 참조(예: *`parameter-name`*)

```
{{ aws:ssm:resolve(dynamic_variable) }}
```

예: 런타임 시 SSM 파라미터 확인

다음 예제에서는 YAML 구성 요소 문서에서 `resolve` 함수를 사용하는 방법을 보여줍니다.

```
name: SsmParameterTest
description: This component verifies an SSM parameter variable reference with the echo
 command.
schemaVersion: 1.0

parameters:
  - parameter-name:
      type: string
      description: "test"

phases:
  - name: validate
    steps:
      - name: PrintDynamicVariable
        action: ExecuteBash
        inputs:
          commands:
            - echo "{{ aws:ssm:resolve(parameter-name) }}"
```

AWSTOE에서 조건부 구문 사용

조건부 구문은 지정된 조건부 표현식이 `true` 또는 `false`로 평가되는지 여부에 따라 구성 요소 문서에서 다양한 작업을 수행합니다. `if` 구문을 사용하여 구성 요소 문서의 실행 흐름을 제어할 수 있습니다.

if 구문

`if` 구문을 사용하여 단계를 실행해야 하는지 여부를 평가할 수 있습니다. 기본적으로 `if` 조건식이 `true`로 평가되면 AWSTOE 는 단계를 실행하고 조건이 `false`로 평가되면 AWSTOE 는 단계를 건너뜁니다. 단계를 건너뛰면 AWSTOE 가 단계와 문서가 성공적으로 실행되었는지 평가할 때 성공 단계로 처리됩니다.

> **Note**
>
> 단계가 재시작을 트리거하더라도 `if` 문은 한 번만 평가됩니다. 단계가 다시 시작되면 `if` 문이 이미 평가되었음을 인식하고 중단된 위치에서 계속됩니다.

구문

```
if:
  - <conditional expression>:
      [then: <step action>]
      [else: <step action>]
```

키 이름	필수	설명
조건식	예	조건식에는 최상위 수준에서 다음 유형의 연산자 중 정확히 하나가 포함될 수 있습니다. • 비교 연산자 - 비교 연산자 목록과 AWSTOE 구성 요소 문서에서 이러한 연산자가 작동하는 방식에 대한 자세한 내용은 섹션을 참조하세요비교 연산자. • 논리 연산자 - 논리 연산자에는 and, or 및 not이 포함되며 하나 이상의 비교 연산자에서 작동합니다. 논리 연산자가 AWSTOE 구성 요소 문서에서 작동하는 방식에 대한 자세한 내용은 논리 연산자 섹션을 참조하세요. 표현식이 여러 조건을 충족해야 하는 경우 논리 연산자를 사용하여 조건을 지정합니다.
then	아니요	

키 이름	필수	설명
		조건식이 `true`로 평가되는 경우 수행할 작업을 정의합니다.
else	아니요	조건식이 `false`로 평가되는 경우 수행할 작업을 정의합니다.
단계 작업	조건	`then` 또는 `else`를 사용할 때는 다음 단계 작업 중 하나를 지정해야 합니다. • 중단 - AWSTOE 에서 단계를 실패로 표시합니다. • 실행 - AWSTOE 에서 단계를 실행합니다. • 건너뛰기 - AWSTOE 에서 단계를 건너뜁니다.

예제 1: 패키지 설치

AWSTOE 구성 요소 문서의 다음 예제 단계는 논리 연산자를 사용하여 파라미터 값을 테스트하고 패키지의 압축이 풀린 경우 적절한 패키지 관리자 명령을 실행하여 애플리케이션을 설치합니다.

```
- name: InstallUnzipAptGet
    action: ExecuteBash
    if:
      and:
          - binaryExists: 'apt-get'
          - not:
              binaryExists: 'unzip'
    inputs:
      commands:
          - sudo apt-get update
          - sudo apt-get install -y unzip
```

```
    - name: InstallUnzipYum
      action: ExecuteBash
      if:
        and:
            - binaryExists: 'yum'
            - not:
                binaryExists: 'unzip'
      inputs:
        commands:
            - sudo yum install -y unzip

    - name: InstallUnzipZypper
      action: ExecuteBash
      if:
        and:
            - binaryExists: 'zypper'
            - not:
                binaryExists: 'unzip'
      inputs:
        commands:
            - sudo zypper refresh
            - sudo zypper install -y unzip
```

예제 2: 단계 건너뛰기

다음 예제에서는 단계를 건너뛰는 두 가지 방법을 보여줍니다. 하나는 논리적 연산자를 사용하고 다른 하나는 Skip 단계 작업과 비교 연산자를 사용합니다.

```
# Creates a file if it does not exist using not
- name: CreateMyConfigFile-1
  action: ExecuteBash
  if:
    not:
      fileExists: '/etc/my_config'
  inputs:
    commands:
      - echo "Hello world" > '/etc/my_config'

# Creates a file if it does not exist using then and else
- name: CreateMyConfigFile-2
  action: ExecuteBash
  if:
```

```
    fileExists: '/etc/my_config'
    then: Skip
    else: Execute
  inputs:
    commands:
      - echo "Hello world" > '/etc/my_config'
```

AWSTOE 구성 요소 문서에서 비교 연산자 사용

다음 비교 연산자를 Assert 작업 모듈 및 if 구문를 사용하는 조건식과 함께 사용할 수 있습니다. 비교 연산자는 예를 들어 `stringIsEmpty`와 같은 단일 값으로 작동하거나 기준 값을 두 번째 값(변수 값)과 비교하여 조건식이 `true` 또는 `false`로 평가되는지 여부를 결정할 수 있습니다.

두 값에 대해 비교가 작동하는 경우 두 번째 값은 체인 변수일 수 있습니다.

다른 유형의 값을 비교할 때 비교 전에 다음과 같은 값 변환이 발생할 수 있습니다.

- 숫자 비교의 경우 변수 값이 문자열인 경우는 평가 전에 문자열을 숫자로 AWSTOE 변환합니다. 변환이 불가능한 경우 비교에서 `false`를 반환합니다. 예를 들어 변수 값이 `"1.0"`인 경우 변환이 작동하지만 변수 값이 `"a10"`인 경우 변환이 실패합니다.
- 문자열 비교의 경우 변수 값이 숫자인 경우는 평가 전에 문자열로 AWSTOE 변환합니다.

문자열 비교

다음 비교 연산자는 문자열을 사용하여 값을 비교하거나, 스페이스 또는 빈 문자열을 테스트하거나, 입력 값을 정규식 패턴과 비교합니다. 문자열 비교에서는 대/소문자를 구분하지 않으며 문자열 입력의 시작 또는 끝에 있는 스페이스를 제거하지 않습니다.

문자열 비교 연산자

- stringIsEmpty
- stringIsWhitespace
- stringEquals
- stringLessThan
- stringLessThanEquals
- stringGreaterThan
- stringGreaterThanEquals
- patternMatches

stringIsEmpty

지정된 문자열에 문자가 없는 경우 `stringIsEmpty` 연산자가 `true`를 반환합니다. 예시:

```
# Evaluates to true
stringIsEmpty: ""

# Evaluates to false
stringIsEmpty: " "

# Evaluates to false
stringIsEmpty: "Hello."
```

stringIsWhitespace

`stringIsWhitespace`에 지정된 문자열에 스페이스만 포함되어 있는지 테스트합니다. 예시:

```
# Evaluates to true
stringIsWhitespace: "   "

# Evaluates to false
stringIsWhitespace: ""

# Evaluates to false
stringIsWhitespace: " Hello?"
```

stringEquals

`stringEquals`에 지정된 문자열이 `value` 파라미터에 지정된 문자열과 정확히 일치하는지 테스트합니다. 예시:

```
# Evaluates to true
stringEquals: 'Testing, testing...'
value: 'Testing, testing...'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'Hello again.'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'TESTING, TESTING....'
```

```
# Evaluates to false
stringEquals: 'Testing, testing...'
value: '   Testing, testing...'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'Testing, testing...   '
```

stringLessThan

`stringLessThan`에 지정된 문자열이 `value` 파라미터에 지정된 문자열보다 작은지 테스트합니다. 예시:

```
# Evaluates to true
# This comparison operator isn't case sensitive
stringlessThan: 'A'
value: 'a'

# Evaluates to true - 'a' is less than 'b'
stringlessThan: 'b'
value: 'a'

# Evaluates to true
# Numeric strings compare as less than alphabetic strings
stringlessThan: 'a'
value: '0'

# Evaluates to false
stringlessThan: '0'
value: 'a'
```

stringLessThanEquals

`stringLessThanEquals`에 지정된 문자열이 `value` 파라미터에 지정된 문자열보다 작거나 같은지 테스트합니다. 예시:

```
# Evaluates to true - 'a' is equal to 'a'
stringLessThanEquals: 'a'
value: 'a'

# Evaluates to true - since the comparison isn't case sensitive, 'a' is equal to 'A'
stringLessThanEquals: 'A'
```

```
value: 'a'

# Evaluates to true - 'a' is less than 'b'
stringLessThanEquals: 'b'
value: 'a'

# Evaluates to true - '0' is less than 'a'
stringLessThanEquals: 'a'
value: '0'

# Evaluates to false - 'a' is greater than '0'
stringLessThanEquals: '0'
value: 'a'
```

stringGreaterThan

`stringGreaterThan`에 지정된 문자열이 `value` 파라미터에 지정된 문자열보다 큰지 테스트합니다. 예시:

```
# Evaluates to false - since the comparison isn't case sensitive, 'A' is equal to
 'a'
stringGreaterThan: 'a'
value: 'A'

# Evaluates to true - 'b' is greater than 'a'
stringGreaterThan: 'a'
value: 'b'

# Evaluates to true - 'a' is greater than '0'
stringGreaterThan: '0'
value: 'a'

# Evaluates to false - '0' is less than 'a'
stringGreaterThan: 'a'
value: '0'
```

stringGreaterThanEquals

`stringGreaterThanEquals`에 지정된 문자열이 `value` 파라미터에 지정된 문자열보다 크거나 같은지 테스트합니다. 예시:

```
# Evaluates to true - 'a' is equal to 'A'
stringGreaterThanEquals: 'A'
```

```
value: 'a'

# Evaluates to true - 'b' is greater than 'a'
stringGreaterThanEquals: 'a'
value: 'b'

# Evaluates to true - 'a' is greater than '0'
stringGreaterThanEquals: '0'
value: 'a'

# Evaluates to false - '0' is less than 'a'
stringGreaterThanEquals: 'a'
value: '0'
```

patternMatches

`value` 파라미터에 지정된 문자열이 `patternMatches`에 지정된 정규식 패턴과 일치하는지 테스트합니다. 비교에서 RE2 구문을 준수하는 Golang regexp 패키지를 사용합니다. RE2 규칙에 대한 자세한 내용은 GitHub의 Google / re2 리포지토리를 참조하세요.

다음 예제에서는 `true`를 반환하는 패턴 일치를 보여줍니다.

```
patternMatches: '^[a-z]+$'
value: 'ThisIsValue'
```

숫자 비교

다음 비교 연산자는 숫자에 적용됩니다. 이러한 연산자에 대해 제공된 값은 YAML 사양에 따라 다음 유형 중 하나여야 합니다. 숫자 비교에 대한 지원은 예를 들어 func(*Float) Cmp와 같은 golang big 패키지 비교 연산자를 사용합니다.

- Integer
- Float(-1.7e+308~+1.7e+308의 숫자를 지원하는 float64 기반)
- 다음 정규식 패턴과 일치하는 문자열: `^[-+]?([0-9]+[.])?[0-9]+$`

숫자 비교 연산자

- numberEquals
- numberLessThan

- numberLessThanEquals
- numberGreaterThan
- numberGreaterThanEquals

numberEquals

numberEquals에 지정된 숫자가 value 파라미터에 지정된 숫자와 동일한지 테스트합니다. 다음 예제 비교에서는 모두 true를 반환합니다.

```
# Values provided as a positive number
numberEquals: 1
value: 1

# Comparison value provided as a string
numberEquals: '1'
value: 1

# Value provided as a string
numberEquals: 1
value: '1'

# Values provided as floats
numberEquals: 5.0
value: 5.0

# Values provided as a negative number
numberEquals: -1
value: -1
```

numberLessThan

numberLessThan에 지정된 숫자가 value 파라미터에 지정된 숫자보다 작은지 테스트합니다. 예시:

```
# Evaluates to true
numberLessThan: 2
value: 1

# Evaluates to true
numberLessThan: 2
value: 1.9
```

```
# Evaluates to false
numberLessThan: 2
value: '2'
```

numberLessThanEquals

`numberLessThanEquals`에 지정된 숫자가 `value` 파라미터에 지정된 숫자보다 작거나 같은지 테스트합니다. 예시:

```
# Evaluates to true
numberLessThanEquals: 2
value: 1

# Evaluates to true
numberLessThanEquals: 2
value: 1.9

# Evaluates to true
numberLessThanEquals: 2
value: '2'

# Evaluates to false
numberLessThanEquals: 2
value: 2.1
```

numberGreaterThan

`numberGreaterThan`에 지정된 숫자가 `value` 파라미터에 지정된 숫자보다 큰지 테스트합니다. 예시:

```
# Evaluates to true
numberGreaterThan: 1
value: 2

# Evaluates to true
numberGreaterThan: 1
value: 1.1

# Evaluates to false
numberGreaterThan: 1
value: '1'
```

numberGreaterThanEquals

numberGreaterThanEquals에 지정된 숫자가 value 파라미터에 지정된 숫자보다 크거나 같은지 테스트합니다. 예시:

```
# Evaluates to true
numberGreaterThanEquals: 1
value: 2

# Evaluates to true
numberGreaterThanEquals: 1
value: 1.1

# Evaluates to true
numberGreaterThanEquals: 1
value: '1'

# Evaluates to false
numberGreaterThanEquals: 1
value: 0.8
```

파일 확인

다음 비교 연산자는 파일 해시를 확인하거나 파일 또는 폴더가 있는지 확인합니다.

파일 및 폴더 연산자

- binaryExists
- fileExists
- folderExists
- fileMD5Equals
- fileSHA1Equals
- fileSHA256Equals
- fileSHA512Equals

binaryExists

현재 경로에서 애플리케이션을 사용할 수 있는지 테스트합니다. 예시:

```
binaryExists: 'foo'
```

> **Note**
>
> Linux 및 macOS 시스템에서 *foo*라는 애플리케이션의 경우 bash 명령 type ***foo*** >/dev/null 2>&1와 동일하게 작동합니다. 여기서 $? == 0은 성공적인 비교를 나타냅니다.
> Windows 시스템에서 *foo*라는 애플리케이션의 경우 PowerShell 명령 & C:\Windows\System32\where.exe /Q ***foo***와 동일하게 작동합니다. 여기서 $LASTEXITCODE = 0은 성공적인 비교를 나타냅니다.

fileExists

파일이 지정된 경로에 존재하는지 테스트합니다. 절대 또는 상대 경로를 제공할 수 있습니다. 지정한 위치가 존재하고 파일인 경우 비교는 `true`로 평가됩니다. 예시:

```
fileExists: '/path/to/file'
```

> **Note**
>
> Linux 및 macOS 시스템에서는 bash 명령 -d ***/path/to/file***와 동일하게 작동합니다. 여기서 $? == 0은 성공적인 비교를 나타냅니다.
> Windows 시스템에서는 PowerShell 명령 Test-Path -Path '***C:\path\to\file***' -PathType 'Leaf'와 동일하게 작동합니다.

folderExists

폴더가 지정된 경로에 존재하는지 테스트합니다. 절대 또는 상대 경로를 제공할 수 있습니다. 지정한 위치가 존재하고 폴더인 경우 비교는 `true`로 평가됩니다. 예시:

```
folderExists: '/path/to/folder'
```

> **Note**
>
> Linux 및 macOS 시스템에서는 bash 명령 -d ***/path/to/folder***와 동일하게 작동합니다. 여기서 $? == 0은 성공적인 비교를 나타냅니다.

Windows 시스템에서는 PowerShell 명령 Test-Path -Path '*C:\path\to\folder*' -PathType 'Container'와 동일하게 작동합니다.

fileMD5Equals

파일의 MD5 해시가 지정된 값과 동일한지 테스트합니다. 예시:

```
fileMD5Equals: '<MD5Hash>'
path: '/path/to/file'
```

fileSHA1Equals

파일의 SHA1 해시가 지정된 값과 동일한지 테스트합니다. 예시:

```
fileSHA1Equals: '<SHA1Hash>'
path: '/path/to/file'
```

fileSHA256Equals

파일의 SHA256 해시가 지정된 값과 동일한지 테스트합니다. 예시:

```
fileSHA256Equals: '<SHA256Hash>'
path: '/path/to/file'
```

fileSHA512Equals

파일의 SHA512 해시가 지정된 값과 동일한지 테스트합니다. 예시:

```
fileSHA512Equals: '<SHA512Hash>'
path: '/path/to/file'
```

AWSTOE 구성 요소 문서에서 논리 연산자 사용

다음 논리 연산자를 사용하여 구성 요소 문서에서 조건식을 추가하거나 수정할 수 있습니다.는 조건이 지정된 순서대로 조건식을 AWSTOE 평가합니다. 구성 요소 문서의 비교 연산자에 대한 자세한 내용은 AWSTOE 구성 요소 문서에서 비교 연산자 사용 섹션을 참조하세요.

그리고

and 연산자를 사용하면 두 개 이상의 비교를 단일 표현식으로 평가할 수 있습니다. 이 표현식은 목록의 모든 조건이 true일 때 `true`로 평가됩니다. 그렇지 않으면 표현식은 `false`로 평가됩니다.

예시:

다음 예제에서는 문자열과 숫자의 두 가지 비교를 수행합니다. 두 비교 모두 true이므로 표현식은 true로 평가됩니다.

```
and:
  - stringEquals: 'test_string'
    value: 'test_string'
  - numberEquals: 1
    value: 1
```

다음 예제에서도 두 가지 비교를 수행합니다. 첫 번째 비교는 false로, 이 시점에서 평가가 중지되고 두 번째 비교는 건너뜁니다. 표현식은 `false`로 평가됩니다.

```
and:
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 1
```

또는

or 연산자를 사용하면 두 개 이상의 비교를 단일 표현식으로 평가할 수 있습니다. 표현식은 지정된 비교 중 하나가 true일 때 `true`로 평가됩니다. 지정된 비교 중 어느 것도 `true`로 평가되지 않으면 표현식은 `false`로 평가됩니다.

예시:

다음 예제에서는 문자열과 숫자의 두 가지 비교를 수행합니다. 첫 번째 비교는 true이므로 표현식은 `true`로 평가되고 두 번째 비교는 건너뜁니다.

```
or:
  - stringEquals: 'test_string'
    value: 'test_string'
  - numberEquals: 1
```

```
    value: 3
```

다음 예제에서도 두 가지 비교를 수행합니다. 첫 번째 비교는 false이며 평가는 계속됩니다. 두 번째 비교는 true이므로 표현식은 `true`로 평가됩니다.

```
or:
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 1
```

마지막 예제에서는 두 비교가 모두 false이므로 표현식이 `false`로 평가됩니다.

```
or:
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 3
```

not

`not` 연산자를 사용하면 단일 비교를 부정할 수 있습니다. 비교가 false인 경우 표현식이 `true`로 평가됩니다. 비교가 true인 경우 표현식이 `false`로 평가됩니다.

예시:

다음 예제에서는 문자열 비교를 수행합니다. 비교가 false이므로 표현식이 `true`로 평가됩니다.

```
not:
  - stringEquals: 'test_string'
    value: 'Hello world!'
```

다음 예제에서도 문자열 비교를 수행합니다. 비교가 true이므로 표현식이 `false`로 평가됩니다.

```
not:
  - stringEquals: 'test_string'
    value: 'test_string'
```

AWSTOE에서 루프 구문 사용

이 섹션에서는 AWSTOE에서 반복 구문을 작성하는 데 도움이 되는 정보를 제공합니다. 반복 구문은 반복되는 명령 시퀀스를 정의합니다. AWSTOE에서는 다음과 같은 유형의 반복 구문을 사용할 수 있습니다.

- `for` 구문 - 한정된 정수 시퀀스를 반복합니다.
- `forEach` 구문
  - 입력 목록이 있는 `forEach` 루프 - 유한한 문자열 컬렉션을 반복합니다.
  - 구분된 목록이 있는 `forEach` 루프 - 구분 기호로 연결된 유한한 문자열 컬렉션을 반복합니다.

> **Note**
>
> 반복 구문은 문자열 데이터 유형만 지원합니다.

구문 주제 반복

- 참조 반복 변수
- 반복 구문의 유형
- 단계 필드
- 단계별 및 반복 출력

참조 반복 변수

현재 반복 변수의 인덱스와 값을 참조하려면 반복 구문이 포함된 단계의 입력 본문 내에서 참조 표현식 `{{ loop.* }}`(을)를 사용해야 합니다. 이 표현식은 다른 단계의 반복 구문의 반복 변수를 참조하는 데 사용할 수 없습니다.

참조 표현식은 다음 멤버로 구성됩니다.

- `{{ loop.index }}` - `0`에서 인덱싱된 현재 반복의 순서 위치입니다.
- `{{ loop.value }}` - 현재 반복 변수와 관련된 값입니다.

루프 이름

모든 반복 구문에는 식별을 위한 선택적 이름 필드가 있습니다. 루프 이름을 제공하면 이 이름을 사용하여 단계 입력 본문에 있는 반복 변수를 참조할 수 있습니다. 명명된 반복의 반복 인덱스 및 값을 참조하려면 단계의 입력 본문에서 `{{ <loop_name>.* }}` 및 `{{ loop.* }}`(을)를 사용하세요. 이 표현식은 다른 단계의 명명된 반복 구문을 참조하는 데 사용할 수 없습니다.

참조 표현식은 다음 멤버로 구성됩니다.

- `{{ <loop_name>.index }}` - 0에서 인덱싱되는 명명된 반복의 현재 반복의 순서 위치입니다.
- `{{ <loop_name>.value }}` - 명명된 루프의 현재 반복 변수와 관련된 값입니다.

참조 표현식 해결

는 다음과 같이 참조 표현식을 AWSTOE 확인합니다.

- `{{ <loop_name>.* }}` –는 다음 로직을 사용하여이 표현식을 AWSTOE 해결합니다.
  - 현재 실행 중인 단계의 루프가 `<loop_name>` 값과 일치하면 참조 표현식은 현재 실행 중인 단계의 반복 구조로 해석됩니다.
  - `<loop_name>`에서 명명된 반복 구조가 현재 실행 중인 단계 내에 나타나는 경우 해당 구문으로 해석됩니다.
- `{{ loop.* }}` - 현재 실행 중인 단계에 정의된 반복 구문을 사용하여 표현식을 AWSTOE 해결합니다.

루프가 포함되지 않은 단계에서 참조 표현식을 사용하는 경우는 표현식을 확인하지 AWSTOE 않으며 대체 없이 단계에 표시됩니다.

> **Note**
>
> YAML 컴파일러에서 올바르게 해석되려면 참조 표현식을 큰따옴표로 묶어야 합니다.

반복 구문의 유형

이 섹션에서는 AWSTOE에서 사용할 수 있는 반복 구문 유형에 대한 정보와 예제를 제공합니다.

반복 구문 유형

- for 루프

- 입력 목록이 있는 forEach 루프
- 구분된 목록이 있는 forEach 루프

for 루프

for 루프는 변수의 시작과 끝으로 윤곽이 그려진 경계 내에 지정된 정수 범위에 대해 반복됩니다. 반복 값은 세트 [start, end] 내에 있으며 경계 값을 포함합니다.

AWSTOE 는 start, end및 updateBy 값을 확인하여 조합으로 인해 무한 루프가 발생하지 않도록 합니다.

for 루프 스키마

```
  - name: "StepName"
    action: "ActionModule"
    loop:
      name: "string"
      for:
        start: int
        end: int
        updateBy: int
inputs:
  ...
```

for 루프 입력

필드	설명	형식	필수	기본값
name	루프의 고유 이름. 동일한 단계의 다른 루프 이름과 비교하여 고유해야 합니다.	String	No	""
start	반복 시작 값. 체인 표현식은 허용되지 않습니다.	Integer	예	해당 사항 없음

필드	설명	형식	필수	기본값
end	반복 종료 값. 체인 표현식은 허용되지 않습니다.	Integer	예	해당 사항 없음
updateBy	덧셈을 통해 반복 값이 업데이트되는 경우의 차이. 0이 아닌 음수 또는 양수여야 합니다. 체인 표현식은 허용되지 않습니다.	Integer	예	해당 사항 없음

for 루프 입력 예제

```
  - name: "CalculateFileUploadLatencies"
    action: "ExecutePowerShell"
    loop:
      for:
        start: 100000
        end: 1000000
        updateBy: 100000
    inputs:
      commands:
        - |
          $f = new-object System.IO.FileStream c:\temp\test{{ loop.index }}.txt,
 Create, ReadWrite
          $f.SetLength({{ loop.value }}MB)
          $f.Close()
        - c:\users\administrator\downloads\latencyTest.exe --file c:\temp
\test{{ loop.index }}.txt
        - AWS s3 cp c:\users\administrator\downloads\latencyMetrics.json s3://bucket/
latencyMetrics.json
        - |
          Remove-Item -Path c:\temp\test{{ loop.index }}.txt
          Remove-Item -Path c:\users\administrator\downloads\latencyMetrics.json
```

입력 목록이 있는 **forEach** 루프

forEach 루프는 문자열과 체인 표현식일 수 있는 명시적 값 목록에서 반복됩니다.

입력 목록 스키마가 있는 forEach 루프

```
- name: "StepName"
  action: "ActionModule"
  loop:
    name: "string"
    forEach:
      - "string"
  inputs:
...
```

입력 목록 입력이 있는 **forEach** 루프

필드	설명	형식	필수	기본값
name	루프의 고유 이름. 동일한 단계의 다른 루프 이름과 비교하여 고유해야 합니다.	String	No	""
forEach 루프 문자열 목록	반복할 문자열 목록. 연결된 표현식을 목록에서 문자열로 받아들입니다. YAML 컴파일러가 올바르게 해석하려면 체인 표현식을 큰따옴표로 묶어야 합니다.	문자열 목록	예	해당 사항 없음

입력 목록이 있는 forEach 루프 예제 1

```
- name: "ExecuteCustomScripts"
```

```
    action: "ExecuteBash"
    loop:
      name: BatchExecLoop
      forEach:
        - /tmp/script1.sh
        - /tmp/script2.sh
        - /tmp/script3.sh
    inputs:
      commands:
        - echo "Count {{ BatchExecLoop.index }}"
        - sh "{{ loop.value }}"
        - |
          retVal=$?
          if [ $retVal -ne 0 ]; then
            echo "Failed"
          else
            echo "Passed"
         fi
```

입력 목록이 있는 forEach 루프 예제 2

```
- name: "RunMSIWithDifferentArgs"
  action: "ExecuteBinary"
  loop:
    name: MultiArgLoop
    forEach:
      - "ARG1=C:\Users ARG2=1"
      - "ARG1=C:\Users"
      - "ARG1=C:\Users ARG3=C:\Users\Administrator\Documents\f1.txt"
  inputs:
    commands:
      path: "c:\users\administrator\downloads\runner.exe"
      args:
        - "{{ MultiArgLoop.value }}"
```

입력 목록이 있는 forEach 루프 예제 3

```
- name: "DownloadAllBinaries"
  action: "S3Download"
  loop:
    name: MultiArgLoop
    forEach:
      - "bin1.exe"
```

```
        - "bin10.exe"
        - "bin5.exe"
    inputs:
      - source: "s3://bucket/{{ loop.value }}"
        destination: "c:\temp\{{ loop.value }}"
```

구분된 목록이 있는 **forEach** 루프

루프는 구분자로 구분된 값을 포함하는 문자열을 반복합니다. 문자열의 구성 요소를 반복하려면 구분 기호를 AWSTOE 사용하여 문자열을 반복에 적합한 배열로 분할합니다.

구분된 목록 스키마가 있는 forEach 루프

```
- name: "StepName"
  action: "ActionModule"
  loop:
    name: "string"
    forEach:
      list: "string"
      delimiter: ".,;:\n\t -_"
  inputs:
...
```

구분된 목록 입력이 있는 **forEach** 루프

필드	설명	형식	필수	기본값
name	루프에 부여된 고유 이름입니다. 동일한 단계의 다른 루프 이름과 비교할 때 고유해야 합니다.	String	No	""
list	구성 문자열이 공통 구분 문자로 연결된 문자열로 구성된 문자열입니다. 체인 표현식도 사용할 수 있습니다. 연결	String	예	해당 사항 없음

필드	설명	형식	필수	기본값
	된 표현식의 경우 YAML 컴파일러가 올바르게 해석할 수 있도록 큰 따옴표로 묶어야 합니다.			

필드	설명	형식	필수	기본값
delimiter	블록 내에서 문자열을 구분하는 데 사용되는 문자입니다. 기본값은 쉼표 문자입니다. 지정된 목록에서 구분자를 하나만 사용할 수 있습니다. • 점: "." • 쉼표: "," • 세미콜론: ";" • 콜론: ":" • 줄 바꿈: "\n" • 탭: "\t" • 스페이스: " " • 하이픈: "-" • 밑줄: "_" 체인 표현식은 사용할 수 없습니다.	String	No	쉼표: ","

> **Note**
>
> list의 값은 변경할 수 없는 문자열로 취급됩니다. 런타임 중에 list 소스가 변경된 경우, 실행 중에는 반영되지 않습니다.

구분된 목록이 있는 forEach 루프 예제 1

이 예제에서는 다음 체인 표현식 패턴을 사용하여 다른 단계의 출력(<phase_name>.<step_name>.[inputs | outputs].<var_name>)을 참조합니다.

```
- name: "RunMSIs"
  action: "ExecuteBinary"
  loop:
    forEach:
      list: "{{ build.GetAllMSIPathsForInstallation.outputs.stdout }}"
      delimiter: "\n"
  inputs:
    commands:
      path: "{{ loop.value }}"
```

구분된 목록이 있는 forEach 루프 예제 2

```
- name: "UploadMetricFiles"
  action: "S3Upload"
  loop:
    forEach:
      list: "/tmp/m1.txt,/tmp/m2.txt,/tmp/m3.txt,..."
  inputs:
    commands:
      - source: "{{ loop.value }}"
        destination: "s3://bucket/key/{{ loop.value }}"
```

단계 필드

루프는 단계의 일부입니다. 단계 실행과 관련된 모든 필드는 개별 반복에 적용되지 않습니다. 단계 필드는 다음과 같이 단계 수준에서만 적용됩니다.

- timeoutSeconds - 루프의 모든 루프는 이 필드에 지정된 기간 내에 실행되어야 합니다. 루프 실행 시간이 초과되면는 단계의 재시도 정책을 AWSTOE 실행하고 새 시도마다 제한 시간 파라미터를 재설

정합니다. 최대 재시도 횟수에 도달한 후 루프 실행이 제한 시간 값을 초과하면, 루프 실행 시간이 초과되었다는 내용의 단계 실패 메시지가 나타납니다.

- onFailure – 실패 처리가 단계에 다음과 같이 적용됩니다.
  - onFailure가 로 설정된 경우 루프를 `Abort` AWSTOE 종료하고 재시도 정책에 따라 단계를 재시도합니다. 최대 재시도 횟수가 지나면는 현재 단계를 실패로 AWSTOE 표시하고 프로세스 실행을 중지합니다.

    AWSTOE 는 상위 단계 및 문서의 상태 코드를 로 설정합니다`Failed`.

    > **Note**
    >
    > 실패한 단계 이후에는 더 이상 단계가 실행되지 않습니다.

  - onFailure가 `Continue`(으)로 설정된 경우, AWSTOE 에서 루프를 종료하고 재시도 정책에 따라 단계를 재시도합니다. 최대 재시도 횟수가 지나면는 현재 단계를 실패로 AWSTOE 표시하고 다음 단계를 계속 실행합니다.

    AWSTOE 는 상위 단계 및 문서의 상태 코드를 로 설정합니다`Failed`.

  - onFailure가 `Ignore`(으)로 설정된 경우, AWSTOE 에서 루프를 종료하고 재시도 정책에 따라 단계를 재시도합니다. 최대 재시도 횟수가 지나면는 현재 단계를 로 AWSTOE 표시하고 다음 단계를 `IgnoredFailure`계속 실행합니다.

    AWSTOE 는 상위 단계 및 문서의 상태 코드를 로 설정합니다`SuccessWithIgnoredFailure`.

    > **Note**
    >
    > 이는 여전히 성공적인 실행으로 간주되지만 하나 이상의 단계가 실패하여 무시되었음을 알려주는 정보가 포함되어 있습니다.

- maxAttempts – 모든 재시도에 대해 전체 단계와 모든 반복이 처음부터 실행됩니다.
- status – `status`단계 실행의 전체 상태에서 개별 반복의 상태를 나타내지는 않습니다. 루프가 있는 단계의 상태는 다음과 같이 결정합니다.
  - 단일 반복 실행에 실패할 경우, 단계 상태는 실패로 표시됩니다.
  - 모든 반복이 성공하면, 단계 상태가 성공으로 표시됩니다.
- startTime - 단계 실행의 전체 시작 시간입니다. 개별 반복의 시작 시간을 나타내지 않습니다.
- endTime - 단계 실행의 전체 종료 시간입니다. 개별 반복의 종료 시간을 나타내지 않습니다.

- failureMessage - 제한 시간이 초과되지 않은 오류가 발생한 경우 실패한 반복 인덱스를 포함합니다. 시간 초과 오류가 발생한 경우, 루프 실행이 실패했다는 메시지가 표시됩니다. 실패 메시지의 크기를 최소화하기 위해 각 반복에 대한 개별 오류 메시지는 제공되지 않습니다.

단계별 및 반복 출력

모든 반복에는 출력값이 포함됩니다. 루프 실행이 끝나면는 성공한 모든 반복 출력을에 AWSTOE 통합합니다`detailedOutput.json`. 통합 출력은 작업 모듈의 출력 스키마에 정의된 해당 출력 키에 속하는 값을 정렬한 것입니다. 다음 예에서는 출력이 통합되는 방법을 보여줍니다.

반복 1에 **`ExecuteBash`**에 대한 출력

```
{
 "stdout":"Hello"
}
```

반복 2에 **`ExecuteBash`**에 대한 출력

```
{
 "stdout":"World"
}
```

단계에 **`ExecuteBash`**에 대한 출력

```
{
 "stdout":"Hello\nWorld"
}
```

예를 들어, `ExecuteBash`, `ExecutePowerShell`, `ExecuteBinary`에서 작업 모듈 출력으로 `STDOUT`(을)를 반환하는 작업 모듈입니다. `STDOUT` 메시지는 새 줄 문자와 결합되어 `detailedOutput.json` 단계의 전체 출력을 생성합니다.

AWSTOE 는 실패한 반복의 출력을 통합하지 않습니다.

AWSTOE 구성 요소 관리자가 지원하는 작업 모듈

EC2 Image Builder와 같은 이미지 구축 서비스는 AWSTOE 작업 모듈을 사용하여 사용자 지정 머신 이미지를 구축하고 테스트하는 데 사용되는 EC2 인스턴스를 구성하는 데 도움이 됩니다. 이 섹션에서는 일반적으로 사용되는 AWSTOE 작업 모듈의 기능과 예제를 포함하여 이를 구성하는 방법을 설명합니다.

구성 요소는 일반 텍스트 YAML 문서를 사용하여 작성됩니다. 문서 구문에 대한 자세한 내용은 사용자 지정 AWSTOE 구성 요소에 구성 요소 문서 프레임워크 사용 섹션을 참조하세요.

> **Note**
>
> 모든 작업 모듈은 실행할 때 Systems Manager 에이전트와 동일한 계정을 사용하며, Linux의 경우 `root`이고 Windows의 경우 `NT Authority\SYSTEM`입니다.

다음 상호 참조는 수행하는 작업 유형별로 작업 모듈을 분류합니다.

일반 실행

- Assert(Linux, Windows, macOS)
- ExecuteBash(Linux, macOS)
- ExecuteBinary(Linux, Windows, macOS)
- ExecuteDocument(Linux, Windows, macOS)
- ExecutePowerShell(Windows)

파일 다운로드 및 업로드

- S3Download(Linux, Windows, macOS)
- S3Upload(Linux, Windows, macOS)
- WebDownload(Linux, Windows, macOS)

파일 시스템 작업

- AppendFile(Linux, Windows, macOS)
- CopyFile(Linux, Windows, macOS)
- CopyFolder(Linux, Windows, macOS)
- CreateFile(Linux, Windows, macOS)

- CreateFolder(Linux, Windows, macOS)
- CreateSymlink(Linux, Windows, macOS)
- DeleteFile(Linux, Windows, macOS)
- DeleteFolder(Linux, Windows, macOS)
- ListFiles(Linux, Windows, macOS)
- MoveFile(Linux, Windows, macOS)
- MoveFolder(Linux, Windows, macOS)
- ReadFile(Linux, Windows, macOS)
- SetFileEncoding(Linux, Windows, macOS)
- SetFileOwner(Linux, Windows, macOS)
- SetFolderOwner(Linux, Windows, macOS)
- SetFilePermissions(Linux, Windows, macOS)
- SetFolderPermissions(Linux, Windows, macOS)

소프트웨어 설치 작업

- InstallMSI(Windows)
- UninstallMSI(Windows)

시스템 작업

- Reboot(Linux, Windows)
- SetRegistry(Windows)
- UpdateOS(Linux, Windows)

일반 실행 모듈

다음 섹션에는 명령을 실행하고 실행 워크플로를 제어하는 작업 모듈에 대한 세부 정보가 포함되어 있습니다.

일반 실행 작업 모듈

- Assert(Linux, Windows, macOS)
- ExecuteBash(Linux, macOS)
- ExecuteBinary(Linux, Windows, macOS)
- ExecuteDocument(Linux, Windows, macOS)
- ExecutePowerShell(Windows)

Assert(Linux, Windows, macOS)

Assert 작업 모듈은 비교 연산자 또는 논리 연산자를 입력으로 사용하여 값 비교를 수행합니다. 연산자 표현식(true 또는 false)의 결과는 단계의 전체 성공 또는 실패 상태를 나타냅니다.

비교 또는 논리 연산자 표현식이 `true`로 평가되면 단계가 `Success`로 표시됩니다. 그렇지 않으면 단계가 `Failed`로 표시됩니다. 단계가 실패하면 `onFailure` 파라미터가 단계의 결과를 결정합니다.

입력

키 이름	설명	형식	필수
`input`	단일 비교 또는 논리 연산자를 포함합니다. 참고로 논리 연산자는 둘 이상의 비교 연산자를 포함할 수 있습니다.	이는 연산자에 따라 가변적입니다.	예

입력 예제: **`stringEquals`** 비교 연산자를 사용한 간단한 비교

이 예제는 `true`로 평가됩니다.

```
- name: StringComparison
  action: Assert
  inputs:
    stringEquals: '2.1.1'
    value: '{{ validate.ApplicationVersion.outputs.stdout }}'
```

입력 예제: **`patternMatches`** 비교 연산자를 사용한 Regex 비교

이 예제는 모두 true로 평가됩니다.

```
- name: Letters only
  action: Assert
  inputs:
    patternMatches: '^[a-zA-Z]+$'
    value: 'ThisIsOnlyLetters'

- name: Letters and spaces only
  action: Assert
  inputs:
    patternMatches: '^[a-zA-Z\s]+$'
    value: 'This text contains spaces'

- name: Numbers only
  action: Assert
  inputs:
    patternMatches: '^[0-9]+$'
    value: '1234567890'
```

입력 예제: 논리 연산자 및 체인 변수를 사용한 중첩 비교

다음 예제에서는 체인 변수와의 비교를 사용하는 논리 연산자와의 중첩 비교를 보여줍니다. 다음 중 하나가 true인 경우 Assert는 true로 평가됩니다.

- ApplicationVersion은 2.0보다 크고 CPUArchitecture는 arm64와 같습니다.
- CPUArchitecture는 x86_64와 같습니다.

```
- name: NestedComparisons
  action: Assert
  inputs:
    or: # <- first level deep
      - and: # <- second level deep
          - numberGreaterThan: 2.0 # <- third level deep
            value: '{{ validate.ApplicationVersion.outputs.stdout }}'
          - stringEquals: 'arm64'
            value: '{{ validate.CPUArchitecture.outputs.stdout }}'
      - stringEquals: 'x86_64'
        value: '{{ validate.CPUArchitecture.outputs.stdout }}'
```

출력:

Assert의 출력은 단계의 성공 또는 실패입니다.

ExecuteBash(Linux, macOS)

ExecuteBash 작업 모듈을 사용하면 인라인 쉘 코드/명령을 사용하여 bash 스크립트를 실행할 수 있습니다. 이 모듈은 Linux를 지원합니다.

명령 블록에서 지정하는 모든 명령과 지침은 파일(예: input.sh)로 변환되고 bash 쉘과 함께 실행됩니다. 쉘 파일을 실행한 결과는 해당 단계의 종료 코드입니다.

ExecuteBash 모듈은 스크립트가 종료 코드 194(으)로 종료되는 경우 시스템 재시작을 처리합니다. 애플리케이션이 시작되면 다음 작업 중 하나를 수행합니다.

- Systems Manager Agent에서 실행하는 경우, 애플리케이션은 종료 코드를 호출자에게 전달합니다. Systems Manager Agent는 스크립트에서 관리형 인스턴스 재부팅에 설명된 대로 시스템 재부팅을 처리하고 재시작을 시작한 것과 동일한 단계를 실행합니다.
- 애플리케이션은 현재 executionstate(을)를 저장하고 애플리케이션을 다시 실행하도록 재시작 트리거를 구성한 다음 시스템을 다시 시작합니다.

시스템을 다시 시작한 후 애플리케이션은 재시작을 시작한 단계와 동일한 단계를 실행합니다. 이 기능이 필요한 경우 동일한 쉘 명령의 여러 호출을 처리할 수 있는 idempotent 스크립트를 작성해야 합니다.

입력

키 이름	설명	형식	필수
commands	bash 구문에 따라 실행할 지침 또는 명령 목록이 포함되어 있습니다. 여러 줄의 YAML이 허용됩니다.	나열	예

입력 예제: 재부팅 전과 재부팅 후

```
name: ExitCode194Example
description: This shows how the exit code can be used to restart a system with
 ExecuteBash
schemaVersion: 1.0
```

```
phases:
  - name: build
    steps:
      - name: RestartTrigger
        action: ExecuteBash
        inputs:
          commands:
            - |
              REBOOT_INDICATOR=/var/tmp/reboot-indicator
              if [ -f "${REBOOT_INDICATOR}" ]; then
                echo 'The reboot file exists. Deleting it and exiting with success.'
                rm "${REBOOT_INDICATOR}"
                exit 0
              fi
              echo 'The reboot file does not exist. Creating it and triggering a
 restart.'
              touch "${REBOOT_INDICATOR}"
              exit 194
```

출력

필드	설명	형식
stdout	명령 실행의 표준 출력.	문자열

재부팅을 시작하고 작업 모듈의 일부로 종료 코드 194(을)를 반환하면 재부팅을 시작한 동일한 작업 모듈 단계에서 빌드가 재개됩니다. 종료 코드 없이 재부팅을 시작하면 빌드 프로세스가 실패할 수 있습니다.

출력 예제: 재부팅 전(문서를 통해 처음으로)

```
{
 “stdout”: “The reboot file does not exist. Creating it and triggering a restart."
}
```

출력 예제: 재부팅 후(문서를 통해 두 번째로)

```
{
 “stdout”: “The reboot file exists. Deleting it and exiting with success."
}
```

ExecuteBinary(Linux, Windows, macOS)

ExecuteBinary 작업 모듈을 사용하면 명령줄 인수 목록을 사용하여 바이너리 파일을 실행할 수 있습니다.

ExecuteBinary 모듈은 바이너리 파일이 종료 코드 194(Linux) 또는 3010(Windows)으로 종료될 경우 시스템 재시작을 처리합니다. 이 경우 애플리케이션은 다음 작업 중 하나를 수행합니다.

- Systems Manager Agent에서 실행하는 경우, 애플리케이션은 종료 코드를 호출자에게 전달합니다. Systems Manager Agent는 스크립트에서 관리형 인스턴스 재부팅에 설명된 대로 시스템 재시작을 처리하고 재시작을 시작한 것과 동일한 단계를 실행합니다.
- 애플리케이션은 현재 executionstate(을)를 저장하고 애플리케이션을 다시 실행하도록 재시작 트리거를 구성한 다음 시스템을 다시 시작합니다.

시스템이 재시작된 후 애플리케이션은 재시작을 시작한 단계와 동일한 단계를 실행합니다. 이 기능이 필요한 경우 동일한 쉘 명령의 여러 호출을 처리할 수 있는 idempotent 스크립트를 작성해야 합니다.

입력

키 이름	설명	형식	필수
path	실행할 바이너리 파일 경로입니다.	String	예
arguments	바이너리를 실행할 때 사용할 명령줄 인수 목록이 포함되어 있습니다.	문자열 목록	아니요

입력 예제: .NET 설치

```
- name: "InstallDotnet"
  action: ExecuteBinary
  inputs:
    path: C:\PathTo\dotnet_installer.exe
    arguments:
      - /qb
      - /norestart
```

출력

필드	설명	형식
stdout	명령 실행의 표준 출력.	문자열

출력 예제

```
{
 "stdout": "success"
}
```

ExecuteDocument(Linux, Windows, macOS)

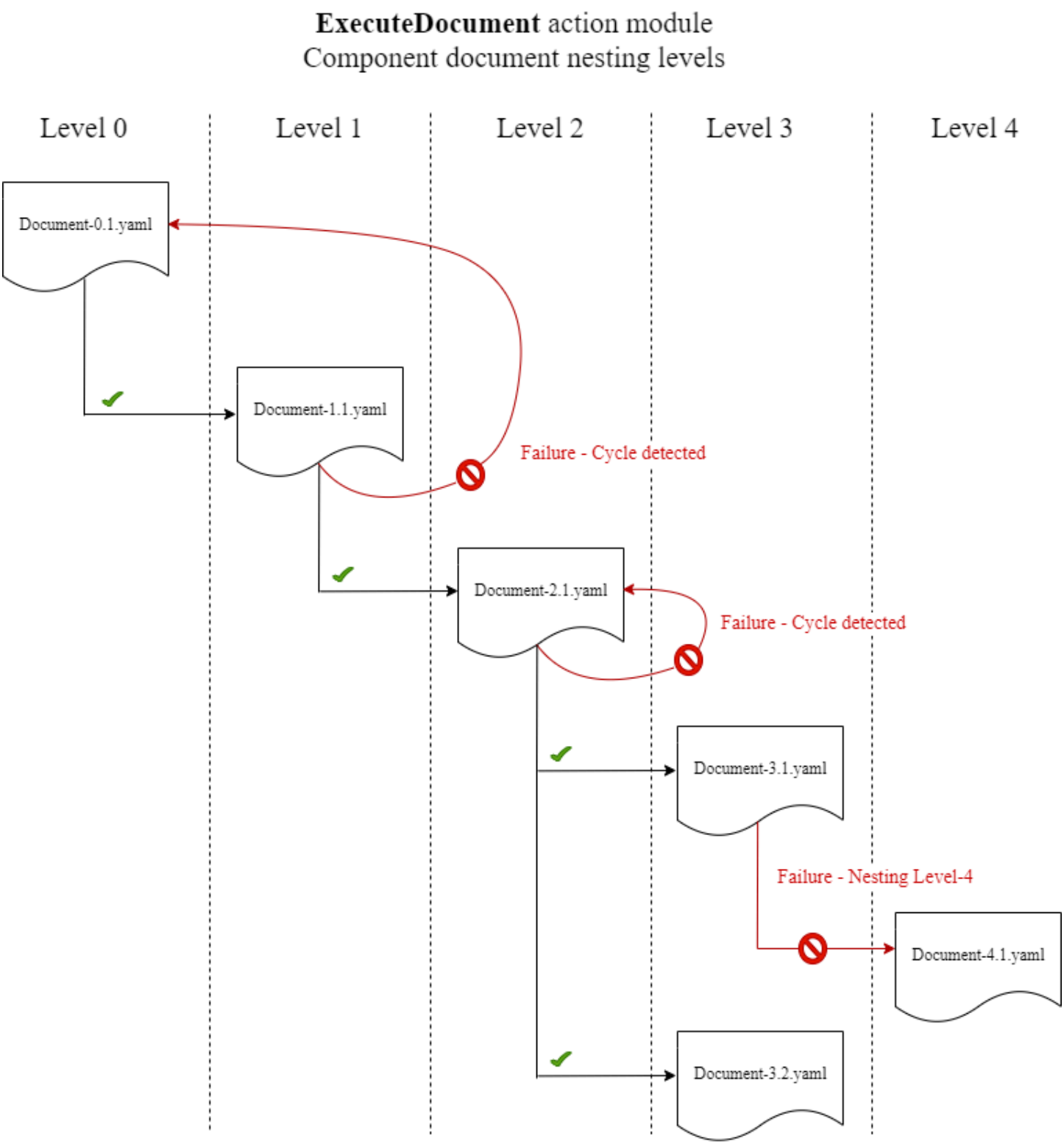
ExecuteDocument 작업 모듈은 한 문서에서 여러 구성 요소 문서를 실행하는 중첩된 구성 요소 문서에 대한 지원을 추가합니다. AWSTOE 는 런타임 시 입력 파라미터로 전달된 문서의 유효성을 검사합니다.

제한 사항

- 이 작업 모듈은 한 번만 실행되고 재시도는 허용되지 않으며 제한 시간을 설정하는 옵션도 없습니다. ExecuteDocument는 다음과 같은 기본값을 설정하며, 기본값을 변경하려고 하면 오류를 반환합니다.
  - `timeoutSeconds`: -1
  - `maxAttempts`: 1

  > **Note**
  >
  > 이러한 값은 비워 둘 수 있으며 기본값을 AWSTOE 사용합니다.

- 문서 중첩은 최대 세 개 수준까지 허용되지만, 그 이상은 허용되지 않습니다. 최상위 수준은 중첩되지 않으므로 세 개 수준의 중첩은 네 가지 문서 수준으로 변환됩니다. 이 시나리오에서는 최하위 수준 문서가 다른 문서를 호출해서는 안 됩니다.
- 구성 요소 문서의 순환 실행은 허용되지 않습니다. 반복 구성 외부에서 자신을 호출하거나 현재 실행 체인의 상위에 있는 다른 문서를 호출하는 모든 문서는 사이클을 시작하여 무한 루프를 초래할 수 있습니다. AWSTOE 가 순환 실행을 감지하면 실행을 중지하고 실패를 기록합니다.



구성 요소 문서가 자체적으로 실행되거나 현재 실행 체인의 상위에 있는 구성 요소 문서를 실행하려고 하면 실행되지 않습니다.

입력

키 이름	설명	형식	필수
document	구성 요소 문서의 경로. 유효한 옵션은 다음과 같습니다. • 로컬 파일 경로 • S3 URI • EC2 Image Builder 구성 요소 빌드 버전 ARN	String	예
document-s3-bucket-owner	구성 요소 문서가 저장된 S3 버킷의 S3 버킷 소유자의 계정 ID입니다. (구성 요소 문서에서 S3 URI를 사용하는 경우 권장합니다.)	String	No
phases	구성 요소 문서에서 실행할 수 있는 단계로, 쉼표로 구분된 목록으로 표시됩니다. 단계가 지정되지 않은 경우 모든 단계가 실행됩니다.	String	No
parameters	런타임 시 구성 요소 문서에 키 값 페어로 전달되는 입력 파라미터입니다.	파라미터 맵 목록	아니요

파라미터 맵 입력

키 이름	설명	형식	필수
name	ExecuteDocument 작업 모듈이 실행 중인 구성 요소 문서에 전달할 입력 파라미터의 이름입니다.	String	예
value	입력 파라미터의 값입니다.	String	예

입력 예제

다음 예제는 설치 경로에 따른 구성 요소 문서 입력의 변형을 보여줍니다.

입력 예제: 로컬 문서 경로

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        inputs:
          document: Sample-1.yaml
          phases: build
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

입력 예제: 문서 경로로서의 S3 URI

```
# main.yaml
schemaVersion: 1.0
```

```
phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        inputs:
          document: s3://my-bucket/Sample-1.yaml
          document-s3-bucket-owner: 123456789012
          phases: build,validate
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

입력 예제: 문서 경로로서의 EC2 Image Builder 구성 요소 ARN

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        inputs:
          document: arn:aws:imagebuilder:us-west-2:aws:component/Sample-Test/1.0.0
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

ForEach 루프를 사용하여 문서 실행

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
```

```
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        loop:
          name: 'myForEachLoop'
          forEach:
            - Sample-1.yaml
            - Sample-2.yaml
        inputs:
          document: "{{myForEachLoop.value}}"
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

For 루프를 사용하여 문서 실행

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        loop:
          name: 'myForLoop'
          for:
            start: 1
            end: 2
            updateBy: 1
        inputs:
          document: "Sample-{{myForLoop.value}}.yaml"
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

출력

AWSTOE 는 실행될 detailedoutput.json 때마다 라는 출력 파일을 생성합니다. 이 파일에는 실행 중에 호출되는 모든 구성 요소 문서의 모든 단계에 대한 세부 정보가 포함됩니다. ExecuteDocument 작업 모듈의 경우 outputs 필드에서 간략한 런타임 요약과 detailedOutput에서 실행되는 단계(phases), 단계(steps) 및 문서에 대한 세부 정보를 찾을 수 있습니다.

```
{
 \"executedStepCount\":1,\"executionId\":\"97054e22-06cc-11ec-9b14-acde48001122\",
\"failedStepCount\":0,\"failureMessage\":\"\",\"ignoredFailedStepCount\":0,\"logUrl\":
\"\",\"status\":\"success\"
}",
```

각 구성 요소 문서의 출력 요약 개체에는 다음과 같은 세부 정보가 여기 나온 것처럼 샘플 값과 함께 포함되어 있습니다.

- executedStepCount":1
- "executionId":"12345a67-89bc-01de-2f34-abcd56789012"
- "failedStepCount":0
- "failureMessage":""
- "ignoredFailedStepCount":0
- "logUrl":""
- "status":"success"

출력 예시

다음 예제에서는 중첩 실행이 발생할 때 ExecuteDocument 작업 모듈의 출력을 보여 줍니다. 이 예제에서 main.yaml 구성 요소 문서는 Sample-1.yaml 구성 요소 문서를 성공적으로 실행합니다.

```
{
    "executionId": "12345a67-89bc-01de-2f34-abcd56789012",
    "status": "success",
    "startTime": "2021-08-26T17:20:31-07:00",
    "endTime": "2021-08-26T17:20:31-07:00",
    "failureMessage": "",
    "documents": [
        {
            "name": "",
            "filePath": "main.yaml",
            "status": "success",
            "description": "",
```

```
            "startTime": "2021-08-26T17:20:31-07:00",
            "endTime": "2021-08-26T17:20:31-07:00",
            "failureMessage": "",
            "phases": [
                {
                    "name": "build",
                    "status": "success",
                    "startTime": "2021-08-26T17:20:31-07:00",
                    "endTime": "2021-08-26T17:20:31-07:00",
                    "failureMessage": "",
                    "steps": [
                        {
                            "name": "ExecuteNestedDocument",
                            "status": "success",
                            "failureMessage": "",
                            "timeoutSeconds": -1,
                            "onFailure": "Abort",
                            "maxAttempts": 1,
                            "action": "ExecuteDocument",
                            "startTime": "2021-08-26T17:20:31-07:00",
                            "endTime": "2021-08-26T17:20:31-07:00",
                            "inputs": "[{\"document\":\"Sample-1.yaml\",\"document-s3-
bucket-owner\":\"\",\"phases\":\"\",\"parameters\":null}]",
                            "outputs": "[{\"executedStepCount\":1,\"executionId\":
\"98765f43-21ed-09cb-8a76-fedc54321098\",\"failedStepCount\":0,\"failureMessage\":\"\",
\"ignoredFailedStepCount\":0,\"logUrl\":\"\",\"status\":\"success\"}]",
                            "loop": null,
                            "detailedOutput": [
                                {
                                    "executionId": "98765f43-21ed-09cb-8a76-
fedc54321098",
                                    "status": "success",
                                    "startTime": "2021-08-26T17:20:31-07:00",
                                    "endTime": "2021-08-26T17:20:31-07:00",
                                    "failureMessage": "",
                                    "documents": [
                                        {
                                            "name": "",
                                            "filePath": "Sample-1.yaml",
                                            "status": "success",
                                            "description": "",
                                            "startTime": "2021-08-26T17:20:31-07:00",
                                            "endTime": "2021-08-26T17:20:31-07:00",
                                            "failureMessage": "",
```

```
                                            "phases": [
                                                {
                                                    "name": "build",
                                                    "status": "success",
                                                    "startTime":
 "2021-08-26T17:20:31-07:00",
                                                    "endTime":
 "2021-08-26T17:20:31-07:00",
                                                    "failureMessage": "",
                                                    "steps": [
                                                        {
                                                            "name": "ExecuteBashStep",
                                                            "status": "success",
                                                            "failureMessage": "",
                                                            "timeoutSeconds": 7200,
                                                            "onFailure": "Abort",
                                                            "maxAttempts": 1,
                                                            "action": "ExecuteBash",
                                                            "startTime":
 "2021-08-26T17:20:31-07:00",
                                                            "endTime":
 "2021-08-26T17:20:31-07:00",
                                                            "inputs": "[{\"commands\":
[\"echo \\\"Hello World!\\\"\"]}]",
                                                            "outputs": "[{\"stdout\":
\"Hello World!\"}]",
                                                            "loop": null,
                                                            "detailedOutput": null
                                                        }]
                                                }]
                                        }]
                                }]
                        }]

                }]
        }]
}
```

ExecutePowerShell(Windows)

ExecutePowerShell 작업 모듈을 사용하면 인라인 쉘 코드/명령을 사용하여 PowerShell 스크립트를 실행할 수 있습니다. 이 모듈은 Windows 플랫폼과 Windows PowerShell을 지원합니다.

명령 블록에 지정된 모든 명령/지침은 스크립트 파일(예: `input.ps1`)로 변환되며 Windows PowerShell을 사용하여 실행됩니다. 쉘 파일을 실행한 결과는 종료 코드입니다.

ExecutePowerShell 모듈은 쉘 명령이 종료 코드 `3010`(으)로 종료된 경우, 시스템 재시작을 처리합니다. 애플리케이션이 시작되면 다음 작업 중 하나를 수행합니다.

- Systems Manager 에이전트가 실행하는 경우 종료 코드를 호출자에게 전달합니다. Systems Manager Agent는 스크립트에서 관리형 인스턴스 재부팅에 설명된 대로 시스템 재부팅을 처리하고 재시작을 시작한 것과 동일한 단계를 실행합니다.
- 현재 `executionstate`(을)를 저장하고 애플리케이션을 다시 실행하도록 재시작 트리거를 구성한 다음 시스템을 재부팅합니다.

시스템을 다시 시작한 후 애플리케이션은 재시작을 시작한 단계와 동일한 단계를 실행합니다. 이 기능이 필요한 경우 동일한 쉘 명령의 여러 호출을 처리할 수 있는 idempotent 스크립트를 작성해야 합니다.

입력

키 이름	설명	형식	필수
`commands`	PowerShell 구문에 따라 실행할 지침 또는 명령 목록이 포함됩니다. 여러 줄의 YAML이 허용됩니다.	문자열 목록	예. `commands` 또는 `file`(을)를 지정해야 하며, 둘 다 지정하지는 않습니다.
`file`	PowerShell 스크립트 파일에 대한 경로를 포함합니다. PowerShell은 `-file` 명령줄 인수를 사용하여 이 파일에 대해 실행됩니다. 이 경로는 `.ps1` 파일을 가리켜야 합니다.	String	예. `commands` 또는 `file`(을)를 지정해야 하며, 둘 다 지정하지는 않습니다.

입력 예제: 재부팅 전과 재부팅 후

```
name: ExitCode3010Example
```

```
description: This shows how the exit code can be used to restart a system with
 ExecutePowerShell
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: RestartTrigger
        action: ExecutePowerShell
        inputs:
          commands:
            - |
              $rebootIndicator = Join-Path -Path $env:SystemDrive -ChildPath 'reboot-
indicator'
              if (Test-Path -Path $rebootIndicator) {
                Write-Host 'The reboot file exists. Deleting it and exiting with
 success.'
                Remove-Item -Path $rebootIndicator -Force | Out-Null
                [System.Environment]::Exit(0)
              }
              Write-Host 'The reboot file does not exist. Creating it and triggering a
 restart.'
              New-Item -Path $rebootIndicator -ItemType File | Out-Null
              [System.Environment]::Exit(3010)
```

출력

필드	설명	형식
stdout	명령 실행의 표준 출력.	문자열

재부팅을 실행하고 작업 모듈의 일부로 종료 코드 3010(을)를 반환하면, 재부팅을 시작한 동일한 작업 모듈 단계에서 빌드가 재개됩니다. 종료 코드 없이 재부팅을 실행하면, 빌드 프로세스가 실패할 수 있습니다.

출력 예제: 재부팅 전(문서를 통해 처음으로)

```
{
 “stdout”: “The reboot file does not exist. Creating it and triggering a restart."
}
```

출력 예제: 재부팅 후(문서를 통해 두 번째로)

```
{
 “stdout”: “The reboot file exists. Deleting it and exiting with success."
}
```

파일 다운로드 및 업로드 모듈

다음 섹션에는 파일을 업로드하거나 다운로드하는 작업 모듈에 대한 세부 정보가 포함되어 있습니다.

작업 모듈 다운로드 및 업로드

- S3Download(Linux, Windows, macOS)
- S3Upload(Linux, Windows, macOS)
- WebDownload(Linux, Windows, macOS)

S3Download(Linux, Windows, macOS)

`S3Download` 작업 모듈을 사용하면, Amazon S3 객체 또는 객체 집합을 `destination` 경로로 지정한 로컬 파일 또는 폴더에 다운로드할 수 있습니다. 지정된 위치에 파일이 이미 있고 `overwrite` 플래그가 true로 설정된 경우, `S3Download`(이)가 파일을 덮어씁니다.

`source` 위치는 Amazon S3의 특정 객체를 가리키거나 별표 와일드카드(*)와 함께 키 접두사를 사용하여 키 접두사 경로와 일치하는 객체 집합을 다운로드할 수 있습니다. `source` 위치에서 키 접두사를 지정하면 `S3Download` 작업 모듈이 접두사와 일치하는 모든 항목(파일 및 폴더 포함)을 다운로드합니다. 접두사와 일치하는 모든 항목을 다운로드할 수 있도록 키 접두사가 슬래시 뒤에 별표(/*)로 끝나는지 확인합니다. 예를 들어 *`s3://my-bucket/my-folder/*`*입니다.

다운로드 중에 지정된 키 접두사에 대한 `S3Download` 작업이 실패하는 경우, 폴더 콘텐츠는 실패 이전 상태로 롤백되지 않습니다. 대상 폴더는 장애 발생 시점의 상태로 유지됩니다.

지원되는 사용 사례

`S3Download` 작업 모듈은 다음 사용 사례를 지원합니다.

- Amazon S3 객체는 다운로드 경로에 지정된 대로 로컬 폴더에 다운로드됩니다.
- Amazon S3 파일 경로에 키 접두사가 있는 Amazon S3 객체는 지정된 로컬 폴더로 다운로드되며, 이 폴더는 키 접두사와 일치하는 모든 Amazon S3 객체를 로컬 폴더에 반복적으로 복사합니다.

IAM 요구 사항

인스턴스 프로파일에 연결하는 IAM 역할에는 S3Download 작업 모듈을 실행할 권한이 있어야 합니다. 다음 IAM 정책을 인스턴스 프로파일과 관련된 IAM 역할에 연결해야 합니다.

- 단일 파일: 버킷/객체에 대한 `s3:GetObject`(예: `arn:aws:s3:::BucketName/*`)
- 다중 파일: 버킷/객체에 대한 `s3:ListBucket`(예: `arn:aws:s3:::BucketName`) 및 버킷/객체에 대한 `s3:GetObject`(예: `arn:aws:s3:::BucketName/*`)

입력

키	설명	형식	필수	기본값
`source`	다운로드의 소스가 되는 Amazon S3 버킷입니다. 특정 객체의 경로를 지정하거나, 슬래시 다음 별표 와일드카드(/*)로 끝나는 키 접두사를 사용하여 키 접두사와 일치하는 객체 집합을 다운로드할 수 있습니다.	String	예	N/A
`destination`	Amazon S3 객체가 다운로드되는 로컬 경로입니다. 단일 파일을 다운로드하려면, 파일 이름을 경로의 일부로 지정해야 합니다. 예를 들어 */myfolder*	String	예	N/A

키	설명	형식	필수	기본값
	/package.zip 입니다.			
expectedBucketOwner	source 경로에 제공된 버킷의 예상 소유자 계정 ID입니다. 원본에 지정된 Amazon S3 버킷의 소유권을 확인하는 것이 좋습니다.	String	No	N/A

키	설명	형식	필수	기본값
overwrite	true로 설정하면 지정된 로컬 경로의 대상 폴더에 같은 이름의 파일이 이미 있는 경우 다운로드 파일이 로컬 파일을 덮어씁니다. false로 설정하면 로컬 시스템의 기존 파일이 덮어써지지 않고, 작업 모듈이 실패하며 다운로드 오류가 발생합니다. 예: `Error: S3Download: File already exists and "overwrite" property for "destination" file is set to false. Cannot download.`	불린(Boolean)	아니요	true

> **Note**
>
> 다음 예제에서는 Windows 폴더 경로를 Linux 경로로 바꿀 수 있습니다. 예를 들어, *C:\myfolder\package.zip*(을)를 */myfolder/package.zip*(으)로 바꿀 수 있습니다.

입력 예제: Amazon S3 객체를 로컬 파일에 복사

다음 예제에서는 Amazon S3 객체를 로컬 파일에 복사하는 방법을 보여 줍니다.

```
- name: DownloadMyFile
  action: S3Download
  inputs:
    - source: s3://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: true
    - source: s3://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
```

입력 예제: 키 접두사가 있는 Amazon S3 버킷의 모든 Amazon S3 객체를 로컬 폴더에 복사

다음 예제에서는 키 접두사가 있는 Amazon S3 버킷의 모든 Amazon S3 객체를 로컬 폴더로 복사하는 방법을 보여 줍니다. Amazon S3에는 폴더라는 개념이 없으므로, 키 접두사와 일치하는 모든 객체가 복사됩니다. 다운로드할 수 있는 최대 객체 수는 1,000개입니다.

```
- name: MyS3DownloadKeyprefix
  action: S3Download
  maxAttempts: 3
  inputs:
    - source: s3://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
```

```
        expectedBucketOwner: 123456789022
        overwrite: true
      - source: s3://amzn-s3-demo-source-bucket/path/to/*
        destination: C:\myfolder\
        expectedBucketOwner: 123456789022
```

출력

없음.

S3Upload(Linux, Windows, macOS)

S3Upload 작업 모듈을 사용하면 원본 파일 또는 폴더에서 Amazon S3 위치로 파일을 업로드할 수 있습니다. 소스 위치에 지정된 경로에 와일드카드(*)를 사용하여 경로가 와일드카드 패턴과 일치하는 모든 파일을 업로드할 수 있습니다.

재귀 S3Upload 작업이 실패하는 경우 이미 업로드된 모든 파일은 대상 Amazon S3 버킷에 남아 있게 됩니다.

지원되는 사용 사례

- 로컬 파일을 Amazon S3 객체로.
- 폴더의 로컬 파일(와일드카드 포함)을 Amazon S3 키 접두사로.
- 로컬 폴더(`recurse`를 `true`로 설정해야 함)를 Amazon S3 키 접두사로 복사합니다.

IAM 요구 사항

인스턴스 프로파일에 연결하는 IAM 역할에는 `S3Upload` 작업 모듈을 실행할 권한이 있어야 합니다. 다음 IAM 정책을 인스턴스 프로파일과 관련된 IAM 역할에 연결해야 합니다. 정책은 대상 Amazon S3 버킷에 `s3:PutObject` 권한을 부여해야 합니다. 예: `arn:aws:s3:::BucketName/*`).

입력

키	설명	형식	필수	기본값
`source`	소스 파일/폴더가 시작되는 로컬 경로. `source`(은)는 별표 와일드카	String	예	N/A

키	설명	형식	필수	기본값
	드(*)(을)를 지원합니다.			
destination	원본 파일/폴더가 업로드되는 대상 Amazon S3 버킷의 경로입니다.	String	예	N/A
recurse	true(으)로 설정하면 S3Upload(을)를 재귀적으로 수행합니다.	String	No	false
expectedBucketOwner	대상 경로에 지정된 Amazon S3 버킷의 예상 소유자 계정 ID입니다. 대상에 지정된 Amazon S3 버킷의 소유권을 확인하는 것이 좋습니다.	String	No	N/A

입력 예제: 로컬 파일을 Amazon S3 객체에 복사

다음 예제에서는 로컬 파일을 Amazon S3 객체에 복사하는 방법을 보여 줍니다.

```
- name: MyS3UploadFile
  action: S3Upload
  onFailure: Abort
  maxAttempts: 3
  inputs:
    - source: C:\myfolder\package.zip
```

```
        destination: s3://amzn-s3-demo-destination-bucket/path/to/package.zip
        expectedBucketOwner: 123456789022
```

입력 예제: 키 접두사를 사용하여 로컬 폴더의 모든 파일을 Amazon S3 버킷으로 복사

다음 예제에서는 키 접두사를 사용하여 로컬 폴더의 모든 파일을 Amazon S3 버킷으로 복사하는 방법을 보여 줍니다. `recurse`(이)가 지정되지 않았으므로 이 예제는 하위 폴더나 해당 콘텐츠를 복사하지 않으며 기본값은 `false`입니다.

```
- name: MyS3UploadMultipleFiles
  action: S3Upload
  onFailure: Abort
  maxAttempts: 3
  inputs:
    - source: C:\myfolder\*
      destination: s3://amzn-s3-demo-destination-bucket/path/to/
      expectedBucketOwner: 123456789022
```

입력 예제: 모든 파일 및 폴더를 로컬 폴더에서 Amazon S3 버킷으로 재귀적으로 복사

다음 예제에서는 모든 파일 및 폴더를 로컬 폴더에서 키 접두사를 사용하여 Amazon S3 버킷으로 재귀적으로 복사하는 방법을 보여줍니다.

```
- name: MyS3UploadFolder
  action: S3Upload
  onFailure: Abort
  maxAttempts: 3
  inputs:
    - source: C:\myfolder\*
      destination: s3://amzn-s3-demo-destination-bucket/path/to/
      recurse: true
      expectedBucketOwner: 123456789022
```

출력

없음.

WebDownload(Linux, Windows, macOS)

WebDownload 작업 모듈을 사용하면 HTTP/HTTPS 프로토콜을 통해 원격 위치에서 파일 및 리소스를 다운로드할 수 있습니다(HTTPS 권장). 다운로드 수나 크기에는 제한이 없습니다. 이 모듈은 재시도 및 지수 백오프 로직을 처리합니다.

사용자 입력에 따라 각 다운로드 작업의 성공 시도는 최대 5회까지 할당됩니다. 이러한 시도는 작업 모듈 오류와 관련된 문서 `steps`의 `maxAttempts` 필드에 지정된 시도와 다릅니다.

이 작업 모듈은 리디렉션을 묵시적으로 처리합니다. `200`을(를) 제외한 모든 HTTP 상태 코드에서 오류가 발생합니다.

입력

키 이름	설명	형식	필수	기본값
`source`	RFC 3986 표준을 따르는 유효한 HTTP/HTTPS URL(HTTPS 권장)입니다. chaining 표현식은 허용됩니다.	String	예	N/A
`destination`	로컬 시스템의 절대 또는 상대 파일이나 폴더 경로입니다. 폴더 경로는 /(으)로 끝나야 합니다. /(으)로 끝나지 않는 경우, 파일 경로로 취급됩니다. 모듈은 성공적인 다운로드를 위해 필요한 파일이나 폴더를 생성합니다. chaining 표현식은 허용됩니다.	String	예	N/A
`overwrite`	활성화하면 로컬 시스템의 기존 파일을 다운로드한 파일 또는 리소스	불린(Boolean)	아니요	`true`

키 이름	설명	형식	필수	기본값
	로 덮어씁니다. 활성화하지 않으면 로컬 시스템의 기존 파일을 덮어쓰지 않고, 작업 모듈이 실패하며 오류가 발생합니다. 덮어쓰기가 활성화되고 체크섬과 알고리즘이 지정된 경우, 작업 모듈은 기존 파일의 체크섬과 해시가 일치하지 않는 경우에만 파일을 다운로드합니다.			
checksum	체크섬을 지정하면, 제공된 알고리즘으로 생성된 다운로드한 파일의 해시와 비교하여 체크섬이 검사됩니다. 파일 검증을 활성화하려면, 체크섬과 알고리즘을 모두 제공해야 합니다. chaining 표현식은 허용됩니다.	String	No	N/A

키 이름	설명	형식	필수	기본값
algorithm	체크섬을 계산하는 데 사용되는 알고리즘입니다. 옵션은 MD5, SHA1, SHA256 및 SHA512입니다. 파일 검증을 활성화하려면, 체크섬과 알고리즘을 모두 제공해야 합니다. chaining 표현식은 허용됩니다.	String	No	N/A
ignoreCertificateErrors	활성화된 경우 SSL 인증서 검증이 무시됩니다.	불린(Boolean)	아니요	false

출력

키 이름	설명	형식
destination	다운로드한 파일 또는 리소스가 저장되는 대상 경로를 지정하는 줄 바꿈 문자로 구분된 문자열입니다.	String

입력 예제: 로컬 대상으로 원격 파일 다운로드

```
  - name: DownloadRemoteFile
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: C:\testfolder\package.zip
```

출력:

```
{
 "destination": "C:\\testfolder\\package.zip"
}
```

입력 예제: 둘 이상의 로컬 대상에 둘 이상의 원격 파일 다운로드

```
  - name: DownloadRemoteFiles
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: /tmp/java14_renamed.zip
      - source: https://testdomain/path/to/java14.zip
        destination: /tmp/create_new_folder_and_add_java14_as_zip/
```

출력:

```
{
 "destination": "/tmp/create_new_folder/java14_renamed.zip\n/tmp/
create_new_folder_and_add_java14_as_zip/java14.zip"
}
```

입력 예제: 로컬 대상을 덮어쓰지 않고 원격 파일 하나를 다운로드하고 파일 확인을 통해 다른 원격 파일을 다운로드합니다.

```
  - name: DownloadRemoteMultipleProperties
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: C:\create_new_folder\java14_renamed.zip
        overwrite: false
```

```
      - source: https://testdomain/path/to/java14.zip
        destination: C:\create_new_folder_and_add_java14_as_zip\
        checksum: ac68bbf921d953d1cfab916cb6120864
        algorithm: MD5
        overwrite: true
```

출력:

```
{
 "destination": "C:\\create_new_folder\\java14_renamed.zip\nC:\
\create_new_folder_and_add_java14_as_zip\\java14.zip"
}
```

입력 예제: 원격 파일 다운로드 및 SSL 인증 검증 무시

```
  - name: DownloadRemoteIgnoreValidation
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://www.bad-ssl.com/resource
        destination: /tmp/downloads/
        ignoreCertificateErrors: true
```

출력:

```
{
 "destination": "/tmp/downloads/resource"
}
```

파일 시스템 작업 모듈

다음 섹션에는 파일 시스템 작업을 수행하는 작업 모듈에 대한 세부 정보가 포함되어 있습니다.

파일 시스템 운영 작업 모듈

- AppendFile(Linux, Windows, macOS)
- CopyFile(Linux, Windows, macOS)
- CopyFolder(Linux, Windows, macOS)
- CreateFile(Linux, Windows, macOS)
- CreateFolder(Linux, Windows, macOS)

- CreateSymlink(Linux, Windows, macOS)
- DeleteFile(Linux, Windows, macOS)
- DeleteFolder(Linux, Windows, macOS)
- ListFiles(Linux, Windows, macOS)
- MoveFile(Linux, Windows, macOS)
- MoveFolder(Linux, Windows, macOS)
- ReadFile(Linux, Windows, macOS)
- SetFileEncoding(Linux, Windows, macOS)
- SetFileOwner(Linux, Windows, macOS)
- SetFolderOwner(Linux, Windows, macOS)
- SetFilePermissions(Linux, Windows, macOS)
- SetFolderPermissions(Linux, Windows, macOS)

AppendFile(Linux, Windows, macOS)

AppendFile 작업 모듈은 지정된 콘텐츠를 파일의 기존 내용에 추가합니다.

파일 인코딩 값이 기본 인코딩(`utf-8`) 값과 다른 경우, `encoding` 옵션을 사용하여 파일 인코딩 값을 지정할 수 있습니다. 기본적으로 `utf-16` 및 `utf-32`(은)는 리틀 엔디안 인코딩을 사용하는 것으로 간주됩니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 파일이 런타임에 존재하지 않습니다.
- 파일 내용을 수정할 수 있는 쓰기 권한이 없습니다.
- 파일 작업 중에 모듈에서 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
`path`	파일 경로.	String	예	해당 사항 없음	N/A	예

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
content	파일에 추가할 콘텐츠입니다.	String	No	빈 문자열	N/A	예
encoding	인코딩 표준입니다.	String	No	utf8	utf8, utf-8, utf16, utf-16, utf16-LE, utf-16-LE , utf16-BE, utf-16-BE , utf32, utf-32, utf32-LE, utf-32-LE , utf32-BE 및 utf-32-BE . 인코딩 옵션 값은 대/소문자를 구분하지 않습니다.	예

입력 예제: 인코딩 없이 파일 추가(Linux)

```
  - name: AppendingFileWithOutEncodingLinux
```

```
  action: AppendFile
  inputs:
    - path: ./Sample.txt
      content: "The string to be appended to the file"
```

입력 예제: 인코딩 없이 파일 추가(Windows)

```
- name: AppendingFileWithOutEncodingWindows
  action: AppendFile
  inputs:
    - path: C:\MyFolder\MyFile.txt
      content: "The string to be appended to the file"
```

입력 예제: 인코딩으로 파일 추가(Linux)

```
- name: AppendingFileWithEncodingLinux
  action: AppendFile
  inputs:
    - path: /FolderName/SampleFile.txt
      content: "The string to be appended to the file"
      encoding: UTF-32
```

입력 예제: 인코딩으로 파일 추가(Windows)

```
- name: AppendingFileWithEncodingWindows
  action: AppendFile
  inputs:
    - path: C:\MyFolderName\SampleFile.txt
      content: "The string to be appended to the file"
      encoding: UTF-32
```

입력 예제: 빈 문자열로 파일 추가(Linux)

```
- name: AppendingEmptyStringLinux
  action: AppendFile
  inputs:
    - path: /FolderName/SampleFile.txt
```

입력 예제: 빈 문자열로 파일 추가(Windows)

```
- name: AppendingEmptyStringWindows
```

```
    action: AppendFile
    inputs:
      - path: C:\MyFolderName\SampleFile.txt
```

출력

없음.

CopyFile(Linux, Windows, macOS)

CopyFile 작업 모듈은 지정된 소스의 파일을 지정된 대상으로 복사합니다. 기본적으로 모듈은 런타임 시 대상 폴더가 없는 경우 대상 폴더를 재귀적으로 만듭니다.

지정된 이름의 파일이 지정된 폴더에 이미 있는 경우, 작업 모듈은 기본적으로 기존 파일을 덮어씁니다. 덮어쓰기 옵션을 `false`(으)로 설정하여 이 기본 동작을 재정의할 수 있습니다. 덮어쓰기 옵션이 `false`(으)로 설정되어 있고 지정된 위치에 지정된 이름을 가진 파일이 이미 있는 경우, 작업 모듈은 오류를 반환합니다. 이 옵션은 기본적으로 덮어쓰는 Linux의 `cp` 명령과 동일하게 작동합니다.

소스 파일 이름에는 와일드카드(*)가 포함될 수 있습니다. 와일드카드 문자는 마지막 파일 경로 구분자(/ 또는 \) 뒤에만 사용할 수 있습니다. 와일드카드 문자가 소스 파일 이름에 포함된 경우, 와일드카드와 일치하는 모든 파일이 대상 폴더에 복사됩니다. 와일드카드 문자를 사용하여 둘 이상의 파일을 이동하려는 경우, `destination` 옵션 입력 시 대상 입력이 폴더임을 나타내는 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

대상 파일 이름이 원본 파일 이름과 다른 경우, `destination` 옵션을 사용하여 대상 파일 이름을 지정할 수 있습니다. 대상 파일 이름을 지정하지 않으면, 원본 파일의 이름이 대상 파일을 만드는 데 사용됩니다. 마지막 파일 경로 구분자(/ 또는 \) 뒤에 오는 모든 텍스트는 파일 이름으로 취급됩니다. 소스 파일과 같은 파일 이름을 사용하려면, `destination` 옵션 입력 시 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 폴더에 파일을 생성할 수 있는 권한이 없습니다.
- 소스 파일이 런타임에 존재하지 않습니다.
- 지정된 파일 이름을 가진 폴더가 이미 있으며 `overwrite` 옵션이 `false`(으)로 설정되어 있습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
source	소스 파일 경로입니다.	String	예	해당 사항 없음	N/A	예
destination	대상 파일 경로입니다.	String	예	해당 사항 없음	N/A	예
overwrite	false로 설정하면 지정된 위치에 지정된 이름을 가진 파일이 이미 있는 경우 대상 파일이 교체되지 않습니다.	불린(Boolean)	아니요	true	N/A	예

입력 예제: 파일 복사(Linux)

```
- name: CopyingAFileLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
```

입력 예제: 파일 복사(Windows)

```
- name: CopyingAFileWindows
  action: CopyFile
  inputs:
```

```
      - source: C:\MyFolder\Sample.txt
        destination: C:\MyFolder\destinationFile.txt
```

입력 예제: 소스 파일 이름을 사용하여 파일 복사(Linux)

```
- name: CopyingFileWithSourceFileNameLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/
```

입력 예제: 소스 파일 이름을 사용하여 파일 복사(Windows)

```
- name: CopyingFileWithSourceFileNameWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\
```

입력 예제: 와일드카드 문자를 사용하여 파일 복사(Linux)

```
- name: CopyingFilesWithWildCardLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

입력 예제: 와일드카드 문자를 사용하여 파일 복사(Windows)

```
- name: CopyingFilesWithWildCardWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

입력 예제: 덮어쓰지 않고 파일 복사(Linux)

```
- name: CopyingFilesWithoutOverwriteLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
```

```
        destination: /MyFolder/destinationFile.txt
        overwrite: false
```

입력 예제: 덮어쓰지 않고 파일 복사(Windows)

```
- name: CopyingFilesWithoutOverwriteWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
      overwrite: false
```

출력

없음.

CopyFolder(Linux, Windows, macOS)

CopyFolder 작업 모듈은 지정된 소스의 폴더를 지정된 대상으로 복사합니다. source 옵션에 대한 입력은 복사할 폴더이고, destination 옵션에 대한 입력은 소스 폴더의 콘텐츠가 복사되는 폴더입니다. 기본적으로 모듈은 런타임 시 대상 폴더가 없는 경우 대상 폴더를 재귀적으로 만듭니다.

지정된 이름의 폴더가 지정된 폴더에 이미 있는 경우, 작업 모듈은 기본적으로 기존 폴더를 덮어씁니다. 덮어쓰기 옵션을 false(으)로 설정하여 이 기본 동작을 재정의할 수 있습니다. 덮어쓰기 옵션이 false(으)로 설정되어 있고 지정된 위치에 지정된 이름을 가진 폴더가 이미 있는 경우, 작업 모듈은 오류를 반환합니다.

소스 폴더 이름에는 와일드카드(*)가 포함될 수 있습니다. 와일드카드 문자는 마지막 파일 경로 구분자(/ 또는 \) 뒤에만 사용할 수 있습니다. 와일드카드 문자가 소스 폴더 이름에 포함된 경우, 와일드카드와 일치하는 모든 폴더가 대상 폴더에 복사됩니다. 와일드카드 문자를 사용하여 폴더를 두 개 이상 복사하려는 경우, destination 옵션 입력 시 대상 입력이 폴더임을 나타내는 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

대상 폴더 이름이 소스 폴더 이름과 다른 경우, destination 옵션을 사용하여 대상 폴더 이름을 지정할 수 있습니다. 대상 폴더 이름을 지정하지 않으면, 원본 폴더의 이름이 대상 폴더를 만드는 데 사용됩니다. 마지막 파일 경로 구분자(/ 또는 \) 뒤에 오는 모든 텍스트는 폴더 이름으로 취급됩니다. 소스 폴더와 같은 폴더 이름을 사용하려면, destination 옵션 입력 시 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 폴더에 폴더를 생성할 수 있는 권한이 없습니다.
- 소스 폴더가 런타임에 존재하지 않습니다.
- 지정된 폴더 이름을 가진 폴더가 이미 있으며 `overwrite` 옵션이 `false`(으)로 설정되어 있습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
`source`	소스 폴더 경로입니다.	String	예	해당 사항 없음	N/A	예
`destination`	대상 폴더 경로입니다.	String	예	해당 사항 없음	N/A	예
`overwrite`	false로 설정하면 지정된 위치에 지정된 이름을 가진 폴더가 이미 있는 경우 대상 폴더가 바뀌지 않습니다.	불린(Boolean)	아니요	`true`	N/A	예

입력 예제: 폴더 복사(Linux)

```
- name: CopyingAFolderLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
```

```
        destination: /MyFolder/destinationFolder
```

입력 예제: 폴더 복사(Windows)

```
- name: CopyingAFolderWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\destinationFolder
```

입력 예제: 소스 폴더 이름을 사용하여 폴더 복사(Linux)

```
- name: CopyingFolderSourceFolderNameLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/
```

입력 예제: 소스 폴더 이름을 사용하여 폴더 복사(Windows)

```
- name: CopyingFolderSourceFolderNameWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\
```

입력 예제: 와일드카드 문자를 사용하여 폴더 복사(Linux)

```
- name: CopyingFoldersWithWildCardLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

입력 예제: 와일드카드 문자를 사용하여 폴더 복사(Windows)

```
- name: CopyingFoldersWithWildCardWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

입력 예제: 덮어쓰지 않고 폴더 복사(Linux)

```
- name: CopyingFoldersWithoutOverwriteLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/destinationFolder
      overwrite: false
```

입력 예제: 덮어쓰지 않고 폴더 복사(Windows)

```
- name: CopyingFoldersWithoutOverwrite
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SourceFolder
      destination: C:\MyFolder\destinationFolder
      overwrite: false
```

출력

없음.

CreateFile(Linux, Windows, macOS)

CreateFile 작업 모듈은 지정된 위치에 파일을 만듭니다. 기본적으로 필요한 경우, 모듈은 상위 폴더를 반복적으로 생성하기도 합니다.

파일이 지정된 폴더에 이미 있는 경우, 작업 모듈은 기본적으로 기존 파일을 잘라내거나 덮어씁니다. 덮어쓰기 옵션을 `false`(으)로 설정하여 이 기본 동작을 재정의할 수 있습니다. 덮어쓰기 옵션이 `false`(으)로 설정되어 있고 지정된 위치에 지정된 이름을 가진 파일이 이미 있는 경우, 작업 모듈은 오류를 반환합니다.

파일 인코딩 값이 기본 인코딩(`utf-8`) 값과 다른 경우, `encoding` 옵션을 사용하여 파일 인코딩 값을 지정할 수 있습니다. 기본적으로 `utf-16` 및 `utf-32`(은)는 리틀 엔디안 인코딩을 사용하는 것으로 간주됩니다.

`owner`, `group` 및 `permissions`(은)는 선택적 입력 사항입니다. `permissions`의 입력은 문자열 값이어야 합니다. 제공하지 않을 경우 파일은 기본값으로 생성됩니다. Windows 플랫폼에서는 이러한 옵션이 지원되지 않습니다. 이 작업 모듈은 Windows 플랫폼에서 `owner`, `group` 및 `permissions` 옵션을 사용하는 경우 검증하고 오류를 반환합니다.

이 작업 모듈은 운영 체제의 umask 기본값으로 정의된 권한을 가진 파일을 만들 수 있습니다. 기본값을 재정의하려면 umask 값을 설정해야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 상위 폴더에 파일이나 폴더를 만들 권한이 없습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	파일 경로.	String	예	해당 사항 없음	N/A	예
content	파일의 텍스트 콘텐츠입니다.	String	No	N/A	N/A	예
encoding	인코딩 표준입니다.	String	No	utf8	utf8, utf-8, utf16, utf-16, utf16-LE, utf-16-LE , utf16-BE, utf-16-BE , utf32, utf-32, utf32-LE, utf-32-LE , utf32-BE 및	예

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
					utf-32-BE . 인코딩 옵션 값은 대/소문자를 구분하지 않습니다.	
owner	사용자 이름 또는 ID입니다.	String	No	N/A	N/A	Windows에서 지원되지 않습니다.
group	그룹 이름 또는 ID입니다.	String	No	현재 사용자입니다.	N/A	Windows에서 지원되지 않습니다.
permissions	파일 권한입니다.	String	No	0666	N/A	Windows에서 지원되지 않습니다.

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
overwrite	지정된 파일의 이름이 이미 있는 경우, 이 값을 false(으)로 설정하면 기본적으로 파일이 잘리거나 덮어쓰기 되지 않도록 할 수 있습니다.	불린(Boolean)	아니요	true	N/A	예

입력 예제: 덮어쓰지 않고 파일 생성(Linux)

```
- name: CreatingFileWithoutOverwriteLinux
  action: CreateFile
  inputs:
    - path: /home/UserName/Sample.txt
      content: The text content of the sample file.
      overwrite: false
```

입력 예제: 덮어쓰지 않고 파일 생성(Windows)

```
- name: CreatingFileWithoutOverwriteWindows
  action: CreateFile
  inputs:
    - path: C:\Temp\Sample.txt
      content: The text content of the sample file.
      overwrite: false
```

입력 예제: 파일 속성이 있는 파일 생성

```
- name: CreatingFileWithFileProperties
  action: CreateFile
  inputs:
    - path: SampleFolder/Sample.txt
      content: The text content of the sample file.
      encoding: UTF-16
      owner: Ubuntu
      group: UbuntuGroup
      permissions: 0777
   - path: SampleFolder/SampleFile.txt
      permissions: 755
    - path: SampleFolder/TextFile.txt
      encoding: UTF-16
      owner: root
      group: rootUserGroup
```

입력 예제: 파일 속성이 없는 파일 생성

```
- name: CreatingFileWithoutFileProperties
  action: CreateFile
  inputs:
    - path: ./Sample.txt
    - path: Sample1.txt
```

입력 예제: Linux 정리 스크립트에서 섹션을 건너뛰기 위한 빈 파일 생성

```
- name: CreateSkipCleanupfile
  action: CreateFile
  inputs:
    - path: <skip section file name>
```

자세한 내용은 Linux 정리 스크립트를 오버라이드합니다. 섹션을 참조하세요.

출력

없음.

CreateFolder(Linux, Windows, macOS)

CreateFolder 작업 모듈은 지정된 위치에 폴더를 생성합니다. 기본적으로 필요한 경우, 모듈은 상위 폴더를 반복적으로 생성하기도 합니다.

폴더가 지정된 폴더에 이미 있는 경우, 작업 모듈은 기본적으로 기존 폴더를 잘라내거나 덮어씁니다. 덮어쓰기 옵션을 false(으)로 설정하여 이 기본 동작을 재정의할 수 있습니다. 덮어쓰기 옵션이 false(으)로 설정되어 있고 지정된 위치에 지정된 이름을 가진 폴더가 이미 있는 경우, 작업 모듈은 오류를 반환합니다.

owner, group 및 permissions(은)는 선택적 입력 사항입니다. permissions의 입력은 문자열 값이어야 합니다. Windows 플랫폼에서는 이러한 옵션이 지원되지 않습니다. 이 작업 모듈은 Windows 플랫폼에서 owner, group 및 permissions 옵션을 사용하는 경우 검증하고 오류를 반환합니다.

이 작업 모듈은 운영 체제의 umask 기본값으로 정의된 권한을 가진 폴더를 만들 수 있습니다. 기본값을 재정의하려면 umask 값을 설정해야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 위치에 폴더를 생성할 수 있는 권한이 없습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	폴더 경로입니다.	String	예	해당 사항 없음	N/A	예
owner	사용자 이름 또는 ID입니다.	String	No	현재 사용자입니다.	N/A	Windows에서 지원되지 않습니다.
group	그룹 이름 또는 ID입니다.	String	No	현재 사용자의 그룹입니다.	N/A	Windows에서 지원되지 않습니다.
permissions	폴더 권한입니다.	String	No	0777	N/A	Windows에서 지원되

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
						지 않습니다.
overwrite	지정된 파일의 이름이 이미 있는 경우, 이 값을 false(으)로 설정하면 기본적으로 파일이 잘리거나 덮어쓰기 되지 않도록 할 수 있습니다.	불린(Boolean)	아니요	true	N/A	예

입력 예제: 폴더 생성(Linux)

```
- name: CreatingFolderLinux
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/
```

입력 예제: 폴더 생성(Windows)

```
- name: CreatingFolderWindows
  action: CreateFolder
  inputs:
    - path: C:\MyFolder
```

입력 예제: 폴더 속성을 지정하는 폴더 생성

```
- name: CreatingFolderWithFolderProperties
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      owner: SampleOwnerName
      group: SampleGroupName
      permissions: 0777
    - path: /Sample/MyFolder/SampleFoler/
      permissions: 777
```

입력 예제: 기존 폴더를 덮어쓰는 폴더가 있는 경우 해당 폴더를 생성합니다.

```
- name: CreatingFolderWithOverwrite
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      overwrite: true
```

출력

없음.

CreateSymlink(Linux, Windows, macOS)

CreateSymlink 작업 모듈은 심볼 링크 또는 다른 파일에 대한 참조가 포함된 파일을 생성합니다. 이 모듈은 Windows 플랫폼에서 지원되지 않습니다.

`path` 및 `target` 옵션의 입력은 절대 경로이거나 상대 경로일 수 있습니다. `path` 옵션의 입력이 상대 경로인 경우, 링크를 만들 때 절대 경로로 대체됩니다.

기본적으로 지정된 이름의 링크가 지정된 폴더에 이미 있는 경우, 작업 모듈은 오류를 반환합니다. `force` 옵션을 `true`(으)로 설정하여 이 기본 동작을 재정의할 수 있습니다. `force` 옵션을 `true`(으)로 설정하면 모듈이 기존 링크를 덮어씁니다.

상위 폴더가 없는 경우, 작업 모듈은 기본적으로 폴더를 반복적으로 만듭니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 대상 파일이 런타임에 존재하지 않습니다.
- 지정된 이름을 가진 비 심볼 링크 파일이 이미 있습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	파일 경로.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
target	심볼 링크가 가리키는 대상 파일 경로입니다.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
force	같은 이름의 링크가 이미 있는 경우 링크를 강제로 생성합니다.	불린 (Boolean)	아니요	false	N/A	Windows에서 지원되지 않습니다.

입력 예제: 링크 생성을 강제하는 심볼 링크 생성

```
- name: CreatingSymbolicLinkWithForce
  action: CreateSymlink
  inputs:
    - path: /Folder2/Symboliclink.txt
      target: /Folder/Sample.txt
      force: true
```

입력 예제: 링크 생성을 강제하지 않는 심볼 링크 생성

```
- name: CreatingSymbolicLinkWithOutForce
  action: CreateSymlink
  inputs:
```

```
      - path: Symboliclink.txt
        target: /Folder/Sample.txt
```

출력

없음.

DeleteFile(Linux, Windows, macOS)

DeleteFile 작업 모듈은 지정된 위치에 있는 파일을 하나 또는 여러 개 삭제합니다.

path의 입력은 유효한 파일 경로이거나 파일 이름에 와일드카드 문자(*)가 포함된 파일 경로여야 합니다. 파일 이름에 와일드카드 문자를 지정하면 동일한 폴더 내에서 와일드카드와 일치하는 모든 파일이 삭제됩니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 삭제 작업을 수행할 수 있는 권한이 없습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	파일 경로.	String	예	해당 사항 없음	N/A	예

입력 예제: 단일 파일 삭제(Linux)

```
  - name: DeletingSingleFileLinux
    action: DeleteFile
    inputs:
      - path: /SampleFolder/MyFolder/Sample.txt
```

입력 예제: 단일 파일 삭제(Windows)

```
  - name: DeletingSingleFileWindows
    action: DeleteFile
```

```
    inputs:
      - path: C:\SampleFolder\MyFolder\Sample.txt
```

입력 예제: 'log'로 끝나는 파일 삭제(Linux)

```
- name: DeletingFileEndingWithLogLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/*log
```

입력 예제: 'log'로 끝나는 파일 삭제(Windows)

```
- name: DeletingFileEndingWithLogWindows
  action: DeleteFile
  inputs:
    - path: C:\SampleFolder\MyFolder\*log
```

입력 예제: 지정된 폴더의 모든 파일 삭제(Linux)

```
- name: DeletingAllFilesInAFolderLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/*
```

입력 예제: 지정된 폴더의 모든 파일 삭제(Windows)

```
- name: DeletingAllFilesInAFolderWindows
  action: DeleteFile
  inputs:
    - path: C:\SampleFolder\MyFolder\*
```

출력

없음.

DeleteFolder(Linux, Windows, macOS)

DeleteFolder 작업 모듈은 폴더를 삭제합니다.

폴더가 비어 있지 않은 경우, 폴더와 해당 내용을 제거하려면 `force` 옵션을 `true`로 설정해야 합니다. `force` 옵션을 `true`(으)로 설정하지 않은 상태에서 삭제하려는 폴더가 비어 있지 않으면, 작업 모듈에서 오류를 반환합니다. `force` 옵션의 기본값은 `false`입니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 삭제 작업을 수행할 수 있는 권한이 없습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	폴더 경로입니다.	String	예	해당 사항 없음	N/A	예
force	폴더가 비어 있는지 여부에 관계없이 폴더를 제거합니다.	불린(Boolean)	아니요	false	N/A	예

입력 예제: **force** 옵션을 사용하여 비어 있지 않은 폴더 삭제(Linux)

```
- name: DeletingFolderWithForceOptionLinux
  action: DeleteFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      force: true
```

입력 예제: **force** 옵션을 사용하여 비어 있지 않은 폴더 삭제(Windows)

```
- name: DeletingFolderWithForceOptionWindows
  action: DeleteFolder
  inputs:
    - path: C:\Sample\MyFolder\Sample\
      force: true
```

입력 예제: 폴더 삭제(Linux)

```
- name: DeletingFolderWithOutForceLinux
  action: DeleteFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
```

입력 예제: 폴더 삭제(Windows)

```
- name: DeletingFolderWithOutForce
  action: DeleteFolder
  inputs:
    - path: C:\Sample\MyFolder\Sample\
```

출력

없음.

ListFiles(Linux, Windows, macOS)

ListFiles 작업 모듈은 지정된 폴더의 파일을 나열합니다. 재귀 옵션을 `true`(으)로 설정하면, 하위 폴더의 파일이 나열됩니다. 이 모듈은 기본적으로 하위 폴더의 파일을 나열하지 않습니다.

지정된 패턴과 일치하는 이름을 가진 모든 파일을 나열하려면 `fileNamePattern` 옵션을 사용하여 패턴을 제공하십시오. `fileNamePattern` 옵션에는 와일드카드(*) 값을 적용할 수 있습니다. `fileNamePattern`(이)가 제공되면 지정된 파일 이름 형식과 일치하는 모든 파일이 반환됩니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 폴더가 런타임에 존재하지 않습니다.
- 지정된 상위 폴더에 파일이나 폴더를 만들 권한이 없습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
`path`	폴더 경로입니다.	String	예	해당 사항 없음	N/A	예

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
fileNamePattern	패턴과 일치하는 이름을 가진 모든 파일을 나열하기 위해 일치시킬 패턴입니다.	String	No	N/A	N/A	예
recursive	폴더의 파일을 재귀적으로 나열합니다.	불린(Boolean)	아니요	false	N/A	예

입력 예제: 지정된 폴더의 파일 나열(Linux)

```
- name: ListingFilesInSampleFolderLinux
  action: ListFiles
  inputs:
    - path: /Sample/MyFolder/Sample
```

입력 예제: 지정된 폴더의 파일 나열(Windows)

```
- name: ListingFilesInSampleFolderWindows
  action: ListFiles
  inputs:
    - path: C:\Sample\MyFolder\Sample
```

입력 예제: ‘log’로 끝나는 파일 나열(Linux)

```
- name: ListingFilesWithEndingWithLogLinux
  action: ListFiles
  inputs:
    - path: /Sample/MyFolder/
```

```
        fileNamePattern: *log
```

입력 예제: ‘log’로 끝나는 파일 나열(Windows)

```
  - name: ListingFilesWithEndingWithLogWindows
    action: ListFiles
    inputs:
      - path: C:\Sample\MyFolder\
        fileNamePattern: *log
```

입력 예제: 파일을 재귀적으로 나열

```
  - name: ListingFilesRecursively
    action: ListFiles
    inputs:
      - path: /Sample/MyFolder/
        recursive: true
```

출력

키 이름	설명	형식				
`files`	파일 목록입니다.	String				

출력 예제

```
{
 "files": "/sample1.txt,/sample2.txt,/sample3.txt"
}
```

MoveFile(Linux, Windows, macOS)

MoveFile 작업 모듈은 파일을 지정된 소스에서 지정된 대상으로 이동합니다.

파일이 지정된 폴더에 이미 있는 경우, 작업 모듈은 기본적으로 기존 파일을 덮어씁니다. 덮어쓰기 옵션을 `false`(으)로 설정하여 이 기본 동작을 재정의할 수 있습니다. 덮어쓰기 옵션이 `false`(으)로 설정되어 있고 지정된 위치에 지정된 이름을 가진 파일이 이미 있는 경우, 작업 모듈은 오류를 반환합니다. 이 옵션은 기본적으로 덮어쓰는 Linux의 `mv` 명령과 동일하게 작동합니다.

소스 파일 이름에는 와일드카드(*)가 포함될 수 있습니다. 와일드카드 문자는 마지막 파일 경로 구분자(/ 또는 \) 뒤에만 사용할 수 있습니다. 와일드카드 문자가 소스 파일 이름에 포함된 경우, 와일드카드와 일치하는 모든 파일이 대상 폴더에 복사됩니다. 와일드카드 문자를 사용하여 둘 이상의 파일을 이동하려는 경우, `destination` 옵션 입력 시 대상 입력이 폴더임을 나타내는 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

대상 파일 이름이 원본 파일 이름과 다른 경우, `destination` 옵션을 사용하여 대상 파일 이름을 지정할 수 있습니다. 대상 파일 이름을 지정하지 않으면, 원본 파일의 이름이 대상 파일을 만드는 데 사용됩니다. 마지막 파일 경로 구분자(/ 또는 \) 뒤에 오는 모든 텍스트는 파일 이름으로 취급됩니다. 소스 파일과 같은 파일 이름을 사용하려면, `destination` 옵션 입력 시 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 폴더에 파일을 생성할 수 있는 권한이 없습니다.
- 소스 파일이 런타임에 존재하지 않습니다.
- 지정된 파일 이름을 가진 폴더가 이미 있으며 `overwrite` 옵션이 `false`(으)로 설정되어 있습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
`source`	소스 파일 경로입니다.	String	예	해당 사항 없음	N/A	예
`destination`	대상 파일 경로입니다.	String	예	해당 사항 없음	N/A	예
`overwrite`	false로 설정하면 지정된 위치에 지정된 이름을 가	불린 (Boolean)	아니요	`true`	N/A	예

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
	진 파일이 이미 있는 경우 대상 파일이 교체되지 않습니다.					

입력 예제: 파일 이동(Linux)

```
- name: MovingAFileLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
```

입력 예제: 파일 이동(Windows)

```
- name: MovingAFileWindows
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
```

입력 예제: 소스 파일 이름을 사용하여 파일 이동(Linux)

```
- name: MovingFileWithSourceFileNameLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/
```

입력 예제: 소스 파일 이름을 사용하여 파일 이동(Windows)

```
- name: MovingFileWithSourceFileNameWindows
```

```
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder
```

입력 예제: 와일드카드 문자를 사용하여 파일 이동(Linux)

```
- name: MovingFilesWithWildCardLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

입력 예제: 와일드카드 문자를 사용하여 파일 이동(Windows)

```
- name: MovingFilesWithWildCardWindows
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder
```

입력 예제: 덮어쓰지 않고 파일 이동(Linux)

```
- name: MovingFilesWithoutOverwriteLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
      overwrite: false
```

입력 예제: 덮어쓰지 않고 파일 이동(Windows)

```
- name: MovingFilesWithoutOverwrite
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
      overwrite: false
```

출력

없음.

MoveFolder(Linux, Windows, macOS)

MoveFolder 작업 모듈은 폴더를 지정된 소스에서 지정된 대상으로 이동합니다. source 옵션에 대한 입력은 이동할 폴더이고 destination 옵션에 대한 입력은 소스 폴더의 내용이 이동되는 폴더입니다.

런타임 시 대상 상위 폴더 또는 destination 옵션에 대한 입력이 없는 경우, 모듈의 기본 동작은 지정된 대상에 폴더를 반복적으로 생성하는 것입니다.

소스 폴더와 동일한 폴더가 대상 폴더에 이미 있는 경우, 작업 모듈은 기본적으로 기존 폴더를 덮어씁니다. 덮어쓰기 옵션을 false(으)로 설정하여 이 기본 동작을 재정의할 수 있습니다. 덮어쓰기 옵션이 false(으)로 설정되어 있고 지정된 위치에 지정된 이름을 가진 폴더가 이미 있는 경우, 작업 모듈은 오류를 반환합니다.

소스 폴더 이름에는 와일드카드(*)가 포함될 수 있습니다. 와일드카드 문자는 마지막 파일 경로 구분자(/ 또는 \) 뒤에만 사용할 수 있습니다. 와일드카드 문자가 소스 폴더 이름에 포함된 경우, 와일드카드와 일치하는 모든 폴더가 대상 폴더에 복사됩니다. 와일드카드 문자를 사용하여 폴더를 두 개 이상 이동하려는 경우, destination 옵션 입력 시 대상 입력이 폴더임을 나타내는 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

대상 폴더 이름이 소스 폴더 이름과 다른 경우, destination 옵션을 사용하여 대상 폴더 이름을 지정할 수 있습니다. 대상 폴더 이름을 지정하지 않으면, 원본 폴더의 이름이 대상 폴더를 만드는 데 사용됩니다. 마지막 파일 경로 구분자(/ 또는 \) 뒤에 오는 모든 텍스트는 폴더 이름으로 취급됩니다. 소스 폴더와 같은 폴더 이름을 사용하려면, destination 옵션 입력 시 파일 경로 구분자(/ 또는 \)로 끝나야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 대상 폴더에 폴더를 생성할 수 있는 권한이 없습니다.
- 소스 폴더가 런타임에 존재하지 않습니다.
- 지정된 이름을 가진 폴더가 이미 있으며 overwrite 옵션이 false로 설정되어 있습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
source	소스 폴더 경로입니다.	String	예	해당 사항 없음	N/A	예
destination	대상 폴더 경로입니다.	String	예	해당 사항 없음	N/A	예
overwrite	false로 설정하면 지정된 위치에 지정된 이름을 가진 폴더가 이미 있는 경우 대상 폴더가 바뀌지 않습니다.	불린(Boolean)	아니요	true	N/A	예

입력 예제: 폴더 이동(Linux)

```
- name: MovingAFolderLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/destinationFolder
```

입력 예제: 폴더 이동(Windows)

```
- name: MovingAFolderWindows
  action: MoveFolder
```

```
    inputs:
      - source: C:\Sample\MyFolder\SourceFolder
        destination: C:\MyFolder\destinationFolder
```

입력 예제: 소스 폴더 이름을 사용하여 폴더 이동(Linux)

```
- name: MovingFolderWithSourceFolderNameLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
      destination: /MyFolder/
```

입력 예제: 소스 폴더 이름을 사용하여 폴더 이동(Windows)

```
- name: MovingFolderWithSourceFolderNameWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\
```

입력 예제: 와일드카드 문자를 사용하여 폴더 이동(Linux)

```
- name: MovingFoldersWithWildCardLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

입력 예제: 와일드카드 문자를 사용하여 폴더 이동(Windows)

```
- name: MovingFoldersWithWildCardWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

입력 예제: 덮어쓰지 않고 폴더 이동(Linux)

```
- name: MovingFoldersWithoutOverwriteLinux
  action: MoveFolder
```

```
    inputs:
      - source: /Sample/MyFolder/SampleFolder
    destination: /MyFolder/destinationFolder
    overwrite: false
```

입력 예제: 덮어쓰지 않고 폴더 이동(Windows)

```
  - name: MovingFoldersWithoutOverwriteWindows
    action: MoveFolder
    inputs:
      - source: C:\Sample\MyFolder\SampleFolder
        destination: C:\MyFolder\destinationFolder
        overwrite: false
```

출력

없음.

ReadFile(Linux, Windows, macOS)

ReadFile 작업 모듈은 문자열 형식의 텍스트 파일 내용을 읽습니다. 이 모듈은 체인을 통해 후속 단계에서 사용할 파일 내용을 읽거나 `console.log` 파일로 데이터를 읽는 데 사용할 수 있습니다. 지정된 경로가 심볼 링크인 경우, 이 모듈은 대상 파일의 내용을 반환합니다. 이 모듈은 텍스트 파일만 지원합니다.

파일 인코딩 값이 기본 인코딩(`utf-8`) 값과 다른 경우, `encoding` 옵션을 사용하여 파일 인코딩 값을 지정할 수 있습니다. 기본적으로 `utf-16` 및 `utf-32`(은)는 리틀 엔디안 인코딩을 사용하는 것으로 간주됩니다.

기본적으로 이 모듈은 파일 내용을 `console.log` 파일에 인쇄할 수 없습니다. `printFileContent` 속성을 `true`로 설정하여 이 설정을 재정의할 수 있습니다.

이 모듈은 파일 내용만 반환할 수 있습니다. Excel 또는 JSON 파일과 같은 파일은 구문 분석할 수 없습니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 파일이 런타임에 존재하지 않습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	파일 경로.	String	예	해당 사항 없음	N/A	예
encoding	인코딩 표준입니다.	String	No	utf8	utf8, utf-8, utf16, utf-16, utf16-LE, utf-16-LE , utf16-BE, utf-16-BE , utf32, utf-32, utf32-LE, utf-32-LE , utf32-BE 및 utf-32-BE . 인코딩 옵션 값은 대/소문자를 구분하지 않습니다.	예
printFileContent	파일 내용을 console.log 파일	불린 (Boolean)	아니요	false	N/A	예.

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
	에 인쇄합니다.					

입력 예제: 파일 읽기(Linux)

```
- name: ReadingFileLinux
  action: ReadFile
  inputs:
    - path: /home/UserName/SampleFile.txt
```

입력 예제: 파일 읽기(Windows)

```
- name: ReadingFileWindows
  action: ReadFile
  inputs:
    - path: C:\Windows\WindowsUpdate.log
```

입력 예제: 파일 읽기 및 인코딩 표준 지정

```
- name: ReadingFileWithFileEncoding
  action: ReadFile
  inputs:
    - path: /FolderName/SampleFile.txt
      encoding: UTF-32
```

입력 예제: 파일 읽기 및 **console.log** 파일에 인쇄

```
- name: ReadingFileToConsole
  action: ReadFile
  inputs:
    - path: /home/UserName/SampleFile.txt
      printFileContent: true
```

출력

필드	설명	형식
content	파일 내용입니다.	문자열

출력 예제

```
{
 "content" : "The file content"
}
```

SetFileEncoding(Linux, Windows, macOS)

SetFileEncoding 작업 모듈은 기존 파일의 인코딩 속성을 수정합니다. 이 모듈은 파일 인코딩을 utf-8에서 지정된 인코딩 표준으로 변환할 수 있습니다. 기본적으로 utf-16 및 utf-32(은)는 리틀 엔디안 인코딩으로 간주됩니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 수정을 수행할 수 있는 권한이 없습니다.
- 파일이 런타임에 존재하지 않습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	파일 경로.	String	예	해당 사항 없음	N/A	예
encoding	인코딩 표준입니다.	String	No	utf8	utf8, utf-8, utf16, utf-16, utf16-LE,	예

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
					utf-16-LE , utf16-BE, utf-16-BE , utf32, utf-32, utf32-LE, utf-32-LE , utf32-BE 및 utf-32-BE . 인코딩 옵션 값은 대/소문자를 구분하지 않습니다.	

입력 예제: 파일 인코딩 속성 설정

```
- name: SettingFileEncodingProperty
  action: SetFileEncoding
  inputs:
    - path: /home/UserName/SampleFile.txt
      encoding: UTF-16
```

출력

없음.

SetFileOwner(Linux, Windows, macOS)

setFileOwner 작업 모듈은 기존 파일의 owner 및 group 소유자 속성을 수정합니다. 지정된 파일이 심볼 링크인 경우, 모듈은 소스 파일의 owner 속성을 수정합니다. 이 모듈은 Windows 플랫폼에서 지원되지 않습니다.

이 모듈은 사용자 및 그룹 이름을 입력으로 수용합니다. 그룹 이름을 제공하지 않으면, 모듈은 사용자가 속한 그룹에 파일의 그룹 소유자를 할당합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 수정을 수행할 수 있는 권한이 없습니다.
- 지정된 사용자 또는 그룹 이름이 런타임에 존재하지 않습니다.
- 파일이 런타임에 존재하지 않습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	파일 경로.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
owner	사용자 이름.	문자열	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
group	사용자 그룹의 이름입니다.	String	No	사용자가 속한 그룹의 이름입니다.	N/A	Windows에서 지원되지 않습니다.

입력 예제: 사용자 그룹 이름을 지정하지 않고 파일 소유자 속성 설정

```
- name: SettingFileOwnerPropertyNoGroup
  action: SetFileOwner
  inputs:
    - path: /home/UserName/SampleText.txt
      owner: LinuxUser
```

입력 예제: 소유자 및 사용자 그룹을 지정하여 파일 소유자 속성 설정

```
- name: SettingFileOwnerProperty
  action: SetFileOwner
  inputs:
    - path: /home/UserName/SampleText.txt
      owner: LinuxUser
      group: LinuxUserGroup
```

출력

없음.

SetFolderOwner(Linux, Windows, macOS)

SetFolderOwner 작업 모듈은 기존 폴더의 `owner` 및 `group` 소유자 및 속성을 재귀적으로 수정합니다. 기본적으로 모듈은 폴더의 모든 내용에 대한 소유권을 수정할 수 있습니다. `recursive` 옵션을 `false`(으)로 설정하여 이 동작을 재정의할 수 있습니다. 이 모듈은 Windows 플랫폼에서 지원되지 않습니다.

이 모듈은 사용자 및 그룹 이름을 입력으로 수용합니다. 그룹 이름을 제공하지 않으면, 모듈은 사용자가 속한 그룹에 파일의 그룹 소유자를 할당합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 수정을 수행할 수 있는 권한이 없습니다.
- 지정된 사용자 또는 그룹 이름이 런타임에 존재하지 않습니다.
- 폴더가 런타임에 존재하지 않습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	폴더 경로입니다.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
owner	사용자 이름.	문자열	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
group	사용자 그룹의 이름입니다.	String	No	사용자가 속한 그룹의 이름입니다.	N/A	Windows에서 지원되지 않습니다.
recursive	false(으)로 설정하면 폴더의 모든 내용에 대한 소유권을 수정하는 기본 동작을 재정의합니다.	불린(Boolean)	아니요	true	N/A	Windows에서 지원되지 않습니다.

입력 예제: 사용자 그룹 이름을 지정하지 않고 폴더 소유자 속성 설정

```
- name: SettingFolderPropertyWithOutGroup
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
```

```
        owner: LinuxUser
```

입력 예제: 폴더 내 모든 콘텐츠의 소유권을 재정의하지 않고 폴더 소유자 속성 설정

```
- name: SettingFolderPropertyWithOutRecursively
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
      recursive: false
```

입력 예제: 사용자 그룹 이름을 지정하여 파일 소유권 속성 설정

```
- name: SettingFolderPropertyWithGroup
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
      group: LinuxUserGroup
```

출력

없음.

SetFilePermissions(Linux, Windows, macOS)

SetFilePermissions 작업 모듈은 기존 파일의 `permissions`(을)를 수정합니다. 이 모듈은 Windows 플랫폼에서 지원되지 않습니다.

`permissions`의 입력은 문자열 값이어야 합니다.

이 작업 모듈은 운영 체제의 기본 umask 값으로 정의된 권한을 가진 파일을 만들 수 있습니다. 기본값을 재정의하려면 umask 값을 설정해야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 수정을 수행할 수 있는 권한이 없습니다.
- 파일이 런타임에 존재하지 않습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
path	파일 경로.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
permissions	파일 권한입니다.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.

입력 예제: 파일 권한 수정

```
- name: ModifyingFilePermissions
  action: SetFilePermissions
  inputs:
    - path: /home/UserName/SampleFile.txt
      permissions: 766
```

출력

없음.

SetFolderPermissions(Linux, Windows, macOS)

SetFolderPermissions 작업 모듈은 기존 폴더와 모든 하위 파일 및 하위 폴더의 permissions을 재귀적으로 수정합니다. 기본적으로 이 모듈은 지정된 폴더의 모든 내용에 대한 권한을 수정할 수 있습니다. recursive 옵션을 false(으)로 설정하여 이 동작을 재정의할 수 있습니다. 이 모듈은 Windows 플랫폼에서 지원되지 않습니다.

permissions의 입력은 문자열 값이어야 합니다.

이 작업 모듈은 운영 체제의 기본 umask 값에 따라 권한을 수정할 수 있습니다. 기본값을 재정의하려면 umask 값을 설정해야 합니다.

다음과 같은 상황이 발생하면 작업 모듈이 오류를 반환합니다.

- 지정된 수정을 수행할 수 있는 권한이 없습니다.
- 폴더가 런타임에 존재하지 않습니다.
- 작업 모듈에서 작업을 수행하는 동안 오류가 발생했습니다.

입력

키 이름	설명	형식	필수	기본값	허용되는 값	모든 플랫폼에서 지원됩니다.
`path`	폴더 경로입니다.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
`permissions`	폴더 권한입니다.	String	예	해당 사항 없음	N/A	Windows에서 지원되지 않습니다.
`recursive`	`false`로 설정된 경우 폴더의 모든 내용에 대한 권한을 수정하는 기본 동작을 재정의합니다.	불린(Boolean)	아니요	`true`	N/A	Windows에서 지원되지 않습니다.

입력 예제: 폴더 권한 설정

```
- name: SettingFolderPermissions
  action: SetFolderPermissions
```

```
    inputs:
      - path: SampleFolder/
        permissions: 0777
```

입력 예제: 폴더의 모든 콘텐츠에 대한 권한을 수정하지 않고 폴더 권한 설정

```
  - name: SettingFolderPermissionsNoRecursive
    action: SetFolderPermissions
    inputs:
      - path: /home/UserName/SampleFolder/
        permissions: 777
        recursive: false
```

출력

없음.

소프트웨어 설치 작업

다음 섹션에서는 소프트웨어를 설치하거나 제거하는 작업 모듈에 대해 설명합니다.

IAM 요구 사항

설치 다운로드 경로가 S3 URI인 경우, 인스턴스 프로파일에 연결하는 IAM 역할에 `S3Download` 작업 모듈을 실행할 권한이 있어야 합니다. 필요한 권한을 부여하려면 `S3:GetObject` IAM 정책을 인스턴스 프로파일과 연결된 IAM 역할에 연결하고 버킷 경로를 지정하십시오. 예: *arn:aws:s3:::**BucketName**/**).

복합 MSI 입력

입력 문자열에 큰따옴표(")가 포함된 경우, 다음 방법 중 하나를 사용하여 올바르게 해석되도록 해야 합니다.

- 다음 예제와 같이 문자열 외부에 작은따옴표(')를 사용하여 문자열을 포함하고, 문자열 안에는 큰따옴표(")를 사용할 수 있습니다.

  ```
  properties:
    COMPANYNAME: '"Acme ""Widgets"" and ""Gizmos."""'
  ```

  이 경우 문자열 안에 아포스트로피를 사용해야 하는 경우에는 이스케이프해야 합니다. 즉, 아포스트로피 앞에 작은따옴표(')를 하나 더 사용해야 합니다.

- 문자열 바깥쪽에 큰따옴표(")를 사용하여 문자열을 포함할 수 있습니다. 또한 다음 예제와 같이 백슬래시 문자(\)를 사용하여 문자열 내의 큰따옴표를 이스케이프할 수 있습니다.

```
properties:
  COMPANYNAME: "\"Acme \"\"Widgets\"\" and \"\"Gizmos.\"\"\""
```

두 방법 모두 `COMPANYNAME="Acme ""Widgets"" and ""Gizmos."""` 값을 msiexec 명령에 전달합니다.

소프트웨어 설치 작업 모듈

- InstallMSI(Windows)
- UninstallMSI(Windows)

InstallMSI(Windows)

`InstallMSI` 작업 모듈은 MSI 파일을 사용하여 Windows 애플리케이션을 설치합니다. 로컬 경로, S3 객체 URI 또는 웹 URL을 사용하여 MSI 파일을 지정할 수 있습니다. 재부팅 옵션은 시스템의 재부팅 동작을 구성합니다.

AWSTOE 는 작업 모듈의 입력 파라미터를 기반으로 msiexec 명령을 생성합니다. `path`(MSI 파일 위치) 및 `logFile`(로그 파일 위치) 입력 파라미터의 값은 따옴표(")로 묶어야 합니다.

다음 MSI 종료 코드는 성공한 것으로 간주됩니다.

- 0(성공)
- 1614(ERROR_PRODUCT_UNINSTALLED)
- 1641(재부팅이 시작됨)
- 3010(재부팅 필요)

입력

키 이름	설명	형식	필수	기본값	허용되는 값
`path`	다음 중 하나를 사용하여 MSI 파일 위	String	예	해당 사항 없음	N/A

키 이름	설명	형식	필수	기본값	허용되는 값
	치를 지정합니다. • 로컬 파일 경로입니다. 경로는 절대 경로일 수도 있고 상대 경로일 수도 있음 • 유효한 S3 객체 URI입니다. • RFC 3986 표준을 따르는 유효한 웹 HTTP/HTTPS URL(HTTPS 권장)입니다. chaining 표현식이 허용됩니다.				

키 이름	설명	형식	필수	기본값	허용되는 값
reboot	작업 모듈이 성공적으로 실행된 후 나타나는 시스템 재부팅 동작을 구성합니다. 설정: • Force - msiexec 명령이 성공적으로 실행된 후 시스템 재부팅을 시작합니다. • Allow - msiexec 명령이 재부팅이 필요함을 나타내는 종료 코드를 반환하는 경우 시스템 재부팅을 시작합니다. • Skip - 재부팅을 건	String	No	Allow	Allow, Force, Skip

키 이름	설명	형식	필수	기본값	허용되는 값
	너뛰었음을 나타내는 정보 메시지를 `console.log` 파일에 기록합니다. 이 옵션은 msiexec 명령이 재부팅이 필요함을 나타내는 종료 코드를 반환하더라도 재부팅을 방지합니다.				

키 이름	설명	형식	필수	기본값	허용되는 값
logOptions	MSI 설치 로깅에 사용할 옵션을 지정합니다. 지정된 플래그는 /L 명령줄 파라미터와 함께 MSI 설치 관리자에 전달되어 로깅을 활성화합니다. 플래그가 지정되지 않은 경우 기본값을 AWSTOE 사용합니다. MSI의 로그 옵션에 대한 자세한 내용은 Microsoft Windows 설치 프로그램 제품 설명서의 명령줄 옵션을 참조하세요.	String	No	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

키 이름	설명	형식	필수	기본값	허용되는 값
`logFile`	로그 파일 위치의 절대 경로 또는 상대 경로입니다. 로그 파일 경로가 없으면 생성됩니다. 로그 파일 경로가 제공되지 않은 경우는 MSI 설치 로그를 저장하지 AWSTOE 않습니다.	String	No	N/A	N/A

키 이름	설명	형식	필수	기본값	허용되는 값
properties	MSI 로깅 속성 키/값 페어, 예: `TARGETDIR: "C:\target\location"` 참고: 다음 속성은 수정할 수 없습니다. • `REBOOT="ReallySupress"` • `REINSTALLMODE="ecmus"` • `REINSTALL="ALL"`	Map[String]String	아니요	N/A	N/A

키 이름	설명	형식	필수	기본값	허용되는 값
ignoreAuthenticodeSignatureErrors	경로에 지정된 설치 프로그램의 Authenticode 서명 검증 오류를 무시하도록 설정하는 플래그입니다. Get-AuthenticodeSignature 명령은 설치 프로그램을 검증하는 데 사용됩니다. 설정: • true - 검증 오류는 무시되고 설치 프로그램이 실행됩니다. • false - 검증 오류가 무시되지 않습니다. 검증이 성공한 경우에만 설치 프로그램이	불린(Boolean)	아니요	false	true, false

키 이름	설명	형식	필수	기본값	허용되는 값
	실행됩니다. 이는 기본 설정 동작입니다.				

키 이름	설명	형식	필수	기본값	허용되는 값
`allowUnsignedInstaller`	경로에 지정된 서명되지 않은 설치 프로그램을 실행할 수 있도록 하는 플래그입니다. Get-AuthenticodeSignature 명령은 설치 프로그램을 검증하는 데 사용됩니다. 설정: • `true` - Get-AuthenticodeSignature 명령에서 반환된 `NotSigned` 상태를 무시하고 설치 프로그램을 실행합니다. • `false` - 설치 프로그램이 서명되어야 합	불린(Boolean)	아니요	`false`	`true`, `false`

키 이름	설명	형식	필수	기본값	허용되는 값
	니다. 서명되지 않은 설치 프로그램은 실행되지 않습니다. 이는 기본 설정 동작입니다.				

예시

다음 예제는 설치 경로에 따른 구성 요소 문서의 입력 섹션 변형을 보여줍니다.

입력 예제: 로컬 문서 경로 설치

```
- name: local-path-install
  steps:
    - name: LocalPathInstaller
      action: InstallMSI
      inputs:
        path: C:\sample.msi
        logFile: C:\msilogs\local-path-install.log
        logOptions: '*VX'
        reboot: Allow
        properties:
          COMPANYNAME: '"Amazon Web Services"'
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: true
```

입력 예제: Amazon S3 경로 설치

```
- name: s3-path-install
  steps:
    - name: S3PathInstaller
      action: InstallMSI
      inputs:
        path: s3://<bucket-name>/sample.msi
```

```
        logFile: s3-path-install.log
        reboot: Force
        ignoreAuthenticodeSignatureErrors: false
        allowUnsignedInstaller: true
```

입력 예제: 웹 경로 설치

```
- name: web-path-install
  steps:
    - name: WebPathInstaller
      action: InstallMSI
      inputs:
        path: https://<some-path>/sample.msi
        logFile: web-path-install.log
        reboot: Skip
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: false
```

출력

다음은 `InstallMSI` 작업 모듈의 출력 예제입니다.

```
{
 "logFile": "web-path-install.log",
 "msiExitCode": 0,
 "stdout": ""
}
```

UninstallMSI(Windows)

`UninstallMSI` 작업 모듈을 사용하면 MSI 파일을 사용하여 Windows 애플리케이션을 제거할 수 있습니다. 로컬 파일 경로, S3 객체 URI 또는 웹 URL을 사용하여 MSI 파일 위치를 지정할 수 있습니다. 재부팅 옵션은 시스템의 재부팅 동작을 구성합니다.

AWSTOE 는 작업 모듈의 입력 파라미터를 기반으로 msiexec 명령을 생성합니다. MSI 파일 위치(`path`)와 로그 파일 위치(`logFile`)는 msiexec 명령을 생성하는 동안 명시적으로 큰따옴표(")로 묶습니다.

다음 MSI 종료 코드는 성공한 것으로 간주됩니다.

- 0(성공)
- 1605(ERROR_UNKNOWN_PRODUCT)

- 1614(ERROR_PRODUCT_UNINSTALLED)
- 1641(재부팅이 시작됨)
- 3010(재부팅 필요)

입력

키 이름	설명	형식	필수	기본값	허용되는 값
path	다음 중 하나를 사용하여 MSI 파일 위치를 지정합니다. • 로컬 파일 경로입니다. 경로는 절대 경로일 수도 있고 상대 경로일 수도 있습니다. • 유효한 S3 객체 URI입니다. • RFC 3986 표준을 따르는 유효한 웹 HTTP/HTTPS URL(HTTPS 권장)입니다.	String	예	해당 사항 없음	N/A

키 이름	설명	형식	필수	기본값	허용되는 값
	chaining 표현식이 허용됩니다.				

키 이름	설명	형식	필수	기본값	허용되는 값
reboot	작업 모듈이 성공적으로 실행된 후 나타나는 시스템 재부팅 동작을 구성합니다. 설정: • Force - msiexec 명령이 성공적으로 실행된 후 시스템 재부팅을 시작합니다. • Allow - msiexec 명령이 재부팅이 필요함을 나타내는 종료 코드를 반환하는 경우 시스템 재부팅을 시작합니다. • Skip - 재부팅을 건	String	No	Allow	Allow, Force, Skip

키 이름	설명	형식	필수	기본값	허용되는 값
	너뛰었음을 나타내는 정보 메시지를 `console.log` 파일에 기록합니다. 이 옵션은 msiexec 명령이 재부팅이 필요함을 나타내는 종료 코드를 반환하더라도 재부팅을 방지합니다.				

키 이름	설명	형식	필수	기본값	허용되는 값
logOptions	MSI 설치 로깅에 사용할 옵션을 지정합니다. 지정된 플래그는 /L 명령줄 파라미터와 함께 MSI 설치 관리자에 전달되어 로깅을 활성화합니다. 플래그가 지정되지 않은 경우 기본값을 AWSTOE 사용합니다. MSI의 로그 옵션에 대한 자세한 내용은 Microsoft Windows 설치 프로그램 제품 설명서의 명령줄 옵션을 참조하세요.	String	No	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

키 이름	설명	형식	필수	기본값	허용되는 값
`logFile`	로그 파일 위치의 절대 경로 또는 상대 경로입니다. 로그 파일 경로가 없으면 생성됩니다. 로그 파일 경로가 제공되지 않은 경우는 MSI 설치 로그를 저장하지 AWSTOE 않습니다.	String	No	N/A	N/A

키 이름	설명	형식	필수	기본값	허용되는 값
properties	MSI 로깅 속성 키/값 페어, 예: `TARGETDIR: "C:\target\location"` 참고: 다음 속성은 수정할 수 없습니다. • `REBOOT="ReallySupress"` • `REINSTALLMODE="ecmus"` • `REINSTALL="ALL"`	Map[String]String	아니요	N/A	N/A

키 이름	설명	형식	필수	기본값	허용되는 값
ignoreAuthenticodeSignatureErrors	경로에 지정된 설치 프로그램의 Authenticode 서명 검증 오류를 무시하도록 설정하는 플래그입니다. Get-AuthenticodeSignature 명령은 설치 프로그램을 검증하는 데 사용됩니다. 설정: • true - 검증 오류는 무시되고 설치 프로그램이 실행됩니다. • false - 검증 오류가 무시되지 않습니다. 검증이 성공한 경우에만 설치 프로그램이	불린(Boolean)	아니요	false	true, false

키 이름	설명	형식	필수	기본값	허용되는 값
	실행됩니다. 이는 기본 설정 동작입니다.				

키 이름	설명	형식	필수	기본값	허용되는 값
`allowUnsignedInstaller`	경로에 지정된 서명되지 않은 설치 프로그램을 실행할 수 있도록 하는 플래그입니다. Get-AuthenticodeSignature 명령은 설치 프로그램을 검증하는 데 사용됩니다. 설정: • `true` - Get-AuthenticodeSignature 명령에서 반환된 `NotSigned` 상태를 무시하고 설치 프로그램을 실행합니다. • `false` - 설치 프로그램이 서명되어야 합	불린(Boolean)	아니요	`false`	`true`, `false`

키 이름	설명	형식	필수	기본값	허용되는 값
	니다. 서명되지 않은 설치 프로그램은 실행되지 않습니다. 이는 기본 설정 동작입니다.				

예시

다음 예제는 설치 경로에 따른 구성 요소 문서의 입력 섹션 변형을 보여줍니다.

입력 예제: 로컬 문서 경로 설치 제거

```
- name: local-path-uninstall
  steps:
    - name: LocalPathUninstaller
      action: UninstallMSI
      inputs:
        path: C:\sample.msi
        logFile: C:\msilogs\local-path-uninstall.log
        logOptions: '*VX'
        reboot: Allow
        properties:
          COMPANYNAME: '"Amazon Web Services"'
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: true
```

입력 예제: Amazon S3 경로 설치 제거

```
- name: s3-path-uninstall
  steps:
    - name: S3PathUninstaller
      action: UninstallMSI
      inputs:
        path: s3://<bucket-name>/sample.msi
```

```
        logFile: s3-path-uninstall.log
        reboot: Force
        ignoreAuthenticodeSignatureErrors: false
        allowUnsignedInstaller: true
```

입력 예제: 웹 경로 설치 제거

```
- name: web-path-uninstall
  steps:
    - name: WebPathUninstaller
      action: UninstallMSI
      inputs:
        path: https://<some-path>/sample.msi
        logFile: web-path-uninstall.log
        reboot: Skip
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: false
```

출력

다음은 `UninstallMSI` 작업 모듈의 출력 예제입니다.

```
{
 "logFile": "web-path-uninstall.log",
 "msiExitCode": 0,
 "stdout": ""
}
```

시스템 작업 모듈

다음 섹션에서는 시스템 작업을 수행하거나 시스템 설정을 업데이트하는 작업 모듈에 대해 설명합니다.

시스템 작업 모듈

- Reboot(Linux, Windows)
- SetRegistry(Windows)
- UpdateOS(Linux, Windows)

Reboot(Linux, Windows)

재부팅 작업 모듈은 인스턴스를 재부팅합니다. 재부팅 시작을 지연시킬 수 있는 구성 가능한 옵션이 포함되어 있습니다. 기본적으로 `delaySeconds`(은)는 `0`(으)로 설정되어 있으며, 이는 지연이 없음을 의미합니다. 재부팅 작업 모듈의 경우, 인스턴스 재부팅 시에는 적용되지 않기 때문에 단계 시간 초과가 지원되지 않습니다.

Systems Manager Agent가 애플리케이션을 호출하면, 종료 코드(Windows의 경우 `3010`, Linux의 경우 194)가 Systems Manager Agent에 전달됩니다. Systems Manager Agent는 스크립트에서 관리형 인스턴스 재부팅에 설명된 대로 시스템 재부팅을 처리합니다.

애플리케이션이 호스트에서 독립 실행형 프로세스로 호출되는 경우, 현재 실행 상태를 저장하고 재부팅 후 애플리케이션을 재실행하도록 재부팅 후 자동 실행 트리거를 구성한 다음 시스템을 재부팅합니다.

재부팅 후 자동 실행 트리거:

- Windows. AWSTOE (은)는 `SystemStartup`에서 자동으로 실행되는 트리거가 포함된 Windows 작업 스케줄러 항목을 만듭니다.
- Linux. AWSTOE (은)는 시스템 재부팅 후 자동으로 실행되는 작업을 crontab에 추가합니다.

```
@reboot /download/path/awstoe run --document s3://bucket/key/doc.yaml
```

이 트리거는 애플리케이션이 시작될 때 정리됩니다.

> **재시도**
>
> 기본적으로 최대 재시도 횟수는 Systems Manager `CommandRetryLimit`으로 설정됩니다. 재부팅 횟수가 재시도 제한을 초과하면, 자동화가 실패합니다. Systems Manager 에이전트 구성 파일(`Mds.CommandRetryLimit`)을 편집하여 제한을 변경할 수 있습니다. Systems Manager 에이전트 오픈 소스의 런타임 구성을 참조하세요.

Reboot 작업 모듈을 사용하려면, `exitcode` 재부팅이 포함된 단계(예: `3010`)의 경우 애플리케이션 바이너리를 `sudo user`(으)로 실행해야 합니다.

입력

키 이름	설명	형식	필수	기본값
delaySeconds	재부팅을 시작하기 전에 특정 시간을 지연시킵니다.	Integer	아니요	0

입력 예제: 재부팅 단계

```
- name: RebootStep
  action: Reboot
  onFailure: Abort
  maxAttempts: 2
  inputs:
    delaySeconds: 60
```

출력

없음.

Reboot 모듈이 완료되면, Image Builder는 빌드의 다음 단계를 계속 진행합니다.

SetRegistry(Windows)

SetRegistry 작업 모듈은 입력 목록을 허용하고 지정된 레지스트리 키의 값을 설정할 수 있도록 합니다. 레지스트리 키가 없으면, 정의된 경로에 새로 생성됩니다. 이 기능은 Windows에만 적용됩니다.

입력

키 이름	설명	형식	필수
path	레지스트리 키의 경로입니다.	String	예
name	레지스트리 키의 이름입니다.	String	예
value	레지스트리 키의 값입니다.	문자열/숫자/배열	예

키 이름	설명	형식	필수
type	레지스트리 키의 값 유형입니다.	String	예

지원되는 경로 접두사

- HKEY_CLASSES_ROOT / HKCR:
- HKEY_USERS / HKU:
- HKEY_LOCAL_MACHINE / HKLM:
- HKEY_CURRENT_CONFIG / HKCC:
- HKEY_CURRENT_USER / HKCU:

지원되는 유형

- BINARY
- DWORD
- QWORD
- SZ
- EXPAND_SZ
- MULTI_SZ

입력 예제: 레지스트리 키 값 설정

```
- name: SetRegistryKeyValues
  action: SetRegistry
  maxAttempts: 3
  inputs:
    - path: HKLM:\SOFTWARE\MySoftWare
      name: MyName
      value: FirstVersionSoftware
      type: SZ
    - path: HKEY_CURRENT_USER\Software\Test
      name: Version
      value: 1.1
```

```
        type: DWORD
```

출력

없음.

UpdateOS(Linux, Windows)

UpdateOS 작업 모듈은 Windows 및 Linux 업데이트 설치에 대한 지원을 추가합니다. 기본적으로 사용 가능한 모든 업데이트가 설치됩니다. 또는 설치할 작업 모듈에 대한 하나 이상의 특정 업데이트 목록을 구성할 수 있습니다. 설치에서 제외할 업데이트를 지정할 수도 있습니다.

'포함' 목록과 '제외' 목록을 모두 제공하는 경우 결과 업데이트 목록에는 '포함' 목록에 나열되어 있지만 '제외' 목록에 나열되지 않은 업데이트만 포함될 수 있습니다.

> **Note**
>
> UpdateOS는 Amazon Linux 2023(AL2023)을 지원하지 않습니다. 기본 AMI를 모든 릴리스와 함께 제공되는 새 버전으로 업데이트하는 것이 좋습니다. 다른 대안은 Amazon Linux 2023 사용 설명서의 메이저 릴리스와 마이너 릴리스에서 받은 업데이트 제어를 참조하세요.

- Windows. 업데이트는 대상 컴퓨터에 구성된 업데이트 소스에서 설치됩니다.
- Linux. 애플리케이션은 Linux 플랫폼에서 지원되는 패키지 관리자를 확인하고 `yum` 또는 `apt-get` 패키지 관리자를 사용합니다. 둘 다 지원되지 않으면 오류가 반환됩니다. UpdateOS 작업 모듈을 실행할 `sudo` 권한이 있어야 합니다. `sudo` 권한이 없는 경우 `error.Input`(이)가 반환됩니다.

입력

키 이름	설명	형식	필수
`include`	Windows의 경우, 다음을 지정할 수 있습니다. • 설치할 수 있는 업데이트 목록에 포함할 하나 이상의	문자열 목록	아니요

키 이름	설명	형식	필수
	Microsoft 기술 자료(KB) 문서 ID입니다. 유효한 값은 KB1234567 또는 1234567입니다. • 와일드카드 값(*)을 사용한 업데이트 이름입니다. 유효한 값은 Security* 또는 *Security* 입니다. Linux의 경우 설치용 업데이트 목록에 포함할 패키지를 하나 이상 지정할 수 있습니다.		

키 이름	설명	형식	필수
exclude	Windows의 경우, 다음을 지정할 수 있습니다. • 설치에서 제외할 업데이트 목록에 포함할 하나 이상의 Microsoft 기술 자료(KB) 문서 ID입니다. 유효한 값은 KB1234567 또는 1234567입니다. • 와일드카드(*) 값을 사용한 업데이트 이름입니다. 유효한 값은 Security* 또는 *Security* 입니다. Linux의 경우, 설치용 업데이트 목록에서 제외할 패키지를 하나 이상 지정할 수 있습니다.	문자열 목록	아니요

입력 예제: Linux 업데이트 설치에 대한 지원 추가

```
- name: UpdateMyLinux
  action: UpdateOS
  onFailure: Abort
  maxAttempts: 3
  inputs:
```

```
      exclude:
        - ec2-hibinit-agent
```

입력 예제: Windows 업데이트 설치에 대한 지원 추가

```
- name: UpdateWindowsOperatingSystem
  action: UpdateOS
  onFailure: Abort
  maxAttempts: 3
  inputs:
    include:
      - KB1234567
      - '*Security*'
```

출력

없음.

AWSTOE run 명령에 대한 입력 구성

명령에 대한 명령줄 입력을 간소화하기 위해 .json 파일 확장자가 있는 JSON 형식 입력 구성 파일에 명령 파라미터 및 옵션에 대한 설정을 포함할 수 있습니다 AWSTOE run.는 다음 위치 중 하나에서 파일을 읽을 AWSTOE 수 있습니다.

- 로컬 파일 경로(*./config.json*).
- S3 버킷(*s3://<bucket-path>/<bucket-name>/config.json*).

run 명령을 입력하면 --config 파라미터를 사용하여 입력 구성 파일을 지정할 수 있습니다. 예시:

```
awstoe run --config <file-path>/config.json
```

구성 파일

입력 구성 JSON 파일에는 run 명령 파라미터 및 옵션을 통해 직접 제공할 수 있는 모든 설정에 대한 키-값 쌍이 포함되어 있습니다. 입력 구성 파일과 run 명령 모두에서 설정을 파라미터 또는 옵션으로 지정하는 경우, 다음과 같은 우선 순위 규칙이 적용됩니다.

우선 순위 규칙

1. 파라미터 또는 옵션을 AWS CLI통해의 run 명령에 직접 제공되는 설정은 동일한 설정에 대해 입력 구성 파일에 정의된 모든 값을 재정의합니다.

2. 입력 구성 파일의 설정이 구성 요소 기본값보다 우선합니다.

3. 구성 요소 문서에 다른 설정이 전달되지 않은 경우, 기본값이 있으면 기본값을 적용할 수 있습니다.

이 규칙에는 문서와 파라미터라는 두 가지 예외가 있습니다. 이러한 설정은 입력 구성과 명령 파라미터에서 다르게 작동합니다. 입력 구성 파일을 사용하는 경우, 이러한 파라미터를 run 명령에 직접 지정해서는 안 됩니다. 이렇게 하면 오류가 발생합니다.

구성 요소 설정

입력 구성 파일에는 다음 설정이 포함되어 있습니다. 파일을 간소화하기 위해, 필요하지 않은 선택적 설정은 생략할 수 있습니다. 달리 명시되지 않는 한 모든 설정은 선택 사항입니다.

- cwIgnoreFailures(부울) - CloudWatch Logs의 로깅 실패를 무시합니다.
- cwLogGroup(문자열) - CloudWatch Log의 `LogGroup` 이름입니다.
- cwLogRegion(문자열) - CloudWatch Logs에 적용되는 AWS 리전입니다.
- cwLogStream(문자열) - `console.log` 파일을 스트리밍할 AWSTOE 위치를 지정하는 CloudWatch Logs의 `LogStream` 이름입니다.
- documentS3BucketOwner(문자열) - S3 URI 기반 문서에 대한 버킷 소유자의 계정 ID입니다.
- documents(객체 배열, 필수) - AWSTOE run 명령이 실행 중인 YAML 구성 요소 문서를 나타내는 JSON 객체 배열입니다. 구성 요소 문서를 하나 이상 지정해야 합니다.

  각 객체는 다음과 같은 필드로 구성됩니다.

  - path(문자열, 필수) - YAML 구성 요소 문서의 파일 위치입니다. 이 값은 다음 중 하나여야 합니다.
    - 로컬 파일 경로(*`./component-doc-example.yaml`*).
    - S3 URI(`s3://`*`bucket/key`*).
    - 이미지 빌더 컴포넌트 빌드 버전 ARN(arn:aws:imagebuilder:us-west-*`2:123456789012`*:component/*`my-example-component`*/2021.12.02/1).
  - parameters(객체 배열) - 키-값 쌍 객체의 배열로, 각 객체는 구성 요소 문서를 실행할 때 run 명령이 전달하는 구성 요소별 파라미터를 나타냅니다. 구성 요소의 경우 파라미터는 선택 사항입니다. 컴포넌트 문서에는 파라미터가 정의되어 있을 수도 있고 그렇지 않을 수도 있습니다.

    각 객체는 다음과 같은 필드로 구성됩니다.

    - name(문자열, 필수) - 구성 요소 파라미터의 이름입니다.
    - value(문자열, 필수) - 명명된 파라미터에 대해 구성 요소 문서에 전달할 값입니다.

구성 요소 파라미터에 대한 자세한 내용은 사용자 지정 구성 요소 문서에서 변수 사용 페이지의 파라미터 섹션을 참조하세요.

- executonId(문자열) - 현재 run 명령 실행에 적용되는 고유 ID입니다. 이 ID는 해당 파일을 고유하게 식별하고 현재 명령 실행에 연결하기 위해 출력 및 로그 파일 이름에 포함됩니다. 이 설정을 생략하면 GUID가 AWSTOE 생성됩니다.
- logDirectory(문자열) -가이 명령 실행의 모든 로그 파일을 AWSTOE 저장하는 대상 디렉터리입니다. 기본적으로 이 파일은 `TOE_<DATETIME>_<EXECUTIONID>` 디렉터리에 위치합니다. 로그 디렉터리를 지정하지 않으면는 현재 작업 디렉터리()를 AWSTOE 사용합니다..
- logS3BucketName(문자열) - 구성 요소 로그가 Amazon S3에 저장된 경우(권장)이 파라미터에 이름이 지정된 S3 버킷에 구성 요소 애플리케이션 로그를 AWSTOE 업로드합니다.
- logS3BucketOwner(문자열) - 구성 요소 로그가 Amazon S3에 저장된 경우(권장),가 로그 파일을 AWSTOE 쓰는 버킷의 소유자 계정 ID입니다.
- logs3KeyPrefix(문자열) - 구성 요소 로그가 Amazon S3에 저장되어 있는 경우(권장), 이 접두사는 버킷 내 로그 위치의 S3 객체 키 접두사입니다.
- parameters(객체 배열) - 현재 run 명령 실행에 포함된 모든 구성 요소에 전체적으로 적용되는 파라미터를 나타내는 키-값 쌍 객체의 배열입니다.
  - name(문자열, 필수) - 글로벌 파라미터의 이름입니다.
  - value(문자열, 필수) - 명명된 파라미터에 대해 모든 구성 요소 문서에 전달할 값입니다.
- phase(문자열) - YAML 구성 요소 문서에서 실행할 단계를 지정하는 쉼표로 구분된 목록입니다. 구성 요소 문서에 추가 단계가 포함된 경우, 해당 단계는 실행되지 않습니다.
- StateDirectory(문자열) - 상태 추적 파일이 저장되는 파일 경로입니다.
- trace(부울) - 콘솔에 대한 자세한 로깅을 활성화합니다.

예시

`sampledoc.yaml` 및 `conversation-intro.yaml` 예제는 두 구성 요소 문서, `build` 및 `test` 단계를 실행하는 입력 구성 파일을 보여줍니다. 각 구성 요소 문서에는 자체에만 적용되는 파라미터가 있으며 두 문서 모두 하나의 공유 파라미터를 사용합니다. `project` 파라미터는 두 컴포넌트 문서에 모두 적용됩니다.

```
{
   "documents": [
     {
       "path": "<file path>/awstoe/sampledoc.yaml>",
       "parameters": [
```

```
        {
          "name": "dayofweek",
          "value": "Monday"
        }
      ]
    },
    {
      "path": "<file path>/awstoe/conversation-intro.yaml>",
      "parameters": [
        {
          "name": "greeting",
          "value": "Hello, HAL."
        }
      ]
    }
  ],
  "phases": "build,test",
  "parameters": [
    {
      "name": "project",
      "value": "examples"
    }
  ],
  "cwLogGroup": "<log_group_name>",
  "cwLogStream": "<log_stream_name>",
  "documentS3BucketOwner": "<owner_aws_account_number>",
  "executionId": "<id_number>",
  "logDirectory": "<local_directory_path>",
  "logS3BucketName": "<bucket_name_for_log_files>",
  "logS3KeyPrefix": "<key_prefix_for_log_files>",
  "logS3BucketOwner": "<owner_aws_account_number>"
}
```

Image Builder 출력 이미지 리소스

Image Builder로 AMI 또는 컨테이너 이미지에 대한 이미지 리소스를 생성한 후에는 Image Builder 콘솔, Image Builder API 또는 AWS CLI의 imagebuilder 명령을 사용하여 이미지 리소스를 관리할 수 있습니다.

> **Tip**
>
> 동일한 유형의 리소스가 여러 개 있는 경우, 태그를 지정하면 할당한 태그에 따라 특정 리소스를 식별하는 데 도움이 됩니다. 에서 Image Builder 명령을 사용하여 리소스에 태그를 지정하는 방법에 대한 자세한 내용은이 가이드의 리소스 태깅 섹션을 AWS CLI참조하세요.

이 섹션에서는 이미지를 나열하고, 보고, 생성하는 방법을 다룹니다. 이미지 워크플로 및 관리 방법에 대한 자세한 내용은 Image Builder 이미지의 빌드 및 테스트 워크플로 관리 섹션을 참조하세요.

내용

- 이미지 및 빌드 버전 나열
- 이미지 리소스 세부 정보 보기
- Image Builder를 사용하여 사용자 지정 이미지 생성
- Image Builder를 사용하여 가상 머신 이미지 가져오기 및 내보내기
- Image Builder를 사용하여 확인된 Windows ISO 디스크 이미지 가져오기
- Image Builder 이미지의 보안 결과 관리
- Image Builder 리소스 정리

이미지 및 빌드 버전 나열

Image Builder 콘솔의 이미지 페이지에서 사용자가 소유하고, 사용자와 공유되고, 사용자가 액세스할 수 있는 모든 Image Builder 이미지 리소스의 목록을 볼 수 있습니다. 목록 결과에는 해당 리소스에 대한 몇 가지 주요 세부 정보가 포함됩니다.

또한 계정에서 워크플로 작업이 보류 중인 이미지도 모두 볼 수 있습니다.

내용

- 이미지 나열

- 작업 대기 중인 이미지 나열
- 이미지 빌드 버전 나열

이미지 나열

이 섹션에서는 이미지에 대한 정보를 나열할 수 있는 다양한 방법을 설명합니다.

액세스할 수 있는 Image Builder 이미지 리소스를 다음 방법 중 하나를 사용하여 나열할 수 있습니다. API 작업에 대한 내용은 EC2 Image Builder API 참조의 ListImages를 참조하십시오. 연결된 SDK 요청의 경우, 같은 페이지의 참고 항목 링크를 참조하세요.

내용

- 콘솔에 이미지 나열
- AWS CLI 명령을 사용하여 이미지 나열

콘솔에 이미지 나열

다음 단계에 따라 콘솔에서 이미지 목록 페이지를 엽니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지를 선택합니다.

콘솔의 이미지 페이지는 이미지 소유권 또는 보류 중인 워크플로 작업에 따라 탭으로 구분됩니다. 이 섹션에서는 소유하거나 액세스 권한이 있는 이미지를 보여주는 처음 3개의 탭을 다룹니다.

콘솔 탭: 내 소유

내 소유 탭에서 다음 필터를 사용하여 이미지 목록 결과를 간소화할 수 있습니다.

- 검색 창에 이름 전체 또는 일부를 검색할 수 있습니다.
- 운영 체제 플랫폼(Windows, Linux 또는 macOS)에 따라 이미지를 필터링할 수 있습니다.
- 생성되는 출력 유형(AMI 또는 컨테이너 이미지)에 따라 이미지를 필터링할 수 있습니다.
- 필터 소스를 사용하여 가상 머신(VMIE) 또는 ISO 디스크 이미지에서 가져온 이미지를 찾을 수 있습니다.

필터 제어에 따라 내 소유 탭에는 사용자가 만든 Image Builder 이미지 목록이 표시되며 나열된 리소스에 대한 다음 세부 정보가 포함됩니다.

이름/버전

Image Builder 이미지 리소스 이름은 빌드된 레시피 이름과 버전으로 시작됩니다. 링크를 선택하면 관련된 모든 이미지 빌드 버전을 볼 수 있습니다.

유형

Image Builder가 이 이미지 리소스(AMI 또는 컨테이너 이미지)에 대해 생성한 출력 이미지 유형입니다.

플랫폼

이미지 리소스의 운영 체제 플랫폼(예: "Linux", "Windows" 또는 "macOS")입니다.

이미지 소스

Image Builder가 이 이미지 리소스를 만드는 데 사용한 기본 이미지의 출처입니다. 주로 가상 컴퓨터(VMIE)에서 가져온 이미지의 결과를 필터링하는 데 사용됩니다.

생성 시간

Image Builder에서 이미지 리소스의 현재 버전을 만든 날짜와 시간입니다.

ARN

현재 이미지 리소스 버전의 Amazon 리소스 이름(ARN)입니다.

콘솔 탭: 나와 공유됨

나와 공유됨 탭에서 다음 필터를 사용하여 이미지 목록 결과를 간소화할 수 있습니다.

- 검색 창에 이름 전체 또는 일부를 검색할 수 있습니다.
- 운영 체제 플랫폼(Windows, Linux 또는 macOS)에 따라 이미지를 필터링할 수 있습니다.
- 생성되는 출력 유형(AMI 또는 컨테이너 이미지)에 따라 이미지를 필터링할 수 있습니다.
- 필터 소스를 사용하여 가상 머신(VMIE) 또는 ISO 디스크 이미지에서 가져온 이미지를 찾을 수 있습니다.

필터 제어에 따라 나와 공유됨 탭에는 사용자와 공유된 Image Builder 이미지 목록이 표시되며 나열된 리소스에 대한 다음 세부 정보가 포함됩니다.

이미지 이름

사용자와 공유된 이미지 리소스의 이름입니다. 레시피에서 공유 이미지를 사용하려면 관리 이미지 선택 옵션을 선택하고 이미지 출처를 나와 공유된 이미지로 변경합니다.

유형

Image Builder가 이 이미지 리소스(AMI 또는 컨테이너 이미지)에 대해 생성한 출력 이미지 유형입니다.

버전

이미지 리소스의 운영 체제 플랫폼 버전으로, 일반적으로 <major>.<minor>.<patch> 형식의 숫자 필드입니다.

이미지 소스

해당하는 경우 Image Builder가 이 이미지 리소스를 만드는 데 사용한 기본 이미지의 출처입니다. 주로 가상 컴퓨터(VMIE)에서 가져온 이미지의 결과를 필터링하는 데 사용됩니다.

플랫폼

이미지 리소스의 운영 체제 플랫폼(예: "Linux", "Windows" 또는 "macOS")입니다.

생성 시간

Image Builder에서 사용자와 공유된 이미지 리소스 버전을 만든 날짜와 시간입니다.

소유자

공유된 이미지 리소스의 소유자입니다.

ARN

사용자와 공유된 이미지 리소스 버전의 Amazon 리소스 이름(ARN)입니다.

콘솔 탭: Amazon에서 관리

Amazon에서 관리 탭에서 다음 필터를 사용하여 이미지 목록 결과를 간소화할 수 있습니다.

- 검색 창에 이름 전체 또는 일부를 검색할 수 있습니다.
- 운영 체제 플랫폼(Windows, Linux 또는 macOS)에 따라 이미지를 필터링할 수 있습니다.
- 생성되는 출력 유형(AMI 또는 컨테이너 이미지)에 따라 이미지를 필터링할 수 있습니다.
- 필터 소스를 사용하여 가상 머신(VMIE) 또는 ISO 디스크 이미지에서 가져온 이미지를 찾을 수 있습니다.

필터 제어에 따라 Amazon에서 관리 탭에는 레시피의 기본 이미지로 사용할 수 있는 Amazon 관리형 Image Builder 이미지 목록이 표시됩니다. Image Builder는 나열된 리소스에 대해 다음과 같은 세부 정보를 표시합니다.

이미지 이름

관리형 이미지의 이름입니다. 레시피를 생성할 때 기본 이미지의 기본값은 빠른 시작(Amazon에서 관리)입니다. 이 탭에 나열된 이미지는 레시피를 생성할 때 기본 이미지로 선택한 운영 체제 플랫폼과 관련된 이미지 이름 목록을 채웁니다.

유형

Image Builder가 이 이미지 리소스(AMI 또는 컨테이너 이미지)에 대해 생성한 출력 이미지 유형입니다.

버전

이미지 리소스의 운영 체제 플랫폼 버전으로, 일반적으로 <major>.<minor>.<patch> 형식의 숫자 필드입니다.

플랫폼

이미지 리소스의 운영 체제 플랫폼(예: "Linux", "Windows" 또는 "macOS")입니다.

생성 시간

Image Builder에서 사용자와 공유된 이미지 리소스 버전을 만든 날짜와 시간입니다.

소유자

Amazon은 관리형 이미지를 소유합니다.

ARN

사용자와 공유된 이미지 리소스 버전의 Amazon 리소스 이름(ARN)입니다.

AWS CLI 명령을 사용하여 이미지 나열

에서 list-images 명령을 실행하면 소유하거나 액세스할 AWS CLI수 있는 이미지 목록을 가져올 수 있습니다.

다음 명령 예제는 필터 없이 list-images 명령을 사용하여 소유한 Image Builder 이미지 리소스를 모두 나열하는 방법을 보여줍니다.

예제: 모든 이미지 나열

```
aws imagebuilder list-images
```

출력:

```
{
 "requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
 "imageVersionList": [
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0",
    "platform": "Linux",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  },
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-win/1.0.1",
    "name": "image-recipe-win",
    "type": "AMI",
    "version": "1.0.1",
    "platform": "Windows",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  },
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
macos/1.1.1",
    "name": "image-recipe-macos",
    "type": "AMI",
    "version": "1.1.1",
    "platform": "macOS",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  }
 ]
}
```


list-images 명령을 실행할 때 다음 예제와 같이 필터를 적용하여 결과를 간소화할 수 있습니다. 결과를 필터링하는 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 list-images 명령을 참조하십시오.

예제: Linux 이미지용 필터

```
aws imagebuilder list-images --filters name="platform",values="Linux"
```

출력:

```
{
 "requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
 "imageVersionList": [
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0",
    "platform": "Linux",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  }
 ]
}
```


작업 대기 중인 이미지 나열

이미지 워크플로에서 `WaitForAction` 단계 작업을 사용하면 처리를 재개하거나 워크플로에 실패하라는 신호를 보낼 때까지 워크플로가 일시 중지됩니다. 계속하기 전에 외부 프로세스를 실행해야 하는 경우 이 단계 작업을 사용할 수 있습니다. 그런 다음 `SendWorkflowStepAction`을 사용하여 일시 중지된 단계에 `RESUME` 또는 `STOP`으로 신호를 보낼 수 있습니다. 콘솔에서 워크플로를 중지하거나 재개할 수도 있습니다.

다음 탭은 현재 일시 중지되어 재개 또는 중지 신호를 기다리는 워크플로 단계를 통해 계정의 모든 이미지 리소스 목록을 가져오는 방법을 보여줍니다. 탭에는 콘솔 단계와 AWS CLI 명령이 포함됩니다.

API 또는 SDK를 사용하여 작업을 기다리고 있는 워크플로 단계 목록을 가져올 수도 있습니다. API 작업에 대한 내용은 EC2 Image Builder API 참조의 ListWaitingWorkflowSteps를 참조하세요. 연결된 SDK 요청의 경우, 같은 페이지의 참고 항목 링크를 참조하세요.

Console

콘솔의 작업 대기 탭으로 이동하려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지를 선택합니다. 그러면 이미지 목록 페이지가 열립니다.
3. 목록 페이지에서 작업 대기 탭을 선택합니다.
4. (선택 사항) 단계를 중지하거나 재개하려면 이름 옆의 확인란을 선택한 다음 단계 중지 또는 단계 재개를 선택합니다. 확인란을 2개 이상 선택하여 선택한 모든 단계에 대해 동일한 작업을 수행할 수 있습니다.

보류 중인 워크플로 단계 세부 정보

보류 단계의 워크플로 세부 정보는 다음과 같습니다.

- 이미지 이름-보류 단계가 있는 이미지 리소스의 이름입니다. 이름 링크를 선택하여 해당 이미지의 세부 정보 페이지를 표시할 수 있습니다.
- 보류 단계 이름-작업을 기다리고 있는 워크플로 단계의 이름입니다.
- 단계 실행 ID-워크플로 단계의 런타임 인스턴스를 고유하게 식별합니다. 연결된 ID를 선택하여 단계의 런타임 세부 정보를 표시할 수 있습니다.
- 단계 시작-이 워크플로의 런타임 인스턴스가 시작된 시점의 타임스탬프입니다.
- 워크플로 ARN-보류 중인 단계가 있는 워크플로의 Amazon 리소스 이름(ARN)입니다.
- 작업-대기 상태에 있는 단계 작업입니다.

AWS CLI

에서 list-waiting-workflow-steps 명령을 실행하면 이미지 생성 프로세스를 완료하기 전에 작업을 기다리는 워크플로 단계가 있는 계정의 모든 이미지 목록이 AWS CLI표시됩니다.

다음 명령 예제는 list-waiting-workflow-steps 명령을 사용하여 작업 대기 중인 워크플로 단계가 있는 계정의 모든 이미지를 나열하는 방법을 보여줍니다.

예: 대기 중인 워크플로 단계가 있는 계정의 이미지 나열

```
aws imagebuilder list-waiting-workflow-steps
```

출력:

이 예제의 출력에는 작업을 대기 중인 단계가 있는 계정 내 이미지 하나가 표시됩니다.

```
{
    "steps": [
        {
            "imageBuildVersionArn": "arn:aws:imagebuilder:us-
west-2:111122223333:image/example-image/1.0.0/8",
            "name": "WaitForAction",
            "workflowExecutionId": "wf-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "stepExecutionId": "step-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "workflowBuildVersionArn": "arn:aws:imagebuilder:us-
west-2:111122223333:workflow/test/wait-for-action/1.0.0/1",
```

```
            "startTime": "2023-11-21T23:21:23.609Z",
            "action": "WaitForAction"
        }
    ]
}
```

이미지 빌드 버전 나열

Image Builder 콘솔의 이미지 빌드 버전 페이지에서 빌드 버전 목록과 소유한 이미지 리소스의 추가 세부 정보를 확인할 수 있습니다. Image Builder API, SDKs 또는에서 명령 또는 작업을 사용하여 이미지 빌드 버전을 나열 AWS CLI 할 수도 있습니다.

다음 방법 중 하나를 사용하여 소유한 이미지 리소스의 이미지 빌드 버전을 나열할 수 있습니다. API 작업에 대한 내용은 EC2 Image Builder API 참조의 ListImageBuildVersions를 참조하십시오. 연결된 SDK 요청의 경우, 같은 페이지의 참고 항목 링크를 참조하세요.

Console

버전 세부 정보

Image Builder 콘솔의 이미지 빌드 버전 페이지에 있는 세부 정보는 다음과 같습니다.

- 버전 — 이미지 리소스 빌드 버전. Image Builder 콘솔에서 버전은 이미지 세부 정보 페이지에 연결됩니다.
- 유형 — Image Builder가 이 이미지 리소스(AMI 또는 컨테이너 이미지)를 생성할 때 배포한 출력 유형.
- 생성 날짜 — Image Builder에서 이미지 빌드 버전을 만든 날짜 및 시간.
- 이미지 상태 – 이미지 빌드 버전의 현재 상태. 상태는 이미지 빌드 또는 배치와 관련될 수 있습니다. 예를 들어 빌드 프로세스 중에 `Building` 또는 `Distributing` 상태가 표시될 수 있습니다. 이미지 배치의 경우 `Deprecated` 또는 `Deleted` 상태가 표시될 수 있습니다.
- 실패 이유 — 이미지 상태의 이유. Image Builder 콘솔에는 빌드가 실패한 이유만 표시됩니다(이미지 상태는 `Failed`(와)과 같음).
- 보안 조사 결과 - 참조된 이미지 빌드 버전의 집계된 이미지 스캔 결과.
- ARN — 이미지 리소스의 참조 버전의 Amazon 리소스 이름(ARN).
- 로그 스트림 - 참조된 이미지 빌드 버전의 로그 스트림 세부 정보에 대한 링크.

목록 버전

Image Builder 콘솔에서 이미지 빌드 버전을 나열하려면 다음 단계를 수행하십시오.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지를 선택합니다. 기본적으로 이미지 목록에는 소유하고 있는 각 이미지의 현재 버전이 표시됩니다.
3. 이미지의 모든 버전 목록을 보려면 현재 버전 링크를 선택합니다. 링크를 클릭하면 특정 이미지의 모든 빌드 버전이 나열된 이미지 빌드 버전 페이지가 열립니다.

AWS CLI

에서 list-image-build-versions 명령을 실행하면 지정된 이미지 리소스에 대한 빌드 버전의 전체 목록이 AWS CLI표시됩니다. 이 명령을 실행하려면 이미지를 소유해야 합니다.

다음 명령 예제는 list-image-build-versions 명령을 사용하여 지정된 이미지의 모든 빌드 버전을 나열하는 방법을 보여줍니다.

예제: 특정 이미지의 빌드 버전 나열

```
aws imagebuilder list-image-build-versions --image-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0
```

출력:

이 예제의 출력에는 지정된 이미지 레시피에 대한 두 개의 빌드 버전이 포함됩니다.

```
{
  "requestId": "12f3e45d-67cb-8901-af23-45ed678c9b01",
  "imageSummaryList": [
    {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
name/1.0.0/2",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0/2",
    "platform": "Linux",
    "osVersion": "Amazon Linux 2",
    "state": {
      "status": "AVAILABLE"
    },
    "owner": "123456789012",
    "dateCreated": "2023-03-10T01:04:40.609Z",
```

```
    "outputResources": {
      "amis": [
        {
        "region": "us-west-2",
        "image": "ami-012b3456789012c3d",
        "name": "image-recipe-name 2023-03-10T01-05-12.541Z",
        "description": "First verison of image-recipe-name",
        "accountId": "123456789012"
        }
      ]
    },
    "tags": {}
    },
    {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
name/1.0.0/1",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0/1",
    "platform": "Linux",
    "osVersion": "Amazon Linux 2",
    "state": {
      "status": "AVAILABLE"
    },
    "owner": "123456789012",
    "dateCreated": "2023-03-10T00:07:16.384Z",
    "outputResources": {
      "amis": [
        {
        "region": "us-west-2",
        "image": "ami-0d1e23456789f0a12",
        "name": "image-recipe-name 2023-03-10T00-07-18.146132Z",
        "description": "First verison of image-recipe-name",
        "accountId": "123456789012"
        }
      ]
    },
    "tags": {}
    }
  ]
}
```

> **Note**
>
> 현재 list-image-build-versions 명령의 출력에는 보안 조사 결과나 로그 스트림이 포함되지 않습니다.

이미지 리소스 세부 정보 보기

Image Builder 콘솔의 이미지 세부 정보 페이지에서 소유하고 있는 특정 이미지 리소스의 세부 정보를 볼 수 있습니다. Image Builder API, SDK 또는 AWS CLI (으)로 명령 또는 작업해서 이미지 세부 정보를 가져올 수도 있습니다.

다른이 AWS Resource Access Manager (AWS RAM) 리소스 AWS 계정 공유를 통해 공유한 리소스에 대한 자세한 내용은 AWS RAM 사용 설명서의 공유 AWS 된 리소스 액세스를 참조하세요.

내용

- Image Builder 콘솔에서 이미지 세부 정보 보기
- 에서 이미지 정책 세부 정보 가져오기 AWS CLI

Image Builder 콘솔에서 이미지 세부 정보 보기

Image Builder 콘솔의 이미지 세부 정보 페이지에는 추가 정보가 탭으로 그룹화된 요약 섹션이 있습니다. 페이지 제목은 이미지를 만든 레시피의 이름 및 빌드 버전입니다. 탭이 이미지에 적용되지 않으면 탭이 비활성화되고 데이터가 표시되지 않습니다.

콘솔 세부 정보 섹션 및 탭

- 요약 섹션
- 출력 리소스 탭
- 인프라 구성 탭
- 배포 설정 탭
- 워크플로 탭
- 보안 조사 결과 탭
- 태그 탭

요약 섹션

요약 섹션은 페이지 너비를 포함하며 다음과 같은 세부 정보를 포함합니다. 이러한 세부 정보는 항상 표시됩니다.

레시피

빌드 버전을 포함하지 않는 레시피 이름 및 버전. 예를 들어 빌드 버전이 `sample-linux-recipe | 1.0.1/2`인 경우 레시피는 `sample-linux-recipe | 1.0.1`(이)고 빌드 버전은 2입니다.

생성 날짜

Image Builder에서 이미지 빌드 버전을 만든 날짜 및 시간.

이미지 상태

Image Build 버전의 현재 상태. 상태는 이미지 빌드 또는 배치와 관련될 수 있습니다. 예를 들어 빌드 프로세스 중에 `Building` 또는 `Distributing` 상태가 표시될 수 있습니다. 이미지 배치의 경우 `Deprecated` 또는 `Deleted` 상태가 표시될 수 있습니다.

실패 이유

이미지 상태의 이유. Image Builder 콘솔에는 빌드가 실패한 이유만 표시됩니다(이미지 상태는 `Failed`(와)과 같음).

출력 리소스 탭

출력 리소스 탭에는 현재 표시된 이미지 리소스의 출력 및 배포 세부 정보가 나열됩니다. Image Builder에서 표시하는 정보는 파이프라인이 이미지를 생성하는 데 사용한 레시피 유형에 따라 다음과 같이 달라집니다.

이미지 레시피

- 리전 — 이미지 열에 지정된 출력 Amazon Machine Image(AMI)의 배포 리전.
- 이미지 — Image Builder가 대상에 배포한 AMI의 ID. 이 ID는 Amazon EC2 콘솔의 Amazon Machine Image(AMI) 페이지에 연결됩니다.

> **Note**
>
> Image Builder는 출력 이미지 리소스를 생성한 후 AMI를 대상에 배포하기 전에 AMI를 생성합니다.

- 이름 — Image Builder가 대상에 배포한 AMI의 이름.
- 설명 - 파이프라인이 출력 이미지 리소스를 생성하는 데 사용한 이미지 레시피의 선택적 설명.
- 계정 - 현재 표시된 Image Builder 이미지 리소스를 소유 AWS 계정 한 입니다.

컨테이너 레시피

Image Builder는 컨테이너 레시피에서 생성된 출력에 대해 다음과 같은 세부 정보를 표시합니다.

- 리전 — 이미지 URI 열에 지정된 컨테이너 이미지의 배포 리전.
- 이미지 URI — Image Builder가 대상 지역의 ECR 리포지토리에 배포한 출력 컨테이너 이미지의 URI.

> **Note**
>
> Image Builder는 대상 당 하나의 행을 표시합니다. 출력 이미지에는 이미지를 만든 계정에 배포하기 위한 항목이 항상 하나 이상 있습니다. 추가 대상에는 리전 간 배포 AWS 계정또는가 포함될 수 있습니다 AWS Organizations. 자세한 내용은 Image Builder 배포 설정 관리 단원을 참조하십시오.

인프라 구성 탭

인프라 구성 탭에는 Image Builder가 현재 표시된 이미지를 구축하고 테스트하는 데 사용한 Amazon EC2 인프라 설정이 표시됩니다. Image Builder는 항상 인프라 구성 리소스 이름(구성 이름)과 해당 Amazon 리소스 이름(ARN)을 표시합니다. 인프라 구성에서 값을 설정하는 경우 추가 인프라 세부 정보에 다음이 포함될 수 있습니다.

- 인스턴스 타입
- 인스턴스 프로파일
- 네트워크 인프라
- 보안 그룹 설정
- Image Builder가 애플리케이션 로그를 저장하는 Amazon S3 위치
- 문제 해결을 위한 Amazon EC2 키 페어
- 이벤트 알림을 위한 Amazon SNS 주제

자세한 내용은 Image Builder 인프라 구성 관리 단원을 참조하십시오.

배포 설정 탭

배포 설정 탭에는 Image Builder에서 출력 이미지를 배포하는 데 사용한 설정이 표시됩니다. Image Builder는 항상 배포 구성 리소스 이름(구성 이름)과 해당 Amazon 리소스 이름(ARN)을 표시합니다. 추가 배포 세부 정보는 Image Builder 파이프라인이 이미지를 생성하는 데 사용한 레시피 유형에 따라 다음과 같이 달라집니다.

이미지 레시피

배포 구성 리소스에서 값을 설정하는 경우 추가 배포 세부 정보에는 다음이 포함될 수 있습니다.

- 리전 — 출력용 Amazon Machine Image(AMI)의 배포 리전.
- 출력 AMI 이름 — Image Builder가 대상에 배포한 AMI의 이름.
- 암호화(KMS 키) - 구성된 경우 Image Builder에서 대상 지역에 배포하기 위해 이미지를 암호화하는 데 사용한 AWS KMS key .
- 배포 대상 계정 - 교차 계정 배포를 구성한 경우이 열에는 대상 리전에서 출력 이미지를 AWS 계정 공유할의 쉼표로 구분된 목록이 표시됩니다.
- 공유 권한이 있는 보안 주체 - 예를 들어 AWS 계정 그룹 AWS Organizations 또는 조직 단위(OU)와 같이 이미지를 시작할 권한이 있는 보안 AWS 주체의 쉼표로 구분된 목록입니다.OUs

> **Note**
>
> 다른 보안 주체에게 이미지를 시작할 수 있는 권한을 부여해도 사용자는 여전히 Amazon EC2가 이미지에서 시작하는 모든 인스턴스에 대해 image. AWS bills 계정을 소유합니다.

- 더 빠른 시작 구성을 위한 대상 계정 - EC2 Fast Launch AWS 계정 가 시작을 위해 사전 프로비저닝된 스냅샷을 배포하는 입니다.
- 연결된 라이선스 구성 - 지정된 리전의 AMI와 연결된 License Manager 라이선스 구성 ARNs입니다.
- 시작 템플릿 구성 - 특정 계정에 사용할 Amazon EC2 시작 템플릿을 식별합니다.
- 시작 템플릿 기본 버전 설정 - 지정된 Amazon EC2 시작 템플릿을 지정된의 기본 시작 템플릿으로 설정합니다 AWS 계정.

컨테이너 레시피

컨테이너 배포에는 항상 다음 세부 정보가 포함됩니다.

- 리전 — 이미지 URI 열에 지정된 컨테이너 이미지의 배포 리전.
- 이미지 URI — Image Builder가 대상 리전의 Amazon ECR 리포지토리에 배포한 출력 컨테이너 이미지의 URI.

> **Note**
>
> Image Builder는 대상 당 하나의 행을 표시합니다. 출력 이미지에는 이미지를 만든 계정에 배포하기 위한 항목이 항상 하나 이상 있습니다. 추가 대상에는 리전 간 배포 AWS 계정또는가 포함될 수 있습니다 AWS Organizations. 자세한 내용은 Image Builder 배포 설정 관리 단원을 참조하십시오.

워크플로 탭

워크플로는 Image Builder가 새 이미지를 생성할 때 수행하는 단계 순서를 정의합니다. 모든 이미지에는 빌드 및 테스트 워크플로가 있습니다. 컨테이너에는 배포를 위한 추가 워크플로가 있습니다. 워크플로 탭에는 Image Builder가 이미지에 대해 실행한 해당 워크플로가 표시됩니다.

워크플로 유형 필터링

Image Builder는 기본적으로 빌드 또는 가져오기 워크플로 요약 및 워크플로 단계를 표시합니다. 하지만 워크플로 필터에는 이미지에 대해 진행 중이거나 완료된 모든 워크플로가 표시됩니다. 다른 워크플로를 보려면 목록에서를 선택합니다.

AMI 출력을 생성하는 이미지 워크플로에는 빌드, 가져오기 또는 테스트 워크플로가 있을 수 있습니다. 컨테이너 출력을 생성하는 컨테이너 워크플로에는 빌드, 테스트 또는 배포 워크플로가 있을 수 있습니다.

> **Note**
>
> 워크플로가 아직 시작되지 않은 경우 목록에 표시되지 않습니다. 예를 들어 빌드 워크플로와 테스트 워크플로가 모두 구성된 이미지 빌드가 방금 시작된 경우 목록에 표시되는 유일한 워크플로 유형은 빌드 워크플로입니다. 테스트 워크플로가 시작되면 Image Builder는 이를 목록에 추가합니다.

워크플로 필터에 따라, 선택한 워크플로에는 각 워크플로 유형에 대한 다음 세부 정보가 포함된 런타임 요약이 표시됩니다.

워크플로 상태

이 워크플로의 현재 런타임 상태. 다음이 값에 포함될 수 있습니다.

- 보류중
- 건너뜀
- 실행 중
- 완료됨
- Failed
- 롤백 진행 중
- 롤백 완료

실행 ID

Image Builder가 워크플로를 실행할 때마다 런타임 리소스를 추적하기 위해 할당하는 고유 식별자.

시작

이 워크플로의 런타임 인스턴스가 시작된 시점의 타임스탬프.

종료

이 워크플로의 런타임 인스턴스가 종료된 시점의 타임스탬프.

총 단계 수

워크플로의 총 단계 수. 이 값은 성공, 건너뛰기, 실패한 단계의 단계 수 합계와 같아야 합니다.

성공한 단계

성공적으로 실행된 워크플로의 단계 수에 대한 런타임 수.

실패한 단계

실패한 워크플로의 단계 수에 대한 런타임 수.

건너뛴 단계

건너뛴 워크플로의 단계 수에 대한 런타임 수.

다음 목록의 세부 정보는 워크플로의 이 런타임 인스턴스에 있는 모든 단계의 현재 상태를 보고합니다. Image Builder는 모든 이미지 유형에 대해 동일한 세부 정보를 표시합니다.

단계 #

Image Builder가 워크플로 단계를 실행하는 순서를 나타내는 숫자.

단계 ID

런타임 시 할당되는 워크플로 단계의 고유 식별자.

단계 상태

지정된 워크플로 단계의 현재 런타임 상태.

롤백 상태

워크플로의 이 런타임 인스턴스가 실패한 경우의 현재 롤백 상태.

단계 이름

지정된 워크플로의 이름.

시작

워크플로의 이 런타임 인스턴스의 특정 단계가 시작된 시점의 타임스탬프.

종료

워크플로의 이 런타임 인스턴스의 특정 단계가 종료된 시점의 타임스탬프.

보안 조사 결과 탭

검사를 활성화한 경우 보안 조사 결과 탭에는 일반적인 취약성 및 노출(CVE) 탐지 결과가 표시됩니다. Amazon Inspector는 Image Builder가 새 이미지를 생성하기 위해 시작한 테스트 인스턴스에서 이러한 결과를 식별했습니다. Image Builder가 이미지에 대한 결과를 캡처하도록 하려면 다음과 같이 스캔을 구성해야 합니다.

1. 계정에 대한 Amazon Inspector 스캔을 활성화하세요. 자세한 내용은 Amazon Inspector 사용 설명서의 Amazon Inspector 시작하기를 참조하세요.
2. 이 이미지를 생성하는 파이프라인의 보안 탐지 결과를 활성화하십시오. 파이프라인에 대한 보안 탐지 결과를 활성화하면 Image Builder는 테스트 인스턴스를 종료하기 전에 조사 결과의 스냅샷을 저장합니다. 자세한 내용은 에서 Image Builder 이미지에 대한 보안 스캔 구성 AWS Management Console 섹션을 참조하세요.

보안 조사 결과 탭에는 Amazon Inspector가 이미지에서 식별한 각 취약성에 대한 다음 세부 정보가 포함되어 있습니다.

심각도

CVE 조사 결과의 심각도 수준 값은 다음과 같습니다.

- 분류되지 않음
- 정보
- 낮음
- 중간
- 높음
- 심각

결과 ID

Amazon Inspector가 테스트 인스턴스를 스캔할 때 이미지에 대해 감지한 CVE 조사 결과의 고유 식별자. ID는 보안 조사 결과 > 취약성별 페이지에 연결되어 있습니다. 자세한 내용은 에서 Image Builder 이미지의 보안 조사 결과 관리 AWS Management Console 단원을 참조하십시오.

소스

CVE 조사 결과에 대한 취약성 정보의 출처.

경과 시간

이미지에서 해당 조사 결과가 처음 관찰된 이후 경과된 일수.

인스펙터 점수

Amazon Inspector에서 CVE 조사 결과에 할당한 점수.

태그 탭

태그 탭에는 이미지에 정의한 모든 태그가 표시됩니다.

에서 이미지 정책 세부 정보 가져오기 AWS CLI

다음 예제는 Amazon 리소스 이름(ARN)을 사용하여 이미지 정책의 세부 정보를 가져오는 방법을 보여 줍니다.

```
aws imagebuilder get-image-policy --image-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/example-image/2019.12.02
```

Image Builder를 사용하여 사용자 지정 이미지 생성

Image Builder 이미지를 새로 생성하는 데는 몇 가지 방법이 있습니다. 예를 들어 다음 방법 중 하나를 사용하여 AWS Management Console 또는를 사용하여 이미지를 생성할 수 있습니다 AWS CLI.

CreateImage API 작업을 사용하거나 빌드 파이프라인을 실행하여 이미지를 생성할 수도 있습니다. API 작업과 관련된 SDK 요청의 경우, EC2 Image Builder API 참조에서 해당 명령에 대한 관련 참고 항목 링크를 참조할 수도 있습니다.

AWS Management Console

기존 파이프라인에서 새 이미지를 생성하려면 다음과 같이 파이프라인을 수동으로 실행합니다. 또는 파이프라인 마법사를 사용하여 새 이미지를 처음부터 생성할 수도 있습니다. 생성할 이미지 유형에 따라 파이프라인 마법사: AMI 생성 또는 파이프라인 마법사: 컨테이너 이미지 생성 섹션을 참조하세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지 파이프라인을 선택합니다.
3. 실행할 파이프라인 이름 옆의 확인란을 선택합니다.
4. 이미지를 생성하려면 작업 메뉴에서 파이프라인 실행을 선택합니다. 이 작업은 파이프라인을 가동합니다.

또한 파이프라인 실행 일정을 지정하거나 Amazon EventBridge를 사용하여 구성한 규칙에 따라 파이프라인을 실행할 수 있습니다.

AWS CLI

에서 create-image 명령을 실행하기 전에 리소스가 아직 없는 경우 다음 리소스를 생성해야 AWS CLI합니다.

필수 리소스

- 레시피 - 다음과 같이 이미지에 대해 정확히 하나의 레시피를 지정해야 합니다.

  이미지 레시피

  `--image-recipe-arn` 파라미터를 사용하여 이미지 레시피 리소스에 대한 Amazon 리소스 이름(ARN)을 지정합니다.

  컨테이너 레시피

  `--container-recipe-arn` 파라미터를 사용하여 컨테이너 레시피 리소스의 ARN을 지정합니다.
- 인프라 구성 - `--infrastructure-configuration-arn` 파라미터를 사용하여 인프라 구성 리소스의 ARN을 지정합니다.

이미지에 필요한 다음 리소스를 지정할 수도 있습니다.

선택적 리소스 및 구성

- 배포 구성 - 기본적으로 Image Builder는 create-image 명령을 실행하는 리전의 계정에 출력 이미지 리소스를 배포합니다. 배포에 추가 대상 또는 구성을 제공하려면 `--distribution-configuration-arn` 파라미터를 사용하여 배포 구성 리소스의 ARN을 지정하세요.
- 이미지 스캔-이미지 또는 컨테이너 테스트 인스턴스에서 Amazon Inspector 결과에 대한 스냅샷을 구성하려면 `--image-scanning-configuration` 파라미터를 사용합니다. 컨테이너 이미지의 경우 Amazon Inspector에서 스캔에 사용하는 ECR 리포지토리도 지정합니다.
- 이미지 테스트 - 이미지 빌더 테스트 단계를 억제하려면, `--image-tests-configuration` 파라미터를 사용합니다. 또는 실행 시간에 대한 제한 시간을 설정할 수 있습니다.
- 이미지 태그 - `--tags` 파라미터를 사용하여 출력 이미지 리소스에 태그를 추가합니다.
- 이미지 워크플로-빌드 또는 테스트 워크플로를 지정하지 않는 경우 Image Builder는 기본 이미지 워크플로를 사용하여 이미지를 만듭니다. 생성한 워크플로를 지정하려면 `--workflows` 파라미터를 사용합니다.

> **Note**
>
> 이미지 워크플로를 지정하는 경우 Image Builder가 워크플로 작업을 실행하는 데 사용하는 IAM 역할의 이름 또는 ARN도 `--execution-role` 파라미터에 제공해야 합니다.

다음 예에서는 create-image AWS CLI 명령을 사용하여 이미지를 생성하는 방법을 보여줍니다. 자세한 내용은 AWS CLI 명령 참조를 참조하세요.

예: 디폴트 분포를 사용하여 기본 이미지 만들기

```
aws imagebuilder create-image --image-recipe-arn arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/simple-recipe-linux/1.0.0 --infrastructure-configuration-arn arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/simple-infra-config-linux
```

출력:

```
{
"requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
"imageVersionList": [
```

```
    {
        "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/simple-recipe-
linux/1.0.0",
        "name": "simple-recipe-linux",
        ...
    }
]
}
```

에서 이미지 생성 취소 AWS CLI

진행 중인 이미지 빌드를 취소하려면 다음과 같이 cancel-image-creation 명령을 사용합니다.

```
aws imagebuilder cancel-image-creation --image-build-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-recipe/2019.12.03/1
```

Image Builder를 사용하여 가상 머신 이미지 가져오기 및 내보내기

가상화 환경에서 VM을 내보내면 해당 프로세스가 VM 환경, 설정 및 데이터의 스냅샷 역할을 하는 하나 이상의 디스크 컨테이너 파일 세트를 생성합니다. 이러한 파일을 사용하여 VM을 가져와서 이미지 레시피의 기본 이미지로 사용할 수 있습니다. 내보내기를 위해 사용자 지정 이미지 빌드의 출력으로 VM 디스크 파일을 생성하고 파일을 배포할 수 있습니다.

Image Builder는 VM 디스크 컨테이너에 대해 다음과 같은 파일 형식을 지원합니다.

- 개방형 가상화 아카이브(OVA)
- 가상 머신 디스크(VMDK)
- 가상 하드 디스크(VHD/VHDX)
- 원시

가져오기는 디스크를 사용하여 Amazon Machine Image(AMI)와 Image Builder 이미지 리소스를 생성합니다. 이 리소스 중 하나는 사용자 지정 이미지 레시피의 기본 이미지로 사용될 수 있습니다. 가져오려면 VM 디스크를 S3 버킷에 저장해야 합니다. 또는 기존 EBS 스냅샷에서 가져올 수 있습니다.

Image Builder 콘솔에서 이미지를 직접 가져온 다음 레시피에 출력 이미지 또는 AMI를 사용하거나 레시피 또는 레시피 버전을 생성할 때 가져오기 파라미터를 지정할 수 있습니다. 이미지 레시피의 일부로 가져오는 방법에 대한 자세한 내용은 VM 가져오기 구성(을)를 참조합니다.

Image Builder로 VM 가져오기

Image Builder는 Amazon EC2 VM 가져오기/내보내기 API와 통합되어 가져오기 프로세스를 백그라운드에서 비동기적으로 실행할 수 있도록 합니다. Image Builder는 VM 가져오기의 작업 ID를 참조하여 진행 상황을 추적하고 Image Builder 이미지 리소스를 출력으로 생성합니다. 이렇게 하면 VM 가져오기가 완료되기 전에 레시피의 Image Builder 이미지 리소스를 참조할 수 있습니다.

Console

Image Builder 콘솔을 사용하여 VM을 가져오려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지를 선택합니다.
3. 가져오기 대화 상자를 열려면 이미지 가져오기를 선택합니다.
4. 다음 일반 정보를 입력합니다.
   - 이미지의 고유한 이름을 지정합니다.
   - 기본 이미지의 버전을 지정합니다. *major.minor.patch* 형식을 사용합니다.
5. 가져오기 유형: VM 가져오기를 선택합니다.
6. 이미지 가져오기 페이지의 다음 섹션에 대한 세부 정보를 제공하세요. 작업을 마치면 이미지 가져오기를 선택합니다.

기본 이미지 운영 체제

1. VM OS 플랫폼에 맞는 이미지 운영 체제(OS) 옵션을 선택합니다.
2. 목록에서 VM 버전과 일치하는 OS 버전을 선택합니다.

VM 가져오기 구성

1. 가상화 환경에서 VM을 내보내는 경우 이 프로세스는 하나 이상의 디스크 컨테이너 파일 세트를 만듭니다. 이는 VM 환경, 설정 및 데이터의 스냅샷 역할을 합니다. 이러한 파일을 사용하여 VM을 이미지 레시피의 기본 이미지로 가져올 수 있습니다. 이미지 빌더에서 VM을 가져오는 작업에 대한 자세한 내용은 VM 이미지 가져오기 및 내보내기 섹션을 참조하세요.

   가져오기 소스의 위치를 지정하려면 다음 단계를 따르세요.

소스 가져오기

디스크 컨테이너 1 섹션에서 가져올 첫 번째 VM 이미지 디스크 컨테이너 또는 스냅샷의 소스를 지정합니다.

a. 소스 - S3 버킷 또는 EBS 스냅샷일 수 있습니다.
b. 디스크의 S3 위치 선택 - 디스크 이미지가 저장되어 있는 Amazon S3의 위치를 입력합니다. 위치를 찾아보려면 S3 찾아보기를 선택합니다.
c. 디스크 컨테이너를 추가하려면 디스크 컨테이너 추가를 선택합니다.

2. IAM 역할

IAM 역할을 VM 가져오기 구성에 연결하려면 IAM 역할 드롭다운 목록에서 역할을 선택하거나 새로운 역할 생성을 선택하여 새 역할을 생성합니다. 새 역할을 생성하면 IAM 역할 콘솔 페이지가 별도의 탭에 열립니다.

3. 고급 설정 - 선택 사항

다음 설정은 선택 사항입니다. 이러한 설정을 사용하여 가져오기로 생성되는 기본 이미지에 대한 암호화, 라이선스, 태그 등을 구성할 수 있습니다.

기본 이미지 아키텍처

VM 가져오기 소스의 아키텍처를 지정하려면 아키텍처 목록에서 값을 선택합니다.

암호화

VM 디스크 이미지가 암호화된 경우 가져오기 프로세스에 사용할 키를 제공해야 합니다. 가져오기에 사용할 KMS 키를 지정하려면 암호화(KMS 키) 목록에서 값을 선택합니다. 목록에는 현재 리전에서 사용자 계정이 액세스할 수 있는 KMS 키가 포함되어 있습니다.

라이선스 관리

VM을 가져오면 가져오기 프로세스에서 VM OS를 자동으로 탐지하고 적절한 라이선스를 기본 이미지에 적용합니다. OS 플랫폼에 따라 라이선스 유형은 다음과 같습니다.

- 라이선스 포함 - 플랫폼에 적합한 AWS 라이선스가 기본 이미지에 적용됩니다.

- 기존 보유 라이선스 사용(BYOL) - 해당하는 경우 VM의 라이선스를 유지합니다.

로 생성된 라이선스 구성을 기본 이미지 AWS License Manager 에 연결하려면 라이선스 구성 이름 목록에서를 선택합니다. License Manager에 대한 자세한 내용은 작업 단원을 참조하십시오. AWS License Manager

> **Note**
>
> - 라이선스 구성은 기업 계약 조건에 기반한 라이선스 규칙을 포함합니다.
> - Linux는 BYOL 라이선스만 지원합니다.

태그(기본 이미지)

태그는 키-값 페어를 사용하여 검색 가능한 텍스트를 Image Builder 리소스에 할당합니다. 가져온 기본 이미지에 태그를 지정하려면 키 및 값 상자를 사용하여 키-값 페어를 입력합니다.

태그를 추가하려면 태그 추가를 선택합니다. 태그를 제거하려면 태그 제거를 선택합니다.

AWS CLI

디스크에서 AMI로 VM을 가져와서 바로 참조할 수 있는 Image Builder 이미지 리소스를 생성하려면 AWS CLI의 다음 단계를 따릅니다.

1. AWS CLI에서 Amazon EC2 VM 가져오기/내보내기 import-image 명령을 사용하여 VM 가져오기를 시작합니다. 명령 응답에 반환되는 작업 ID를 적어둡니다. 이 정보는 다음 단계에 필요합니다. 자세한 내용은 VM 가져오기/내보내기 사용 설명서에서 VM 가져오기/내보내기를 사용하여 VM을 이미지로 가져오기를 참조합니다.

2. CLI 입력 JSON 파일 생성

   에서 사용되는 Image Builder import-vm-image 명령을 간소화하기 위해 명령에 전달하려는 모든 가져오기 구성이 포함된 JSON 파일을 AWS CLI생성합니다.

> **Note**
>
> JSON 파일의 데이터 값에 대한 이름 지정 규칙은 Image Builder API 작업 요청 파라미터에 지정된 패턴을 따릅니다. API 작업 요청 파라미터를 검토하려면 EC2 Image Builder API 참조의 ImportVmImage 작업을 참조하세요.
> 데이터 값을 명령줄 파라미터로 제공하려면 옵션으로서 Image Builder import-vm-image 명령으로 AWS CLI 명령 참조에 지정된 파라미터 이름을 참조합니다.

다음은 이 예제에서 지정하는 파라미터의 요약입니다.

- name(문자열, 필수) - 가져오기에서 출력으로 생성할 이미지 빌더 이미지 리소스의 이름입니다.
- semanticVersion(문자열, 필수) – 특정 버전을 나타내기 위해 각 위치에 숫자 값이 있는 <major>.<minor>.<patch>. 형식으로 버전을 지정하는 출력 이미지의 시맨틱 버전. 예를 들어 `1.0.0`입니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.
- 설명(문자열) – 이미지 레시피에 대한 설명입니다.
- platform(문자열, 필수) - 가져온 VM의 운영 체제 플랫폼.
- vmImportTaskId(문자열, 필수) - Amazon EC2 VM 가져오기 프로세스의 `ImportTaskId`(AWS CLI). Image Builder는 가져오기 프로세스를 모니터링하여 생성한 AMI를 가져와 레시피에 즉시 사용할 수 있는 이미지 빌더 이미지 리소스를 빌드합니다.
- 태그(문자열 맵) - 태그는 가져오기 리소스에 연결되는 키-값 페어입니다. 키-값 페어는 최대 50개까지 허용됩니다.

Image Builder import-vm-image 명령에서 사용할 파일을 `import-vm-image.json`(으)로 저장합니다.

```
{
    "name": "example-request",
    "semanticVersion": "1.0.0",
    "description": "vm-import-test",
    "platform": "Linux",
    "vmImportTaskId": "import-ami-01ab234567890cd1e",
    "tags": {
     "Usage": "VMIE"
```

```
    }
}
```

3. 이미지 가져오기

   생성한 파일을 입력으로 사용하여 import-vm-image 명령을 실행합니다.

```
aws imagebuilder import-vm-image --cli-input-json file://import-vm-image.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

의 이미지 빌드에서 VM 디스크 배포 AWS CLI

AWS CLI에서 Image Builder 배포 구성을 사용하여 정규 이미지 빌드 프로세스의 일부로 대상 리전의 S3 버킷에 지원되는 VM 디스크 형식 파일을 배포하도록 설정할 수 있습니다. 자세한 내용은 에서 출력 VM 디스크에 대한 배포 설정 생성 AWS CLI 단원을 참조하십시오.

Image Builder를 사용하여 확인된 Windows ISO 디스크 이미지 가져오기

Windows 운영 체제 ISO 파일은 특정 버전의 Windows 운영 체제에 대한 전체 설치 패키지가 포함된 디스크 이미지 파일입니다. Microsoft는 웹 사이트에서 직접 또는 공인 리셀러를 통해 다운로드할 수 있는 공식 Windows 운영 체제 ISO 파일을 제공합니다. 잠재적인 맬웨어 또는 무단 버전을 방지하려면 신뢰할 수 있고 합법적인 소스에서 ISO 파일을 얻어야 합니다.

EC2 Image Builder는 `build-image-from-iso` 가져오기 워크플로를 사용하여 ISO 디스크 파일을 가져오고이 파일에서 보조 볼륨을 생성합니다. 구성이 완료되면 Image Builder는 가져오기에서 생성한 볼륨의 스냅샷을 생성하여 이를 사용하여 Amazon Machine Image(AMI)를 생성합니다.

ISO 디스크 이미지 가져오기에 지원되는 운영 체제

Image Builder는 다음과 같은 Windows 운영 체제 ISO 디스크 이미지를 지원합니다.

- Windows 11 Enterprise 버전 24H2
- Windows 11 Enterprise 버전 23H2
- Windows 11 Enterprise 버전 22H2

Image Builder는 다음 Windows 운영 체제 ISO 디스크 이미지를 지원하지 않습니다.

- 장기 서비스 채널(LTSC) 이미지
- Windows Media Creation Tool에서 생성된 ISO 디스크 이미지
- 평가 이미지

ISO 디스크 이미지를 가져오기 위한 사전 조건

ISO 디스크 이미지를 가져오려면 먼저 다음 사전 조건을 충족해야 합니다.

- 디스크 이미지의 운영 체제는 Image Builder가 지원하는 운영 체제여야 합니다. 지원되는 운영 체제 목록은 섹션을 참조하세요ISO 디스크 이미지 가져오기에 지원되는 운영 체제.
- ISO 이미지를 가져올 수 있도록 하려면 Microsoft 365 관리자 센터에서 다운로드합니다.
- 가져오기 프로세스를 실행하려면 먼저 가져오기가 실행되는 동일한 AWS 계정 및 AWS 리전 의 Amazon S3에 ISO 디스크 파일을 업로드해야 합니다.
- 파일 확장자는 가져오기 프로세스에서 대/소문자를 구분하며 여야 합니다`.ISO`. 파일 확장명이 소문자인 경우 다음 명령 중 하나를 실행하여 이름을 바꿀 수 있습니다.

  Command

  ```
  aws s3 cp s3://amzn-s3-demo-bucket/Win11_24H2_English.iso s3://amzn-s3-demo-
  bucket/Win11_24H2_English.ISO
  ```

  PowerShell

  ```
  Copy-S3Object -BucketName amzn-s3-demo-bucket -Key Win11_24H2_English.iso -
  DestinationKey Win11_24H2_English.ISO
  ```

- Microsoft 라이선스는 가져오기에 자동으로 포함되지 않습니다. 자체 라이선스(BYOL)를 가져와야 합니다. Microsoft 소프트웨어 라이선스에 대한 자세한 내용은 Amazon Web Services의 라이선스 및 Microsoft 자주 묻는 질문 페이지를 참조하세요.
- 가져오기 프로세스는 다음과 같이 두 개의 개별 IAM 역할을 사용합니다.

실행 역할

이 역할은 Image Builder가 AWS 서비스 사용자를 대신하여를 호출할 수 있는 권한을 부여합니다. 실행 역할에 필요한 권한이 포함된 AWSServiceRoleForImageBuilder 서비스 연결 역할을 지정하거나 자체 역할을 생성할 수 있습니다.

인스턴스 프로파일 역할

이 역할은 서비스가 EC2 인스턴스에서 수행하는 작업에 대한 권한을 부여합니다. 인프라 구성 리소스에서 인스턴스 프로파일 역할을 지정할 수 있습니다. 인스턴스 프로파일 역할에 다음 관리형 정책을 연결하여 가져오기 프로세스에 필요한 모든 권한이 있는지 확인합니다.

- EC2InstanceProfileForImageBuilder
- AmazonSSMManagedInstanceCore

자세한 내용은 Image Builder 인프라 구성 관리 단원을 참조하십시오.

Image Builder로 ISO 디스크 이미지 가져오기

가져오기 프로세스를 시작하기 전에 모든를 충족했는지 확인합니다사전 조건.

가져오기 프로세스는 이미지에 다음 소프트웨어 및 드라이버를 추가로 설치합니다.

- EC2Launch v2
- AWS Systems Manager 에이전트
- AWS NVMe 드라이버
- AWS ENA 네트워크 드라이버
- AWS PCI 직렬 드라이버
- EC2 Windows 유틸리티

Console

Image Builder 콘솔을 사용하여 ISO 디스크 이미지를 가져오려면 다음 단계를 따릅니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지를 선택합니다.
3. 가져오기 대화 상자를 열려면 이미지 가져오기를 선택합니다.
4. 다음 일반 정보를 입력합니다.

- 이미지의 고유한 이름을 지정합니다.
- 기본 이미지의 버전을 지정합니다. *major.minor.patch* 형식을 사용합니다.

5. 가져오기 유형: ISO 가져오기를 선택합니다.
6. 다음 ISO 가져오기 구성 세부 정보를 입력합니다. 작업을 마치면 이미지 가져오기를 선택합니다.

   - S3 URI - ISO 디스크 파일이 저장되는 위치를 입력합니다. 파일을 찾아보려면 S3 찾아보기를 선택합니다.
   - IAM 역할 - IAM 역할을 가져오기 구성과 연결하려면 IAM 역할 드롭다운 목록에서 역할을 선택하거나 새 역할 생성을 선택하여 새 역할을 생성합니다. 새 역할을 생성하면 IAM 역할 콘솔 페이지가 별도의 탭에 열립니다.

     AWSServiceRoleForImageBuilder 서비스 연결 역할을 지정하거나 서비스 액세스를 위한 사용자 지정 역할을 지정할 수 있습니다.
7. 선택적으로 Image Builder 이미지 리소스에 태그를 추가할 수 있습니다. 이렇게 해도 AMI에 태그가 추가되지 않습니다.
8. ISO 인프라 구성은 Image Builder가 가져오기 프로세스를 호스팅하기 위해 시작하는 인스턴스에 대한 설정을 정의합니다. Image Builder가 서비스 기본값을 기반으로 생성하는 인프라 구성을 사용하거나 기존 인프라 구성을 사용할 수 있습니다. 자세한 내용은 Image Builder 인프라 구성 관리 단원을 참조하십시오.

   새 인프라 구성을 생성하려면 인프라 구성 생성을 선택합니다. 그러면 별도의 탭에서 열립니다. 새 리소스 생성이 완료되면 가져오기 구성으로 돌아가 기존 인프라 구성 사용을 선택할 수 있습니다.
9. 가져오기 프로세스를 시작하려면 이미지 가져오기를 선택합니다.

가져오기가 완료되면 소유한 이미지 목록에 이미지가 나타납니다. 자세한 내용은 이미지 나열을 참조하세요.

AWS CLI

이 예제에서는 ISO 디스크 파일에서 이미지를 가져오고를 사용하여이 파일에서 AMI를 생성하는 방법을 보여줍니다 AWS CLI.

다음은 이 예제에서 지정하는 파라미터의 요약입니다.

- name(문자열, 필수) - 가져오기에서 출력으로 생성할 이미지 빌더 이미지 리소스의 이름입니다.

- semanticVersion(문자열, 필수) – 특정 버전을 나타내기 위해 각 위치에 숫자 값이 있는 <major>.<minor>.<patch>. 형식으로 버전을 지정하는 출력 이미지의 시맨틱 버전. 예를 들어 `1.0.0`입니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.
- 설명(문자열) – 이미지 레시피에 대한 설명입니다.
- executionRole(문자열) - Image Builder에 Microsoft ISO 파일에서 이미지를 가져오는 워크플로 작업을 수행할 수 있는 액세스 권한을 부여하는 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다. AWSServiceRoleForImageBuilder 서비스 연결 역할을 지정하거나 서비스 액세스를 위한 사용자 지정 역할을 지정할 수 있습니다.
- 플랫폼(문자열, 필수) - ISO 디스크 이미지의 운영 체제 플랫폼입니다. 유효한 값으로는 `Windows`가 포함됩니다.
- osVersion(문자열, 필수) - ISO 디스크 이미지의 운영 체제 버전입니다. 유효한 값으로는 `Microsoft Windows 11`가 포함됩니다.
- infrastructureConfigurationArn(문자열, 필수) - ISO 이미지가 빌드된 EC2 인스턴스를 시작하는 데 사용되는 인프라 구성 리소스의 Amazon 리소스 이름(ARN)입니다.
- uri(문자열, 필수) - Amazon S3에 저장된 ISO 디스크 파일의 URI입니다.

```
aws imagebuilder import-disk-image \
    --name "example-iso-disk-import" \
    --semantic-version "1.0.0" \
    --description "Import an ISO disk image" \
    --execution-role "AWSServiceRoleForImageBuilder" \
    --platform "Windows" \
    --os-version "Microsoft Windows 11" \
    --infrastructure-configuration-arn "arn:aws:imagebuilder:us-
east-1:111122223333:infrastructure-configuration/example-infrastructure-
configuration-123456789abc",
    --uri: "s3://amzn-s3-demo-source-bucket/examplefile.iso"
```

가져오기가 완료되면 소유한 이미지 목록에 이미지가 나타납니다. 자세한 내용은 이미지 나열을 참조하세요.

PowerShell

이 예제에서는 PowerShell을 사용하여 ISO 디스크 파일에서 이미지를 가져오고이 파일에서 AMI를 생성하는 방법을 보여줍니다.

다음은 이 예제에서 지정하는 파라미터의 요약입니다.

- name(문자열, 필수) - 가져오기에서 출력으로 생성할 이미지 빌더 이미지 리소스의 이름입니다.
- semanticVersion(문자열, 필수) – 특정 버전을 나타내기 위해 각 위치에 숫자 값이 있는 <major>.<minor>.<patch>. 형식으로 버전을 지정하는 출력 이미지의 시맨틱 버전. 예를 들어 `1.0.0`입니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.
- 설명(문자열) – 이미지 레시피에 대한 설명입니다.
- executionRole(문자열) - Image Builder에 Microsoft ISO 파일에서 이미지를 가져오는 워크플로 작업을 수행할 수 있는 액세스 권한을 부여하는 IAM 역할의 이름 또는 Amazon 리소스 이름(ARN)입니다. AWSServiceRoleForImageBuilder 서비스 연결 역할을 지정하거나 서비스 액세스를 위한 사용자 지정 역할을 지정할 수 있습니다.
- 플랫폼(문자열, 필수) - ISO 디스크 이미지의 운영 체제 플랫폼입니다. 유효한 값으로는 `Windows`가 포함됩니다.
- osVersion(문자열, 필수) - ISO 디스크 이미지의 운영 체제 버전입니다. 유효한 값으로는 `Microsoft Windows 11`가 포함됩니다.
- infrastructureConfigurationArn(문자열, 필수) - ISO 이미지가 빌드된 EC2 인스턴스를 시작하는 데 사용되는 인프라 구성 리소스의 Amazon 리소스 이름(ARN)입니다.
- uri(문자열, 필수) - Amazon S3에 저장된 ISO 디스크 파일의 URI입니다.

```
Import-EC2IBDiskImage `
    -Name "example-iso-disk-import" `
    -SemanticVersion "1.0.0" `
    -Description "Import an ISO disk image" `
    -ExecutionRole "AWSServiceRoleForImageBuilder" `
    -Platform "Windows" `
    -OsVersion "Microsoft Windows 11" `
    -InfrastructureConfigurationArn "arn:aws:imagebuilder:us-
east-1:111122223333:infrastructure-configuration/example-infrastructure-
configuration-123456789abc" `
    -Uri "s3://amzn-s3-demo-source-bucket/examplefile.ISO"
```

가져오기가 완료되면 소유한 이미지 목록에 이미지가 나타납니다. 자세한 내용은 이미지 나열 섹션을 참조하십시오.

Image Builder 이미지의 보안 결과 관리

Amazon Inspector로 보안 스캔을 활성화하면 계정의 시스템 이미지와 실행 중인 인스턴스를 지속적으로 스캔하여 운영 체제 및 프로그래밍 언어 취약성을 확인합니다. 활성화된 경우 보안 검색이 자동으로 수행되며 Image Builder는 새 이미지를 생성할 때 테스트 인스턴스를 통한 결과의 스냅샷을 저장할 수 있습니다. Amazon Inspector는 유료 서비스입니다.

Amazon Inspector는 소프트웨어 또는 네트워크 설정에서 취약성을 발견하면 다음 조치를 취합니다.

- 결과가 있었음을 알려줍니다.
- 결과의 심각도를 평가합니다. 심각도 등급은 발견의 우선 순위를 정하는 데 도움이 되는 취약성을 분류하며 다음 값을 포함합니다.
  - 분류되지 않음
  - 정보
  - 낮음
  - 중간
  - 높음
  - 심각
- 조사 결과에 대한 정보를 제공하고 자세한 내용을 확인할 수 있는 추가 리소스 링크를 제공합니다.
- 결과를 생성한 문제를 해결하는 데 도움이 되는 수정 지침을 제공합니다.

에서 Image Builder 이미지에 대한 보안 스캔 구성 AWS Management Console

계정에 대해 Amazon Inspector를 활성화한 경우, Amazon Inspector는 Image Builder가 시작하는 EC2 인스턴스를 자동으로 스캔하여 새 이미지를 구축하고 테스트합니다. 이러한 인스턴스는 구축 및 테스트 프로세스 동안 수명이 짧으며 일반적으로 해당 인스턴스가 종료되는 즉시 결과가 만료됩니다. 새 이미지에 대한 결과를 조사하고 수정하는 데 도움이 되도록 Image Builder는 Amazon Inspector가 빌드 프로세스 중에 테스트 인스턴스에서 확인한 모든 결과를 스냅샷으로 저장할 수 있습니다.

1단계: 계정에 대한 Amazon Inspector 보안 스캔 활성화

이미지 빌더 콘솔에서 계정에 대한 Amazon Inspector 보안 스캔을 활성화하려면 다음 단계를 따르십시오.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.

2. 탐색 창에서 보안 스캐닝 설정을 선택합니다. 그러면 보안 스캐닝 대화 상자가 열립니다.

   대화 상자에는 계정의 검사 상태가 표시됩니다. 계정에 대해 Amazon Inspector가 이미 활성화되어 있는 경우 상태가 활성화됨으로 표시됩니다.

3. 지침의 1단계와 2단계에 따라 Amazon Inspector 스캔을 활성화하세요.

> **Note**
>
> Amazon Inspector에서 요금이 부과됩니다. 자세한 내용은 Amazon Inspector 요금을 참조하세요.

파이프라인 스캔을 활성화한 경우, Image Builder는 새 이미지를 생성할 때 빌드 인스턴스에 대한 결과의 스냅샷을 생성합니다. 이렇게 하면 Image Builder가 빌드 인스턴스를 종료한 후 결과에 액세스할 수 있습니다.

2단계: 취약성 검색을 위해 스냅샷을 저장하도록 파이프라인 구성

파이프라인에 대한 취약성 검색 스냅샷을 구성하려면 다음 단계를 수행합니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지 파이프라인을 선택합니다.
3. 다음 방법 중 하나를 선택하여 파이프라인 세부 정보를 지정합니다.

   파이프라인 생성

   1. 이미지 파이프라인 페이지에서 이미지 파이프라인 생성을 선택합니다. 그러면 파이프라인 마법사에 파이프라인 세부 정보 지정 페이지가 열립니다.

   기존 이미지 파이프라인 업데이트

   1. 이미지 파이프라인 페이지에서 업데이트하려는 파이프라인 이름 링크를 선택합니다. 그러면 파이프라인 세부 정보 페이지가 열립니다.

      > **Note**
      >
      > 업데이트하려는 파이프라인 이름 옆의 확인란을 선택한 다음 세부 정보 보기를 선택할 수도 있습니다.

2. 파이프라인 세부 정보 페이지의 작업 메뉴에서 파이프라인 편집을 선택합니다. 이렇게 하면 파이프라인 편집 페이지로 이동합니다.

4. 파이프라인 마법사의 일반 섹션 또는 파이프라인 편집 페이지에서 보안 스캔 활성화 확인란을 선택합니다.

> **Note**
>
> 나중에 스냅샷을 끄려면 파이프라인을 편집하여 확인란의 선택을 취소할 수 있습니다. 이렇게 해도 Amazon Inspector의 계정 스캔이 비활성화되지는 않습니다. Amazon Inspector 스캐닝을 비활성화하려면 Amazon Inspector 사용 설명서의 Amazon Inspector 비활성화를 참조하세요.

에서 Image Builder 이미지의 보안 조사 결과 관리 AWS Management Console

보안 조사 결과 목록 페이지에는 적용할 수 있는 여러 필터를 기반으로 한 보기와 함께 리소스 결과에 대한 상위 수준 정보가 표시됩니다. 각 보기의 맨 위에는 보기를 변경할 수 있는 다음과 같은 옵션이 있습니다.

- 모든 보안 결과 - 이미지 빌더 콘솔의 탐색 창에서 보안 결과 페이지를 선택한 경우 기본 보기입니다.
- 취약성별 - 이 보기는 결과가 있는 계정 내 모든 이미지 리소스의 상위 수준 목록을 보여줍니다. 결과 ID는 결과에 대한 자세한 정보와 연결됩니다. 이 정보는 페이지의 오른쪽에 열리는 패널에 표시됩니다. 패널에 포함되는 정보는 다음과 같습니다.
  - 결과에 대한 자세한 설명.
  - 결과 세부 정보 탭. 이 탭에는 결과 개요, 영향을 받는 패키지, 요약 문제 해결 조언, 취약성 세부 정보 및 관련 취약성이 포함되어 있습니다. 취약성 ID는 국가 취약성 데이터베이스의 상세한 취약성 정보에 연결됩니다.
  - 점수 분석 탭. 이 탭에는 CVSS와 Amazon Inspector 점수를 나란히 비교한 내용이 포함되어 있으므로 해당하는 경우 Amazon Inspector에서 점수를 수정한 위치를 확인할 수 있습니다.
- 이미지 파이프라인별 — 이 보기는 계정의 각 이미지 파이프라인에 대한 결과 수를 보여줍니다. Image Builder는 중간 심각도 및 상위 결과 수와 모든 결과의 총계를 표시합니다. 목록의 모든 데이터는 다음과 같이 연결됩니다.
  - 이미지 파이프라인 이름 열은 지정된 이미지 파이프라인의 세부 정보 페이지로 연결됩니다.

- 심각도 수준 열 링크를 클릭하면 관련 이미지 파이프라인 이름 및 심각도 수준을 기준으로 필터링된 모든 보안 결과 보기가 열립니다.

검색 기준을 사용하여 결과를 구체화할 수도 있습니다.

- 이미지별 - 이 보기는 계정의 각 이미지 빌드에 대한 결과 수를 보여줍니다. Image Builder는 중간 심각도 및 상위 결과 수와 모든 결과의 총계를 표시합니다. 목록의 모든 데이터는 다음과 같이 연결됩니다.
  - 이미지 이름 열은 지정된 이미지 빌드의 이미지 세부 정보 페이지로 연결됩니다. 자세한 내용은 이미지 리소스 세부 정보 보기 단원을 참조하십시오.
  - 심각도 수준 열 링크를 클릭하면 관련 이미지 빌드 이름 및 심각도 수준을 기준으로 필터링된 모든 보안 결과 보기가 열립니다.

  검색 기준을 사용하여 결과를 구체화할 수도 있습니다.

이미지 빌더는 기본 모든 보안 결과 보기의 결과 목록 섹션에 다음과 같은 세부 정보를 표시합니다.

심각도

CVE 조사 결과의 심각도 수준 값은 다음과 같습니다.

- 분류되지 않음
- 정보
- 낮음
- 중간
- 높음
- 심각

결과 ID

Amazon Inspector가 빌드 인스턴스를 스캔할 때 이미지에 대해 감지한 CVE 조사 결과의 고유 식별자. ID는 보안 조사 결과 > 취약성별 페이지에 연결되어 있습니다.

이미지 ARN

결과 ID 열에 지정된 결과를 포함하는 이미지의 Amazon 리소스 이름(ARN).

파이프라인

이미지 ARN 열에 지정된 이미지를 빌드한 파이프라인.

설명

결과에 대한 설명.

인스펙터 점수

Amazon Inspector에서 CVE 조사 결과에 할당한 점수.

문제 해결

결과 개선을 위한 권장 조치 방침에 대한 세부 정보로 연결되는 링크.

게시 날짜

이 취약성이 공급업체 데이터베이스에 처음 추가된 날짜와 시간입니다.

Image Builder 리소스 정리

예상치 못한 요금이 부과되지 않도록 하려면 이 가이드의 예제에서 만든 리소스와 파이프라인을 정리하세요. Image Builder에서 리소스 삭제에 대한 자세한 내용은 오래되거나 사용되지 않는 Image Builder 리소스 삭제(을)를 참조하세요.

Image Builder 이미지의 수명 주기 정책 관리

사용자 지정 이미지를 생성하는 때 더 이상 사용되지 않게 되기 전에 해당 이미지를 폐기할 계획을 세우는 것이 중요합니다. Image Builder 파이프라인은 업데이트와 보안 패치를 자동으로 적용할 수 있습니다. 하지만 각 빌드는 이미지의 새 버전과 배포되는 모든 관련 리소스를 생성합니다. 이전 버전은 수동으로 삭제하거나 스크립트를 생성하여 작업을 수행할 때까지 계정에 남아 있습니다.

Image Builder 수명 주기 관리 정책을 사용하면 오래된 이미지와 관련 리소스의 사용 중단, 비활성화, 삭제 등의 프로세스를 자동화할 수 있습니다. 연결된 리소스에는 다른 AWS 계정조직 및 조직 단위(OUs)에 배포한 출력 이미지가 포함될 수 있습니다 AWS 리전. 수명 주기 프로세스의 각 단계를 수행하는 방법과 시기, 정책에 포함할 단계에 대한 규칙을 정의합니다.

자동화된 수명 주기 관리의 이점

수명 주기 관리 자동화의 전반적인 이점은 다음과 같습니다.

- 이미지와 관련 리소스를 자동으로 폐기하는 방법으로 사용자 지정 이미지의 수명 주기 관리를 간소화합니다.
- 오래된 이미지를 사용하여 새 인스턴스를 시작할 때 발생하는 규정 준수 위험을 방지하는 데 도움이 됩니다.
- 오래된 이미지를 제거하여 이미지 인벤토리를 최신 상태로 유지합니다.
- 삭제된 이미지의 관련 리소스를 선택적으로 제거하여 스토리지 및 데이터 전송 비용을 줄일 수 있습니다.

비용 절감 실현

EC2 Image Builder를 사용하여 사용자 지정 AMI 또는 컨테이너 이미지를 생성하는 데는 비용이 들지 않습니다. 그러나 프로세스에 사용되는 다른 서비스에는 표준 요금이 적용됩니다. 사용하지 않거나 오래된 이미지 및 관련 리소스를에서 제거하면 다음과 같은 방법으로 시간과 비용을 절감할 AWS 계정수 있습니다.

- 사용하지 않거나 오래된 이미지도 패치하지 않을 경우 기존 이미지를 패치하는 데 걸리는 시간을 줄입니다.
- 삭제하는 AMI 이미지 리소스의 경우 배포된 AMI와 관련 스냅샷도 제거하도록 선택할 수 있습니다. 이 방식을 사용하면 스냅샷 저장 비용을 절감할 수 있습니다.

- 삭제한 컨테이너 이미지 리소스의 경우 기본 리소스를 삭제하도록 선택할 수 있습니다. 이 방식을 사용하면 ECR 리포지토리에 저장된 Docker 이미지의 Amazon ECR 스토리지 비용과 데이터 전송 요금을 줄일 수 있습니다.

> **Note**
>
> Image Builder는 Auto Scaling 그룹 또는 시작 템플릿과 같이 발생 가능한 모든 다운스트림 종속성에 대한 잠재적 영향을 평가할 수 없습니다. 정책 조치를 구성할 때 이미지의 다운스트림 종속성을 고려해야 합니다.

내용

- Image Builder 이미지에 사용되는 수명 주기 관리 사전 조건
- Image Builder 이미지 리소스의 수명 주기 관리 정책 나열
- 수명 주기 정책 세부 정보 보기
- 수명 주기 정책 생성
- Image Builder 이미지 리소스의 수명 주기 관리 규칙 작동 방식

Image Builder 이미지에 사용되는 수명 주기 관리 사전 조건

이미지 리소스에 대한 EC2 Image Builder 수명 주기 관리 정책 및 규칙을 정의하려면 먼저 다음 사전 조건을 충족해야 합니다.

- Image Builder가 수명 주기 정책을 실행할 수 있는 권한을 부여하는 IAM 역할을 생성합니다. 역할을 생성하려면 Image Builder 수명 주기 관리를 위한 IAM 역할 생성 단원을 참조하십시오.
- 대상 계정에서 여러 계정에 분산된 관련 리소스의 IAM 역할을 생성합니다. 이 역할은 Image Builder가 대상 계정에서 관련 리소스에 대한 수명 주기 작업을 수행할 수 있는 권한을 부여합니다. 역할을 생성하려면 Image Builder 계정 간 수명 주기 관리를 위한 IAM 역할 생성 단원을 참조하십시오.

> **Note**
>
> 출력 AMI에 시작 권한을 부여한 경우에는 이 사전 조건이 적용되지 않습니다. 시작 권한이 있는 경우 사용자와 공유한 계정이 공유 AMI에서 시작된 인스턴스를 소유하지만 모든 AMI 리소스는 사용자의 계정에 남아 있습니다.

- 컨테이너 이미지의 경우 Image Builder가 리포지토리 `LifecycleExecutionAccess: EC2 Image Builder`에 저장된 컨테이너 이미지에 대해 수명 주기 작업을 실행할 수 있도록 액세스 권한을 부여하려면 ECR 리포지토리에 다음 태그를 추가해야 합니다.

Image Builder 수명 주기 관리를 위한 IAM 역할 생성

Image Builder에 수명 주기 정책을 실행할 권한을 부여하려면 먼저 Image Builder에서 수명 주기 작업에 사용하는 IAM 역할을 생성해야 합니다. 다음 단계에 따라 권한을 부여하는 서비스 역할을 생성합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다. 그러면 프로세스의 첫 번째 단계가 열립니다. 역할을 생성할 신뢰할 수 있는 엔터티를 선택합니다.
4. 신뢰할 수 있는 엔터티 유형에서 사용자 지정 신뢰 정책 옵션을 선택합니다.
5. 다음 JSON 신뢰 정책을 복사하고 사용자 지정 신뢰 정책 텍스트 영역에 붙여 넣어 샘플 텍스트를 바꿉니다. 이 신뢰 정책을 통해 Image Builder는 사용자가 생성한 역할을 맡아 수명 주기 작업을 실행할 수 있습니다.

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

6. 목록에서 EC2ImageBuilderLifecycleExecutionPolicy 관리형 정책을 선택하고 다음을 선택합니다. 그러면 이름, 검토 및 생성 페이지가 열립니다.

> **Tip**
>
> image 필터를 켜면 결과를 간소화할 수 있습니다.

7. 역할 이름을 입력합니다.
8. 설정을 검토한 후 역할 생성을 선택합니다.

Image Builder 계정 간 수명 주기 관리를 위한 IAM 역할 생성

Image Builder가 대상 계정에서 관련 리소스의 수명 주기 작업을 수행할 수 있는 권한을 부여하려면 먼저 Image Builder가 해당 계정에서 수명 주기 작업을 수행하는 데 사용하는 IAM 역할을 생성해야 합니다. 대상 계정에서 역할을 생성해야 합니다.

다음 단계에 따라 대상 계정에 권한을 부여하는 서비스 역할을 생성합니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 역할을 선택합니다.
3. 역할 생성을 선택합니다. 그러면 프로세스의 첫 번째 단계가 열립니다. 역할을 생성할 신뢰할 수 있는 엔터티를 선택합니다.
4. 신뢰할 수 있는 엔터티 유형에서 사용자 지정 신뢰 정책 옵션을 선택합니다.
5. 다음 JSON 신뢰 정책을 복사하고 사용자 지정 신뢰 정책 텍스트 영역에 붙여 넣어 샘플 텍스트를 바꿉니다. 이 신뢰 정책을 통해 Image Builder는 사용자가 생성한 역할을 맡아 수명 주기 작업을 실행할 수 있습니다.

> **Note**
>
> Image Builder가 대상 계정에서 이 역할을 사용하여 여러 계정에 분산된 관련 리소스의 작업을 수행하는 경우 대상 계정 소유자를 대신하여 작동합니다. 신뢰 정책aws:SourceAccount에서 로 AWS 계정 구성하는는 Image Builder가 해당 리소스를 배포한 계정입니다.

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

```
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "imagebuilder.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "444455556666"
                },
                "StringLike": {
                    "aws:SourceArn": "arn:*:imagebuilder:*:*:image/*/*/*"
                }
            }
        }
    ]
}
```

6. 목록에서 EC2ImageBuilderLifecycleExecutionPolicy 관리형 정책을 선택하고 다음을 선택합니다. 그러면 이름, 검토 및 생성 페이지가 열립니다.

> **Tip**
>
> image 필터를 켜면 결과를 간소화할 수 있습니다.

7. 역할 이름으로 Ec2ImageBuilderCrossAccountLifecycleAccess를 입력합니다.

> **Important**
>
> Ec2ImageBuilderCrossAccountLifecycleAccess는 이 역할의 이름이어야 합니다.

8. 설정을 검토한 후 역할 생성을 선택합니다.

Image Builder 이미지 리소스의 수명 주기 관리 정책 나열

의 수명 주기 정책 목록 페이지에서 또는 Image Builder API AWS Management Console, SDKs 또는의 명령 또는 작업을 사용하여 키 세부 정보 열을 포함하는 이미지 수명 주기 관리 정책 목록을 가져올 수 있습니다 AWS CLI.

다음 방법 중 하나를 사용하여 AWS 계정에서 Image Builder 이미지 수명 주기 정책 리소스를 나열할 수 있습니다. API 작업에 대한 내용은 EC2 Image Builder API 참조의 ListLifecyclePolicies를 참조하세요. 연결된 SDK 요청의 경우, 같은 페이지의 참고 항목 링크를 참조하세요.

AWS Management Console

기존 정책에 대한 다음 세부 정보가 콘솔에 표시됩니다. 원하는 열을 선택하여 결과의 정렬 순서를 변경할 수 있습니다. 정책 목록은 처음에 정책 이름을 기준으로 정렬됩니다. 현재 정렬 순서의 열 이름은 굵게 표시됩니다.

결과 페이지가 두 페이지 이상인 경우 패널 오른쪽 상단의 페이징 화살표가 활성화됩니다. 검색 창에서 정책 이름, 정책 상태, 출력 이미지 유형, 이미지 리소스 ARN을 기준으로 결과를 필터링할 수 있습니다.

- 정책 이름-정책의 이름입니다.
- 정책 상태-정책의 활성 또는 비활성 상태를 나타냅니다.
- 유형-Image Builder가 새로운 이미지 버전(AMI 또는 컨테이너 이미지)을 생성할 때 배포한 출력 이미지 유형입니다.
- 마지막 실행 날짜-수명 주기 정책이 마지막으로 실행된 시간입니다.
- 생성 날짜-수명 주기 정책을 생성한 시점의 타임스탬프입니다.
- ARN-수명 주기 정책 리소스의 Amazon 리소스 이름(ARN)입니다.

에 수명 주기 정책을 나열하려면 다음 단계를 AWS Management Console따릅니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 수명 주기 정책을 선택합니다. 이것은 계정의 이미지 수명 주기 정책 목록을 보여줍니다.

사용 가능한 작업

수명 주기 정책 목록 페이지에서 수명 주기 정책에 대해 다음 작업을 수행할 수도 있습니다.

새로운 이미지 수명 주기 정책을 만들려면 수명 주기 정책 생성을 선택합니다. 정책을 생성하는 방법에 대한 자세한 내용은 수명 주기 정책 생성 섹션을 참조하세요.

다음 모든 작업의 경우 먼저 정책을 선택해야 합니다. 정책을 선택하려면 정책 이름 옆에 있는 확인란을 선택합니다.

- 정책을 끄거나 켜려면 작업 메뉴에서 정책 비활성화 또는 정책 활성화를 선택합니다.
- 정책을 변경하려면 작업 메뉴에서 정책 편집을 선택합니다.
- 정책을 삭제하려면 작업 메뉴에서 정책 삭제를 선택합니다.
- 선택한 정책을 기준 설정에 사용하는 새 정책을 만들려면 작업 메뉴에서 정책 복제를 선택합니다.

AWS CLI

다음 명령 예제에서는를 사용하여 특정 AWS CLI 에 대한 이미지 수명 주기 정책을 나열하는 방법을 보여줍니다 AWS 리전. 이 명령과 함께 사용할 수 있는 파라미터 및 옵션에 대한 자세한 내용은 AWS CLI 명령 참조의 list-lifecycle-policies 명령을 참조하세요.

예:

```
aws imagebuilder list-lifecycle-policies \
--region us-west-1
```

출력:

```
{
    "lifecyclePolicySummaryList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy1",
            "name": "sample-lifecycle-policy1",
            "status": "DISABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-11-07T14:57:01.603000-08:00",
            "tags": {}
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy2",
            "name": "sample-lifecycle-policy2",
            "status": "ENABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-09-06T10:43:21.436000-07:00",
            "dateLastRun": "2023-11-13T04:43:46.106000-08:00",
```

```
            "tags": {}
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy3",
            "name": "sample-lifecycle-policy3",
            "status": "ENABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-10-19T15:16:40.046000-07:00",
            "dateUpdated": "2023-10-21T20:07:15.958000-07:00",
            "dateLastRun": "2023-11-12T09:27:45.830000-08:00"
}]}
```

> **Note**
>
> 기본값을 사용하려면 `--region` 파라미터 없이이 명령을 AWS 리전실행합니다.

수명 주기 정책 세부 정보 보기

Image Builder 콘솔의 수명 주기 정책 세부 정보 페이지에는 추가 정보가 탭으로 그룹화된 요약 섹션이 있습니다. 페이지 제목이 정책의 이름입니다.

Image Builder 콘솔의 수명 주기 정책 세부 정보 페이지에서 특정 수명 주기 정책의 세부 정보를 확인할 수 있습니다. Image Builder API, SDK 또는 AWS CLI 를 통해 명령 또는 작업을 사용하여 정책 세부 정보를 가져올 수도 있습니다.

내용

- Image Builder 콘솔에서 수명 주기 정책 세부 정보 보기

Image Builder 콘솔에서 수명 주기 정책 세부 정보 보기

Image Builder 콘솔의 이미지 세부 정보 페이지에는 추가 정보가 탭으로 그룹화된 요약 섹션이 있습니다. 페이지 제목은 이미지를 만든 레시피의 이름 및 빌드 버전입니다.

콘솔 세부 정보 섹션 및 탭

- 요약 섹션
- 규칙 탭

- 범위 탭
- RunLog 탭

요약 섹션

요약 섹션은 페이지 너비를 포함하며 다음과 같은 세부 정보를 포함합니다. 이러한 세부 정보는 항상 표시됩니다.

정책 상태

정책의 활성 또는 비활성 상태를 나타냅니다.

유형

Image Builder가 새로운 이미지 버전(AMI 또는 컨테이너 이미지)을 생성할 때 배포한 출력 이미지 유형입니다.

생성 날짜

수명 주기 정책을 생성한 시점의 타임스탬프입니다.

수정 날짜

수명 주기 정책이 마지막으로 업데이트된 시간입니다.

마지막 실행 날짜

수명 주기 정책이 마지막으로 실행된 시간입니다.

IAM 역할

Image Builder가 수명 주기 작업을 수행할 때 사용하는 IAM 역할입니다.

ARN

수명 주기 정책 리소스의 Amazon 리소스 이름(ARN)입니다.

설명

입력한 경우 수명 주기 정책에 대한 설명입니다.

규칙 탭

규칙 탭에는 현재 보고 있는 정책에 대해 구성한 수명 주기 규칙이 표시됩니다. 탭에는 다음 항목이 포함됩니다.

- 이름-규칙의 이름입니다. 이 이름들은 구성할 수 있는 정책 작업에 따라 정적입니다.
  - `Deprecation rule`
  - `Disable rule`
  - `Deletion rule`
- 규칙-규칙에 대해 구성된 작업에 대한 간략한 설명입니다.
- 규칙 조건-관련 리소스 처리를 위한 구성, 규칙 예외, 보존 설정(해당하는 경우)을 나열합니다.

규칙 구성에 대한 자세한 내용은 수명 주기 규칙 작동 방식 섹션을 참조하세요.

범위 탭

범위 탭에는 현재 보고 있는 정책에 대해 구성한 리소스 선택 기준이 표시됩니다. 탭에는 다음 항목이 포함됩니다.

- 필터: ## ##-범위를 정의하는 데 사용한 필터 유형입니다. 필터 유형은 다음 중 하나일 수 있습니다.
  - `recipes`-수명 주기 정책이 적용되는 이미지를 만드는 데 사용한 레시피입니다.
  - `tags`-Image Builder에서 수명 주기 정책이 적용되는 이미지 리소스를 선택하는 데 사용하는 태그 세트입니다.
- 검색 창-목록을 이름별로 필터링하여 탭에 표시되는 결과를 간소화할 수 있습니다.
- 이름-각 행에는 필터 기준에 맞게 구성한 이름 또는 태그가 포함됩니다.
- 버전-레시피 필터를 구성한 경우 Image Builder는 레시피 버전을 표시합니다.

RunLog 탭

구성된 리소스에 대해 정책을 실행할 때마다 Image Builder가 런타임 세부 정보를 저장합니다. 테이블의 각 행은 단일 런타임 인스턴스를 나타냅니다. 탭에는 다음 항목이 포함됩니다.

- 실행 ID-수명 주기 정책 런타임 인스턴스를 식별합니다.
- 실행 상태-정책 작업이 현재 실행 중인지, 성공적으로 실행되었는지, 실패했는지 또는 취소되었는지를 보고하는 런타임 상태입니다.
- 영향을 받는 리소스-런타임 인스턴스가 수명 주기 작업에 사용할 이미지 리소스를 식별했는지 여부를 나타냅니다.
- 시작 날짜-런타임 인스턴스가 시작된 시점의 타임스탬프입니다.
- 종료 날짜-런타임 인스턴스가 종료된 시점의 타임스탬프입니다.

수명 주기 정책 생성

새 EC2 Image Builder 수명 주기 정책을 생성할 때 구성은 정책의 대상이 되는 이미지 종류에 따라 달라집니다. AMI 이미지 리소스와 컨테이너 이미지 리소스의 수명 주기 정책을 생성하는 API 작업(CreateLifecyclePolicy)은 동일합니다. 하지만 이미지 리소스와 관련 리소스의 구성은 다릅니다. 이 섹션에서는 두 가지 모두의 수명 주기 관리 정책을 생성하는 방법을 설명합니다.

> **Note**
>
> 수명 주기 정책을 생성하기 전에 먼저 모든 사전 조건을 충족해야 합니다.

Image Builder AMI 이미지 리소스의 수명 주기 관리 정책 생성

다음 방법 중 하나를 사용하여 AWS Management Console 또는를 사용하여 AMI 이미지 수명 주기 정책을 생성할 수 있습니다 AWS CLI. CreateLifecyclePolicy API 작업을 사용할 수도 있습니다. 관련 SDK 요청의 경우, EC2 Image Builder API 참조에서 해당 명령에 대한 관련 참고 항목 링크를 참조할 수도 있습니다.

AWS Management Console

에서 AMI 이미지 리소스에 대한 수명 주기 정책을 생성하려면 다음 단계를 AWS Management Console따릅니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 수명 주기 정책을 선택합니다.
3. 수명 주기 정책 생성을 선택합니다.
4. 다음 절차에 설명된 정책 설정을 구성합니다.
5. 설정을 구성한 후 정책 생성을 선택하여 수명 주기 정책을 생성합니다.

정책의 일반 설정을 구성합니다.

1. 정책 유형에서 AMI 옵션을 선택합니다.
2. 정책 이름을 입력합니다.
3. 선택적으로 수명 주기 정책에 대한 설명을 입력합니다.
4. 기본적으로 활성화는 켜져 있습니다. 기본 설정은 수명 주기 정책을 활성화하고 일정에 즉시 추가합니다. 처음에 비활성화된 정책을 만들려면 활성화를 끄면 됩니다.

5. 수명 주기 정책 권한을 위해 생성한 IAM 역할을 선택합니다. 이 역할을 아직 생성하지 않았다면 사전 조건 섹션을 참조하세요.

정책의 규칙 범위를 구성합니다.

이 섹션에서는 사용하는 필터 유형에 따라 수명 주기 정책의 리소스 선택을 구성합니다.

1. 필터 유형: 레시피-이미지 리소스를 생성한 레시피를 기반으로 이미지 리소스에 수명 주기 규칙을 적용하려면 정책에 사용할 레시피 버전을 최대 50개까지 선택합니다.
2. 필터 유형: 태그-리소스 태그를 기반으로 이미지 리소스에 수명 주기 규칙을 적용하려면 일치하는 정책이 적용되는 최대 50개의 키 값 페어 목록을 입력합니다.

다음 수명 주기 규칙 중 하나 이상을 켜고 수명 주기 정책에서 선택한 리소스에 적용합니다. 정책이 실행될 때 리소스가 둘 이상의 수명 주기 규칙에서 일치하는 경우 Image Builder는 1) 사용 중지, 2) 비활성화, 3) 삭제 순으로 규칙 작업을 수행합니다.

사용 중지 규칙

Image Builder 이미지 리소스 상태를 `Deprecated`로 설정합니다. Image Builder 파이프라인은 더 이상 사용되지 않는 이미지에 대해 계속 실행됩니다. 새 인스턴스를 시작하는 기능에 영향을 주지 않으면서 관련 AMI의 사용 중지 시간을 선택적으로 설정할 수 있습니다.

- 단위 수-이미지 리소스가 생성된 후 `Deprecated`로 표시되기 전에 경과해야 하는 기간의 정수 값을 지정합니다.
- 단위-사용할 시간 범위를 선택합니다. 범위는 `Days`, `Weeks`, `Months` 또는 `Years`일 수 있습니다.
- AMI 사용 중단-관련 Amazon EC2 AMI를 사용 중단 날짜로 표시하려면 확인란을 선택합니다. AMI는 계속 사용할 수 있으며 AMI에서 새 인스턴스를 시작할 수 있습니다.

규칙 비활성화

Image Builder 이미지 리소스 상태를 `Disabled`로 설정합니다. 이렇게 하면 이 이미지에 대해 Image Builder 파이프라인이 실행되지 않습니다. 필요에 따라 연결된 AMI를 비활성화하여 새 인스턴스가 시작되지 않게 할 수 있습니다.

- 단위 수-이미지 리소스가 생성된 후 `Disabled`로 표시되기 전에 경과해야 하는 기간의 정수 값을 지정합니다.
- 단위-사용할 시간 범위를 선택합니다. 범위는 `Days`, `Weeks`, `Months` 또는 `Years`일 수 있습니다.

- AMI 비활성화-관련 Amazon EC2 AMI를 비활성화하려면 확인란을 선택합니다. 더 이상 AMI를 사용하거나 AMI에서 새 인스턴스를 시작할 수 없습니다.

규칙 삭제

기간별 또는 개수별로 이미지 리소스를 삭제합니다. 요구 사항을 충족하는 임곗값을 정의합니다. Image Builder 이미지 리소스가 임곗값을 초과하면 해당 리소스는 제거됩니다. 선택적으로 관련 AMI를 등록 취소하거나 해당 AMI의 스냅샷을 삭제할 수 있습니다. 임곗값을 초과하여 유지할 리소스의 태그를 지정할 수도 있습니다.

기간별 삭제 규칙을 구성하면 Image Builder는 사용자가 구성한 기간이 지나면 이미지 리소스를 삭제합니다. 예를 들어 6개월 후에 이미지 리소스를 삭제합니다. 개수별로 구성하는 경우 Image Builder는 사용자가 지정한 가장 최근의 이미지 수 또는 해당 수에 최대한 가깝게 유지하고 이전 버전은 삭제합니다.

- 기간 기준
  - 단위 수-이미지 리소스가 생성된 후 삭제되기 전에 경과해야 하는 기간의 정수 값을 지정합니다.
  - 단위-사용할 시간 범위를 선택합니다. 범위는 `Days`, `Weeks`, `Months` 또는 `Years`일 수 있습니다.
  - 레시피당 이미지 하나 이상 유지-이 규칙이 적용되는 각 레시피 버전에 사용 가능한 최신 이미지 리소스를 유지하려면 확인란을 선택합니다.

  개수 기준

  - 이미지 수-각 레시피 버전에 유지할 최근 이미지 리소스 수의 정수 값을 지정합니다.
- AMI 등록 취소-관련 Amazon EC2 AMI를 등록 취소하려면 확인란을 선택합니다. 더 이상 AMI를 사용하거나 AMI에서 새 인스턴스를 시작할 수 없습니다.
- 관련 태그가 있는 이미지, AMI 및 스냅샷 유지-보관할 이미지 리소스의 태그 목록을 입력하려면 확인란을 선택합니다. 태그는 이미지 리소스와 Amazon EC2 AMI에 적용됩니다. 최대 50개의 키 값 페어를 입력할 수 있습니다.

태그(선택 사항)

수명 주기 정책에 태그를 추가합니다.

AWS CLI

Image Builder 수명 주기 정책을 새로 만들려면 AWS CLI에서 create-lifecycle-policy 명령을 사용할 수 있습니다.

Image Builder 컨테이너 이미지 리소스의 수명 주기 관리 정책 생성

다음 방법 중 하나를 사용하여 AWS Management Console 또는를 사용하여 컨테이너 이미지 수명 주기 정책을 생성할 수 있습니다 AWS CLI. CreateLifecyclePolicy API 작업을 사용할 수도 있습니다. 관련 SDK 요청의 경우, EC2 Image Builder API 참조에서 해당 명령에 대한 관련 참고 항목 링크를 참조할 수도 있습니다.

AWS Management Console

에서 컨테이너 이미지 리소스에 대한 수명 주기 정책을 생성하려면 다음 단계를 AWS Management Console따릅니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 수명 주기 정책을 선택합니다.
3. 수명 주기 정책 생성을 선택합니다.
4. 다음 절차에 설명된 정책 설정을 구성합니다.
5. 설정을 구성한 후 정책 생성을 선택하여 수명 주기 정책을 생성합니다.

정책 구성: 일반 설정

정책의 일반 설정을 구성합니다.

1. 정책 유형에서 AMI 옵션을 선택합니다.
2. 정책 이름을 입력합니다.
3. 선택적으로 수명 주기 정책에 대한 설명을 입력합니다.
4. 기본적으로 활성화는 켜져 있습니다. 기본 설정은 수명 주기 정책을 활성화하고 일정에 즉시 추가합니다. 처음에 비활성화된 정책을 만들려면 활성화를 끄면 됩니다.
5. 수명 주기 정책 권한을 위해 생성한 IAM 역할을 선택합니다. 이 역할을 아직 생성하지 않았다면 사전 조건 섹션을 참조하세요.

정책의 규칙 범위를 구성합니다.

이 섹션에서는 사용하는 필터 유형에 따라 수명 주기 정책의 리소스 선택을 구성합니다.

1. 필터 유형: 레시피-이미지 리소스를 생성한 레시피를 기반으로 이미지 리소스에 수명 주기 규칙을 적용하려면 정책에 사용할 레시피 버전을 최대 50개까지 선택합니다.
2. 필터 유형: 태그-리소스 태그를 기반으로 이미지 리소스에 수명 주기 규칙을 적용하려면 일치하는 정책이 적용되는 최대 50개의 키 값 페어 목록을 입력합니다.

규칙 삭제

컨테이너 이미지의 경우 이 규칙은 Image Builder 컨테이너 이미지 리소스를 삭제합니다. ECR 리포지토리에 배포된 도커 이미지를 선택적으로 제거하여 새 컨테이너 실행에 사용되지 않도록 할 수도 있습니다.

기간별 삭제 규칙을 구성하면 Image Builder는 사용자가 구성한 기간이 지나면 이미지 리소스를 삭제합니다. 예를 들어 6개월 후에 이미지 리소스를 삭제합니다. 개수별로 구성하는 경우 Image Builder는 사용자가 지정한 가장 최근의 이미지 수 또는 해당 수에 최대한 가깝게 유지하고 이전 버전은 삭제합니다.

- 기간 기준
  - 단위 수-이미지 리소스가 생성된 후 삭제되기 전에 경과해야 하는 기간의 정수 값을 지정합니다.
  - 단위-사용할 시간 범위를 선택합니다. 범위는 `Days`, `Weeks`, `Months` 또는 `Years`일 수 있습니다.
  - 이미지 하나 이상 유지-이 규칙이 적용되는 각 레시피 버전에 사용 가능한 최신 이미지 리소스만 유지하려면 확인란을 선택합니다.

  개수 기준

  - 이미지 수-각 레시피 버전에 유지할 최근 이미지 리소스 수의 정수 값을 지정합니다.
- ECR 컨테이너 이미지 삭제-ECR 리포지토리에 저장된 관련 컨테이너 이미지를 삭제하려면 확인란을 선택합니다. 더 이상 컨테이너 이미지를 기반으로 사용하여 새 이미지를 만들거나 새 컨테이너를 실행할 수 없습니다.
- 관련 태그가 있는 이미지 유지-보관할 이미지 리소스의 태그 목록을 입력하려면 확인란을 선택합니다.

태그(선택 사항)

수명 주기 정책에 태그를 추가합니다.

AWS CLI

Image Builder 수명 주기 정책을 새로 만들려면 AWS CLI에서 create-lifecycle-policy 명령을 사용할 수 있습니다.

Image Builder 이미지 리소스의 수명 주기 관리 규칙 작동 방식

이미지 수명 주기 정책은 사용자가 정의한 수명 주기 규칙을 사용하여 전체 리소스 관리 전략을 구현합니다. 규칙을 정의하면 사용 가능한 이미지를 최신 상태로 유지하고 출력 AMI를 위한 스냅샷 스토리지, 컨테이너 이미지의 ECR 리포지토리 스토리지 및 데이터 전송 요금과 같은 기본 인프라 비용을 최소화하는 데 도움이 됩니다.

정책에 대해 다음 유형의 규칙을 구성할 수 있습니다.

사용 중지 규칙

Image Builder 이미지 리소스 상태를 `Deprecated`로 설정합니다. Image Builder 파이프라인은 더 이상 사용되지 않는 이미지에 대해 계속 실행됩니다. 새 인스턴스를 시작하는 기능에 영향을 주지 않으면서 관련 AMI의 사용 중지 시간을 선택적으로 설정할 수 있습니다.

AMI가 사용 중단되면 일반 검색에서 무시됩니다. 예를 들어에서 Amazon EC2 describe-images 명령을 실행하면 결과 집합에서 더 이상 사용되지 않는 AMIs가 반환 AWS CLI되지 않습니다. 그러나 AMI ID를 통해 사용 중단된 AMI를 계속 찾을 수 있습니다.

컨테이너 이미지에는 이 규칙을 사용할 수 없습니다.

규칙 비활성화

Image Builder 이미지 리소스 상태를 `Disabled`로 설정합니다. 이렇게 하면 이 이미지에 대해 Image Builder 파이프라인이 실행되지 않습니다. 필요에 따라 연결된 AMI를 비활성화하여 새 인스턴스가 시작되지 않게 할 수 있습니다.

비활성화한 AMI는 비공개 상태가 되고 새 인스턴스를 시작하는 데 사용할 수 없습니다. AMI를 계정, 조직 또는 조직 구성 단위와 공유한 경우, 해당 AMI가 비공개로 전환되면 AMI에 대한 액세스 권한을 잃게 됩니다.

컨테이너 이미지에는 이 규칙을 사용할 수 없습니다.

규칙 삭제

기간별 또는 개수별로 이미지 리소스를 삭제합니다. 요구 사항을 충족하는 임곗값을 정의합니다. Image Builder 이미지 리소스가 임곗값을 초과하면 해당 리소스는 제거됩니다. 선택적으로 관련 AMI를 등록 취소하거나 해당 AMI의 스냅샷을 삭제할 수 있습니다. 임곗값을 초과하여 유지할 리소스의 태그를 지정할 수도 있습니다.

컨테이너 이미지의 경우 이 규칙은 Image Builder 컨테이너 이미지 리소스를 삭제합니다. ECR 리포지토리에 배포된 컨테이너 이미지를 선택적으로 제거하여 새 컨테이너 실행에 사용되지 않도록 할 수도 있습니다.

내용

- AMI 수명 주기 제외 규칙
- 정책의 수명 주기 관리 규칙 세부 정보 보기

AMI 수명 주기 제외 규칙

다음 제외 규칙은 AMI의 수명 주기 규칙에 대한 예외를 정의합니다. 제외 규칙으로 지정된 기준을 충족하는 AMI는 수명 주기 작업에서 제외됩니다. AWS Management Console에서는 제외 규칙을 사용할 수 없습니다.

다음 용어는 `LifecyclePolicyDetailExclusionRules` 데이터 유형의 API 표기법을 사용합니다.

제외 규칙

amis

다음 목록에 표시된 `LifecyclePolicyDetailExclusionRulesAmis`의 설정을 포함합니다.

tagMap

모든 유형의 리소스에 대해 수명 주기 작업을 건너뛰는 태그 목록을 최대 50개 제공할 수 있습니다.

다음 용어는 `LifecyclePolicyDetailExclusionRulesAmis` 데이터 유형의 API 표기법을 사용합니다.

AMI 제외 규칙

isPublic

퍼블릭 AMI를 수명 주기 작업에서 제외할지 여부를 구성합니다.

lastLaunched

수명 주기 작업에서 최신 리소스를 제외하도록 Image Builder의 구성 세부 정보를 지정합니다.

리전

수명 주기 작업에서 제외 AWS 리전 되는를 구성합니다.

sharedAccounts

수명 주기 작업에서 제외 AWS 계정 되는 리소스를 지정합니다.

tagMap

해당 태그가 있는 AMI의 수명 주기 작업에서 제외해야 하는 태그를 나열합니다.

정책의 수명 주기 관리 규칙 세부 정보 보기

규칙은 Image Builder 이미지 리소스에 대해 생성한 수명 주기 관리 정책에 정의됩니다. 콘솔의 수명 주기 정책 세부 정보 페이지에는 해당 정책에 대해 구성한 규칙의 세부 정보를 보여주는 규칙 탭이 있습니다.

에서 정책 세부 정보를 가져오려면 get-lifecycle-policy 명령을 실행할 AWS CLI수 있습니다. 응답의 정책 세부 정보에는 정책에 대해 정의한 작업(규칙) 목록이 포함되며, 여기에는 구성된 모든 설정이 포함됩니다.

Image Builder를 사용하여 사용자 지정 이미지 구성

구성 리소스는 이미지 파이프라인을 구성하는 구성 요소이자 해당 파이프라인이 생성하는 이미지입니다. 이 장에서는 인프라 구성 및 배포 설정과 함께 구성 요소, 레시피, 이미지를 비롯한 이미지 빌더 리소스를 생성, 유지 관리 및 공유하는 방법을 다룹니다.

> **Note**
>
> 이미지 빌더 리소스를 관리하는 데 도움이 되도록 각 리소스에 고유한 메타데이터를 할당할 수 있습니다. 태그를 사용하여 AWS 리소스를 용도, 소유자, 환경 등으로 다양하게 분류할 수 있습니다. 이는 동일한 유형의 리소스가 많을 때 유용합니다. 지정한 태그를 기반으로 특정 리소스를 더 쉽게 식별할 수 있습니다.
> 에서 Image Builder 명령을 사용하여 리소스에 태그를 지정하는 방법에 대한 자세한 내용은이 가이드의 리소스 태깅 섹션을 AWS CLI참조하세요.

내용

- Image Builder에서 레시피 관리
- Image Builder 인프라 구성 관리
- Image Builder 배포 설정 관리

Image Builder에서 레시피 관리

EC2 Image Builder 레시피는 새 이미지를 생성하기 위한 출발점으로 사용할 기본 이미지를 정의합니다. 새 이미지를 정의하는 동시에 이미지를 사용자 정의하고 모든 것이 예상대로 작동하는지 확인하기 위해 사용자가 추가하는 구성 요소 집합도 정의합니다. Image Builder는 각 구성 요소에 대한 자동 버전 선택 기능을 제공합니다. 기본적으로 레시피에 최대 20개의 구성 요소를 적용할 수 있습니다. 여기에는 빌드 구성 요소와 테스트 구성 요소가 모두 포함됩니다.

레시피를 생성한 후에는 수정하거나 이를 대체할 수 없습니다. 레시피를 생성한 후 구성 요소를 업데이트하려면 새 레시피 또는 레시피 버전을 생성해야 합니다. 기존 레시피에 언제든지 태그를 적용할 수 있습니다. 에서 Image Builder 명령을 사용하여 리소스에 태그를 지정하는 방법에 대한 자세한 내용은 이 가이드의 리소스 태깅 섹션을 AWS CLI참조하세요.

> **Tip**
>
> 레시피에서 Amazon 관리형 구성 요소를 사용하거나 자체 사용자 지정 구성 요소를 개발할 수 있습니다. 자세한 내용은 Image Builder 이미지에 대한 사용자 지정 구성 요소 개발 단원을 참조하십시오. 출력 AMIs를 생성하는 이미지 레시피의 경우 AWS Marketplace 이미지 제품 및 구성 요소를 사용할 수도 있습니다. AWS Marketplace 제품과의 통합에 대한 자세한 내용은 단원을 참조하십시오AWS Marketplace Image Builder의 통합.

이 섹션에서는 레시피를 나열하고, 보고, 생성하는 방법을 설명합니다.

내용

- 이미지 레시피 세부 정보 나열 및 보기
- 컨테이너 레시피 세부 정보 나열 및 보기
- 이미지 레시피의 새 버전 생성
- 컨테이너 레시피의 새 버전 생성
- 리소스 정리

이미지 레시피 세부 정보 나열 및 보기

이 섹션에서는 EC2 Image Builder 이미지 레시피에 대한 정보를 찾고 세부 정보를 볼 수 있는 다양한 방법을 설명합니다.

이미지 레시피 세부 정보

- 콘솔에서 이미지 레시피 나열
- 에서 이미지 레시피 나열 AWS CLI
- 콘솔에서 이미지 레시피 세부 정보 보기
- 에서 이미지 레시피 세부 정보 가져오기 AWS CLI
- 에서 이미지 레시피 정책 세부 정보 가져오기 AWS CLI

콘솔에서 이미지 레시피 나열

Image Builder 콘솔에서 사용자 계정으로 생성된 이미지 레시피 목록을 보려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.

2. 탐색 창에서 이미지 레시피를 선택합니다. 계정에서 생성된 이미지 레시피 목록이 표시됩니다.
3. 세부 정보를 보거나 새 레시피 버전을 만들려면 레시피 이름 링크를 선택합니다. 레시피의 세부 정보 보기가 열립니다.

> **Note**
>
> 또한 레시피 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

에서 이미지 레시피 나열 AWS CLI

다음 예제에서는 AWS CLI(을)를 사용하여 모든 이미지 레시피를 나열하는 방법을 보여 줍니다.

```
aws imagebuilder list-image-recipes
```

콘솔에서 이미지 레시피 세부 정보 보기

Image Builder 콘솔을 사용해서 특정 이미지 레시피의 세부 정보를 보려면 검토할 이미지 레시피를 선택하고 콘솔에서 이미지 레시피 나열에 설명된 단계를 사용하십시오.

레시피 세부 정보 페이지에서 다음을 수행할 수 있습니다.

- 레시피를 삭제합니다. Image Builder에서 리소스 삭제에 대한 자세한 내용은 오래되거나 사용되지 않는 Image Builder 리소스 삭제(을)를 참조하세요.
- 새로운 버전을 생성합니다.
- 레시피에서 파이프라인을 생성하십시오. 이 레시피에서 파이프라인 생성을 선택하면 파이프라인 마법사로 이동합니다. 파이프라인 마법사를 이용한 Image Builder 파이프라인을 생성에 대한 자세한 내용은 자습서: Image Builder 콘솔 마법사에서 출력 AMI를 사용하여 이미지 파이프라인 생성(을)를 참조하세요.

> **Note**
>
> 기존 레시피로 파이프라인을 생성하는 경우 새 레시피를 생성하는 옵션을 사용할 수 없습니다.

에서 이미지 레시피 세부 정보 가져오기 AWS CLI

다음 예제는 imagebuilder CLI 명령을 사용하여 Amazon 리소스 이름(ARN)을 지정하여 이미지 레시피의 세부 정보를 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-image-recipe --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2020.12.03
```

에서 이미지 레시피 정책 세부 정보 가져오기 AWS CLI

다음 예제는 imagebuilder CLI 명령을 사용하여 ARN을 지정하여 이미지 레시피 정책의 세부 정보를 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-image-recipe-policy --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2020.12.03
```

컨테이너 레시피 세부 정보 나열 및 보기

이 섹션에서는 EC2 Image Builder 컨테이너 레시피에 대한 정보를 찾고 세부 정보를 볼 수 있는 방법을 설명합니다.

컨테이너 레시피 세부 정보

- 콘솔에 컨테이너 레시피 나열
- AWS CLI(을)를 사용하여 컨테이너 레시피 나열
- 콘솔에서 컨테이너 레시피 세부 정보 보기
- AWS CLI(을)를 사용하여 컨테이너 레시피 세부 정보 가져오기
- 를 사용하여 컨테이너 레시피 정책 세부 정보 가져오기 AWS CLI

콘솔에 컨테이너 레시피 나열

Image Builder 콘솔에서 사용자 계정으로 생성된 컨테이너 레시피 목록을 보려면 다음 단계를 따르십시오.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 컨테이너 레시피를 선택합니다. 계정에서 생성된 컨테이너 레시피 목록이 표시됩니다.

3. 세부 정보를 보거나 새 레시피 버전을 만들려면 레시피 이름 링크를 선택합니다. 레시피의 세부 정보 보기가 열립니다.

> **Note**
>
> 또한 레시피 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

AWS CLI(을)를 사용하여 컨테이너 레시피 나열

다음 예제에서는 AWS CLI(을)를 사용하여 모든 컨테이너 레시피를 나열하는 방법을 보여 줍니다.

```
aws imagebuilder list-container-recipes
```

콘솔에서 컨테이너 레시피 세부 정보 보기

Image Builder 콘솔에서 특정 컨테이너 레시피의 세부 정보를 보려면 검토할 컨테이너 레시피를 선택하고 콘솔에 컨테이너 레시피 나열에 설명된 단계를 사용하십시오.

레시피 세부 정보 페이지에서 다음을 수행할 수 있습니다.

- 레시피를 삭제합니다. Image Builder에서 리소스를 삭제하는 방법에 대한 자세한 내용은 오래되거나 사용되지 않는 Image Builder 리소스 삭제(을)를 참조하십시오.
- 새로운 버전을 생성합니다.
- 레시피에서 파이프라인을 생성하십시오. 이 레시피에서 파이프라인 생성을 선택하면 파이프라인 마법사로 이동합니다. 파이프라인 마법사를 이용해 Image Builder 파이프라인을 생성하는 방법에 대한 자세한 내용은 자습서: Image Builder 콘솔 마법사에서 출력 AMI를 사용하여 이미지 파이프라인 생성(을)를 참조하십시오.

> **Note**
>
> 기존 레시피로 파이프라인을 생성하는 경우 새 레시피를 생성하는 옵션을 사용할 수 없습니다.

AWS CLI(을)를 사용하여 컨테이너 레시피 세부 정보 가져오기

다음 예제는 imagebuilder CLI 명령을 사용하여 ARN을 지정하여 컨테이너 레시피의 세부 정보를 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-container-recipe --container-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:container-recipe/my-example-recipe/2020.12.03
```

를 사용하여 컨테이너 레시피 정책 세부 정보 가져오기 AWS CLI

다음 예제는 imagebuilder CLI 명령을 사용하여 ARN을 지정하여 컨테이너 레시피 정책의 세부 정보를 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-container-recipe-policy --container-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-example-
recipe/2020.12.03
```

이미지 레시피의 새 버전 생성

이 섹션에서는 새 버전의 이미지 레시피를 만드는 방법을 설명합니다.

내용

- 콘솔에서 새 이미지 레시피 버전 생성
- 를 사용하여 이미지 레시피 생성 AWS CLI
- 콘솔에서 VM을 기본 이미지로 가져옵니다.

콘솔에서 새 이미지 레시피 버전 생성

새 레시피 버전을 만드는 것은 새 레시피를 만드는 것과 거의 같습니다. 차이점은 대부분의 경우 기본 레시피와 일치하도록 특정 세부 정보가 미리 선택된다는 것입니다. 다음 목록은 새 레시피 생성과 기존 레시피의 새 버전 생성 간의 차이점을 설명합니다.

새 버전의 기본 레시피 세부 정보

- 이름 – 편집할 수 없습니다.
- 버전 – 필수입니다. 이 기본 세부 정보는 현재 버전이나 다른 종류의 시퀀스로 미리 채워져 있지 않습니다. <major>.<minor>.<patch> 형식으로 생성하려는 버전 번호를 입력합니다. 버전이 이미 있는 경우 오류가 발생합니다.
- 이미지 선택 옵션 - 미리 선택되어 있지만 편집할 수 있습니다. 기본 이미지의 소스 선택을 변경하면 선택한 원본 옵션에 따라 달라지는 기타 세부 정보가 손실될 수 있습니다.

  기본 이미지 선택과 관련된 세부 정보를 보려면 선택 항목과 일치하는 탭을 선택하십시오.

Managed image

- 이미지 운영 체제(OS) - 편집할 수 없습니다.
- 이미지 이름 - 기존 레시피에 대해 선택한 기본 이미지 조합을 기반으로 미리 선택됩니다. 하지만 이미지 선택 옵션을 변경하면 미리 선택한 이미지 이름을 잃게 됩니다.
- 자동 버전 관리 옵션 - 기본 레시피와 일치하지 않습니다. 이 이미지 옵션의 기본값은 선택한 OS 버전 사용 옵션입니다.

> **Important**
>
> 시맨틱 버전 관리를 사용하여 파이프라인 빌드를 시작하는 경우 이 값을 사용 가능한 최신 OS 버전 사용으로 변경해야 합니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

AWS Marketplace image

- 구독 -이 탭은 열려 있어야 하며 기본 레시피와 일치하도록에서 구독한 이미지를 미리 선택해야 AWS Marketplace 합니다. 레시피가 기본 이미지로 사용하는 이미지를 변경하면 선택한 원본 이미지에 의존하는 기타 세부 정보가 손실될 수 있습니다.

AWS Marketplace 제품에 대한 자세한 내용은 AWS Marketplace 구매자 안내서의 제품 구매를 참조하세요.

Custom AMI

AMI 소스(필수) - 기본 이미지로 사용할 AMI ID 또는 AMI ID가 포함된 AWS Systems Manager (SSM) 파라미터 스토어 파라미터를 입력합니다. SSM 에이전트는 선택한 AMI에 사전 설치되어 있어야 합니다.

- AMI ID -이 설정은 원래 항목으로 미리 채워지지 않습니다. 기본 이미지의 AMI ID를 입력합니다. 예시: *ami-1234567890abcdef1*.
- SSM 파라미터 - 기본 이미지의 AMI ID가 포함된 SSM 파라미터 스토어 파라미터의 이름 또는 ARN을 입력합니다. 예: */ib/test/param* 또는 `arn:aws:ssm:`*us-east-1*`:`*111122223333*`:parameter`*/ib/test/param*.

- 인스턴스 구성 - 설정이 미리 선택되어 있지만 편집할 수 있습니다.
  - Systems Manager 에이전트 - 이 확인란을 선택하거나 선택 취소하여 새 이미지에 Systems Manager 에이전트 설치를 제어할 수 있습니다. 새 이미지에 Systems Manager 에이전트를 포함하도록 기본적으로 확인란이 선택되어 있지 않습니다. 최종 이미지에서 Systems Manager 에이전트를 제거하려면 에이전트가 AMI에 포함되지 않도록 확인란을 선택합니다.

- 사용자 데이터 – 이 영역을 사용하여 빌드 인스턴스를 시작할 때 실행할 명령 또는 명령 스크립트를 제공합니다. 하지만 이 값은 Systems Manager가 설치되었는지 확인하기 위해 Image Builder가 추가했을 수 있는 모든 명령을 대체합니다. 이러한 명령에는 새 이미지를 생성하기 전에 Image Builder가 Linux 이미지에 대해 일반적으로 실행하는 정리 스크립트가 포함됩니다.

  Image Builder가 인스턴스를 시작하면 구성 요소 실행이 시작되기 전에 Cloud-init 단계에서 사용자 데이터 스크립트가 실행됩니다. 이 단계는 인스턴스의 파일에 로깅됩니다`var/log/cloud-init.log`.

  > **Note**
  >
  > - 사용자 데이터를 입력하는 경우 기본 이미지에 Systems Manager 에이전트가 사전 설치되어 있거나 사용자 데이터에 설치를 포함해야 합니다.
  > - Linux 이미지의 경우 사용자 데이터 스크립트에 `perform_cleanup`(이)라는 이름이 지정된 빈 파일을 만드는 명령을 포함하여 정리 단계를 실행해야 합니다. Image Builder는 이 파일을 탐지하고 새 이미지를 만들기 전에 정리 스크립트를 실행합니다. 자세한 내용과 샘플 스크립트는 Image Builder의 보안 모범 사례 섹션을 참조하세요.

- 작업 디렉토리 - 미리 선택되어 있지만 편집할 수 있습니다.
- 구성 요소 - 레시피에 이미 포함된 구성 요소는 각 구성 요소 목록(빌드 및 테스트) 끝에 있는 선택된 구성 요소 섹션에 표시됩니다. 필요에 맞게 선택한 구성 요소를 제거하거나 재정렬할 수 있습니다.

  CIS 강화 구성 요소는 Image Builder 레시피의 표준 구성 요소 순서 지정 규칙을 따르지 않습니다. 출력 이미지에 대해 벤치마크 테스트가 실행되도록 하기 위해 CIS 강화 구성 요소는 항상 마지막에 실행됩니다.

  > **Note**
  >
  > 빌드 및 테스트 구성 요소 목록에는 구성 요소 소유자 유형에 따라 사용 가능한 구성 요소가 표시됩니다. 구성 요소를 추가하려면 빌드 구성 요소 추가를 선택하고 적용되는 소유권 필터를 선택합니다. 예를 들어 AWS Marketplace 제품과 연결된 빌드 구성 요소를 추가하려면를 선택합니다`AWS Marketplace`. 그러면 구성 요소를 나열 AWS Marketplace 하는 콘솔 인터페이스 오른쪽에 선택 패널이 열립니다.
  > CIS 구성 요소에서를 선택합니다`Third party managed`.

  선택한 구성 요소에 대해 다음 설정을 구성할 수 있습니다.

- 버전 관리 옵션 - 미리 선택되어 있지만 변경할 수 있습니다. 이미지 빌드가 항상 최신 버전의 구성 요소를 사용하도록 하려면 사용 가능한 최신 구성 요소 버전 사용 옵션을 선택하는 것이 좋습니다. 레시피에서 특정 구성 요소 버전을 사용해야 하는 경우 구성 요소 버전 지정을 선택하고 나타나는 구성 요소 버전 상자에 버전을 입력할 수 있습니다.
- 입력 파라미터 - 구성 요소가 받아들이는 입력 파라미터를 표시합니다. 값은 이전 버전의 레시피 값으로 미리 채워집니다. 이 레시피에서 이 구성 요소를 처음 사용하고 입력 파라미터에 대해 기본값이 정의된 경우 기본값이 값 상자에 회색으로 표시된 텍스트와 함께 나타납니다. 값을 입력하지 않으면 Image Builder에서 기본값을 사용합니다.

  입력 파라미터가 필요하지만 구성 요소에 기본값이 정의되어 있지 않은 경우 값을 제공해야 합니다. 누락된 필수 파라미터가 있고 기본값이 정의되지 않은 경우 Image Builder는 레시피 버전을 생성하지 않습니다.

  > **Important**
  >
  > 구성 요소 파라미터는 일반 텍스트 값이며 AWS CloudTrail에 기록됩니다. AWS Secrets Manager 또는 AWS Systems Manager 파라미터 스토어를 사용하여 보안 암호를 저장하는 것이 좋습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager란 무엇입니까?를 참조하십시오. AWS Systems Manager Parameter Store에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Parameter Store 섹션을 참조하십시오.

  버전 관리 옵션 또는 입력 파라미터 설정을 확장하려면 설정 이름 옆에 있는 화살표를 선택하면 됩니다. 선택한 모든 구성 요소의 모든 설정을 확장하려면 모두 확장 스위치를 끄고 켤 수 있습니다.
- 스토리지(볼륨) — 미리 채워져 있습니다. 루트 볼륨 디바이스 이름, 스냅샷 및 IOPS 선택은 편집할 수 없습니다. 하지만 크기와 같은 나머지 설정은 모두 변경할 수 있습니다. 새 볼륨을 추가하고 새 볼륨 또는 기존 볼륨을 암호화할 수도 있습니다.

  Image Builder가 소스 리전((빌드가 실행되는 리전)의 사용자 계정으로 생성한 이미지의 볼륨을 암호화하려면 이미지 레시피의 스토리지 볼륨 암호화를 사용해야 합니다. 빌드의 배포 단계에서 실행되는 암호화는 다른 계정이나 리전에 배포되는 이미지에만 적용됩니다.

  > **Note**
  >
  > 볼륨에 암호화를 사용하는 경우 키가 루트 볼륨에 사용된 것과 같더라도 각 볼륨에 대해 키를 개별적으로 선택해야 합니다.

새 이미지 레시피 버전을 만들려면

1. 레시피 세부정보 페이지 상단에서 새 버전 생성을 선택합니다. 그러면 이미지 레시피 만들기 페이지로 이동합니다.
2. 새 버전을 만들려면 변경한 다음 레시피 생성을 선택합니다.

   최종 이미지에는 AWS Marketplace 이미지 제품 및 구성 요소의 제품 코드가 최대 4개까지 포함될 수 있습니다. 선택한 기본 이미지와 구성 요소에 4개 이상의 제품 코드가 포함된 경우 레시피를 생성하려고 할 때 Image Builder가 오류를 반환합니다.

이미지 파이프라인을 생성할 때 이미지 레시피를 생성하는 방법에 대한 자세한 내용은 이 가이드의 시작하기 섹션의 2단계: 레시피 선택(을)를 참조하십시오.

를 사용하여 이미지 레시피 생성 AWS CLI

에서 Image Builder `create-image-recipe` 명령을 사용하여 이미지 레시피를 생성하려면 다음 단계를 AWS CLI따릅니다.

사전 조건

이 섹션의 Image Builder 명령을 실행하여 AWS CLI에서 이미지 레시피를 만들려면 먼저 레시피에서 사용하는 구성 요소를 만들어야 합니다. 다음 단계의 이미지 레시피 예제는 이 가이드의 에서 사용자 지정 구성 요소 생성 AWS CLI 섹션에서 만든 예제 구성 요소를 참조합니다.

구성 요소를 만든 후 또는 기존 구성 요소를 사용하는 경우, 레시피에 포함하려는 ARN을 기록해 두십시오.

1. CLI 입력 JSON 파일 생성

   create-image-recipe 명령의 모든 입력을 인라인 명령 파라미터로 제공할 수 있습니다. 하지만 결과로 생성되는 명령은 상당히 길 수 있습니다. 명령을 간소화하기 위해 모든 레시피 설정이 포함된 JSON 파일을 대신 제공할 수 있습니다.

   > **Note**
   >
   > JSON 파일의 데이터 값에 대한 이름 지정 규칙은 Image Builder API 작업 요청 파라미터에 지정된 패턴을 따릅니다. API 작업 요청 파라미터를 검토하려면 EC2 Image Builder API 참조의 CreateImageRecipe 명령을 참조하세요.

데이터 값을 명령줄 파라미터로 제공하려면 AWS CLI 명령 참조에 지정된 파라미터 이름을 참조하십시오.

다음은 이러한 예제에서 지정하는 파라미터를 요약한 것입니다.

- 이름(문자열, 필수) — 이미지 레시피의 이름입니다.
- 설명(문자열) – 이미지 레시피에 대한 설명입니다.
- parentImage(문자열, 필수) — 이미지 레시피가 사용자 지정 이미지의 기본으로 사용하는 이미지입니다. 다음 옵션 중 하나를 사용하여 상위 이미지를 지정할 수 있습니다.
  - AMI ID
  - Image Builder 이미지 리소스 ARN
  - AWS Systems Manager (SSM) Parameter Store 파라미터 앞에 접두사가 붙고 `ssm:`파라미터 이름 또는 ARN이 옵니다.
  - AWS Marketplace 제품 ID

  > **Note**
  >
  > Linux 및 macOS 예제에서는 Image Builder AMI를 사용하고 Windows 예제에서는 ARN을 사용합니다.

- semanticVersion(문자열, 필수) — 이미지 레시피의 시맨틱 버전으로, <major>.<minor>.<patch> 형식으로 표현되며 각 위치에 숫자 값이 있어 특정 버전을 나타냅니다. 예를 들어, 값이 `1.0.0`(이)가 될 수 있습니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.
- 구성 요소(배열, 필수) - `ComponentConfiguration` 객체 배열을 포함합니다. 빌드 구성 요소를 하나 이상 지정해야 합니다.

  > **Note**
  >
  > Image Builder는 레시피에 지정한 순서대로 구성 요소를 설치합니다. 하지만 CIS 하드닝 구성 요소는 출력 이미지에 대해 벤치마크 테스트가 실행되도록 하기 위해 항상 마지막에 실행됩니다.

  - componentARN(문자열, 필수) – 구성 요소 ARN입니다.

> **Tip**
>
> 예제 중 하나를 사용하여 고유한 이미지 레시피를 만들려면 예제 ARN을 레시피에 사용 중인 구성 요소의 ARN으로 바꿔야 합니다.

- 파라미터(객체 배열) - `ComponentParameter` 객체 배열을 포함합니다. 입력 파라미터가 필요하지만 구성 요소에 기본값이 정의되어 있지 않은 경우 값을 제공해야 합니다. 누락된 필수 파라미터가 있고 기본값이 정의되지 않은 경우 Image Builder는 레시피 버전을 생성하지 않습니다.

  > **Important**
  >
  > 구성 요소 파라미터는 일반 텍스트 값이며 AWS CloudTrail에 기록됩니다. AWS Secrets Manager 또는 AWS Systems Manager 파라미터 스토어를 사용하여 보안 암호를 저장하는 것이 좋습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager란 무엇입니까?를 참조하십시오. AWS Systems Manager Parameter Store에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Parameter Store 섹션을 참조하십시오.

  - 이름(문자열, 필수) — 설정할 구성 요소 파라미터의 이름입니다.
  - 값(문자열 배열, 필수) - 명명된 구성 요소 파라미터의 값을 설정하는 문자열 배열을 포함합니다. 구성 요소에 기본값이 정의되어 있고 다른 값이 제공되지 않은 경우는 기본값을 AWSTOE 사용합니다.
- additionalInstanceConfiguration(객체) - 빌드 인스턴스에 대한 추가 설정을 지정하고 스크립트를 시작합니다.
  - systemsManagerAgent(객체) - 빌드 인스턴스의 Systems Manager 에이전트에 대한 설정을 포함합니다.
    - uninstallAfterBuild(부울) - 새 AMI를 생성하기 전에 최종 빌드 이미지에서 Systems Manager 에이전트를 제거할지 여부를 제어합니다. 이 옵션을 `true`(으)로 설정하면 에이전트가 최종 이미지에서 제거됩니다. 이 옵션을 `false`(으)로 설정하면 에이전트가 그대로 남아 있으므로 새 AMI에 포함됩니다. 기본값은 `false`입니다.

> **Note**
>
> `uninstallAfterBuild` 속성이 JSON 파일에 포함되어 있지 않고 다음 조건에 해당하는 경우 Image Builder는 최종 이미지에서 Systems Manager 에이전트를 제거하여 AMI에서 사용할 수 없도록 합니다.
>
> - 이 `userDataOverride` 파일이 비어 있거나 JSON 파일에서 생략되었습니다.
> - Image Builder는 기본 이미지에 에이전트가 사전 설치되어 있지 않은 운영 체제의 빌드 인스턴스에 Systems Manager 에이전트를 자동으로 설치했습니다.

- userDataOverride(문자열) — 빌드 인스턴스를 시작할 때 실행할 명령 또는 명령 스크립트를 제공합니다.

> **Note**
>
> 사용자 데이터는 항상 base 64로 인코딩됩니다. 예를 들어, 다음 명령은 `IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg==`(으)로 인코딩됩니다.
>
> ```
> #!/bin/bash
> mkdir -p /var/bb/
> touch /var
> ```
>
> Linux 예제에서는 이 인코딩된 값을 사용합니다.

Linux

다음 예제의 기본 이미지(`parentImage` 속성)는 AMI입니다. AMI를 사용하는 경우 AMI에 액세스할 수 있어야 하며 AMI는 소스 지역(Image Builder가 명령을 실행하는 동일한 지역)에 있어야 합니다. 파일을 create-image-recipe(으)로 저장하고 `create-image-recipe.json` 명령에 사용합니다.

```
{
"name": "BB Ubuntu Image recipe",
"description": "Hello World image recipe for Linux.",
"parentImage": "ami-1234567890abcdef1",
"semanticVersion": "1.0.0",
"components": [
```

```
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/bb$"
 }
],
"additionalInstanceConfiguration": {
 "systemsManagerAgent": {
   "uninstallAfterBuild": true
 },
 "userDataOverride": "IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg=="
}
}
```

Windows

다음 예제는 최신 버전의 Windows Server 2016 영어 풀 베이스 이미지를 참조합니다. 이 예제의 ARN은 지정한 의미 체계 버전 필터를 기반으로 최신 이미지를 참조합니다arn:aws:imagebuilder:*us-west-2*:aws:image/*windows-server-2016-english-full-base-x86/x.x.x*.

```
{
"name": "MyBasicRecipe",
"description": "This example image recipe creates a Windows 2016 image.",
"parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/x.x.x",
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/my-example-component/2019.12.02/1"
 },
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/my-imported-component/1.0.0/1"
 }
]
}
```

> **Note**
>
> Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

macOS

다음 예제의 기본 이미지(parentImage 속성)는 AMI입니다. AMI를 사용하는 경우 AMI에 액세스할 수 있어야 하며 AMI는 소스 지역(Image Builder가 명령을 실행하는 동일한 지역)에 있어야 합니다. 파일을 create-image-recipe(으)로 저장하고 create-image-recipe.json 명령에 사용합니다.

```
{
"name": "macOS Catalina Image recipe",
"description": "Hello World image recipe for macOS.",
"parentImage": "ami-1234567890abcdef1",
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-
west-2:111122223333:component/catalina$"
 }
],
"additionalInstanceConfiguration": {
 "systemsManagerAgent": {
   "uninstallAfterBuild": true
 },
 "userDataOverride": "IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg=="
}
}
```

2. 레시피 생성

다음 명령을 사용하여 레시피를 생성합니다. 이전 단계에서 생성한 JSON 파일의 이름을 --cli-input-json 파라미터에 입력합니다.

```
aws imagebuilder create-image-recipe --cli-input-json file://create-image-
recipe.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 file:// 표기법을 포함시켜야 합니다.

- JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

최종 이미지에는 AWS Marketplace 이미지 제품 및 구성 요소의 제품 코드가 최대 4개까지 포함될 수 있습니다. 선택한 기본 이미지와 구성 요소에 4개 이상의 제품 코드가 포함된 경우 Image Builder는 `create-image-recipe` 명령을 실행할 때 오류를 반환합니다.

콘솔에서 VM을 기본 이미지로 가져옵니다.

이 섹션에서는 가상 머신(VM)을 이미지 레시피의 기본 이미지로 가져오는 방법을 중점적으로 다룹니다. 레시피 또는 레시피 버전 생성과 관련된 다른 단계는 여기서 다루지 않습니다. Image Builder 콘솔에서 파이프라인 생성 마법사를 사용하여 새 이미지 레시피를 생성하는 추가 단계는 파이프라인 마법사: AMI 생성(을)를 참조하십시오. 새 이미지 레시피 또는 레시피 버전을 생성하기 위한 추가 단계는 이미지 레시피의 새 버전 생성(을)를 참조하십시오.

Image Builder 콘솔에서 VM을 이미지 레시피의 기본 이미지로 가져오려면 다음 단계와 기타 필수 단계를 따라 레시피 또는 레시피 버전을 생성하십시오.

1. 기본 이미지의 이미지 선택 섹션에서 기본 이미지 가져오기 옵션을 선택합니다.
2. 평소와 같이 이미지 운영 체제 (OS)와 OS 버전을 선택합니다.

VM 가져오기 구성

가상화 환경에서 VM을 내보내는 경우 이 프로세스는 VM 환경, 설정 및 데이터의 스냅샷 역할을 하는 하나 이상의 디스크 컨테이너 파일 세트를 만듭니다. 이러한 파일을 사용하여 VM을 이미지 레시피의 기본 이미지로 가져올 수 있습니다. Image Builder에서 VM을 가져오는 작업에 대한 자세한 내용은 VM 이미지 가져오기 및 내보내기(을)를 참조하십시오.

가져오기 소스의 위치를 지정하려면 다음 단계를 따르세요.

소스 가져오기

디스크 컨테이너 1 섹션에서 가져올 첫 번째 VM 이미지 디스크 컨테이너 또는 스냅샷의 소스를 지정합니다.

1. 소스 - S3 버킷 또는 EBS 스냅샷일 수 있습니다.

2. 디스크의 S3 위치 선택 - 디스크 이미지가 저장되어 있는 Amazon S3의 위치를 입력합니다. 위치를 찾아보려면 S3 찾아보기를 선택합니다.
3. 디스크 컨테이너를 추가하려면 디스크 컨테이너 추가를 선택합니다.

IAM 역할

IAM 역할을 VM 가져오기 구성에 연결하려면 IAM 역할 드롭다운 목록에서 역할을 선택하거나 새로운 역할 생성을 선택하여 새 역할을 생성합니다. 새 역할을 생성하면 IAM 역할 콘솔 페이지가 별도의 탭에 열립니다.

고급 설정 - 선택 사항

다음 설정은 선택 사항입니다. 이러한 설정을 사용하여 가져오기로 생성되는 기본 이미지에 대한 암호화, 라이선스, 태그 등을 구성할 수 있습니다.

일반

1. 기본 이미지에 고유한 이름을 지정합니다. 값을 입력하지 않으면 기본 이미지가 레시피 이름을 상속합니다.
2. 기본 이미지의 버전을 지정합니다. *<major>.<minor>.<patch>* 형식을 사용합니다. 값을 입력하지 않으면 기본 이미지가 레시피 버전을 상속합니다.
3. 기본 이미지에 대한 설명을 입력할 수도 있습니다.

기본 이미지 아키텍처

VM 가져오기 소스의 아키텍처를 지정하려면 아키텍처 목록에서 값을 선택합니다.

암호화

VM 디스크 이미지가 암호화된 경우 가져오기 프로세스에 사용할 키를 제공해야 합니다. 가져오기에 AWS KMS key 대한를 지정하려면 암호화(KMS 키) 목록에서 값을 선택합니다. 목록에는 현재 리전에서 사용자 계정이 액세스할 수 있는 KMS 키가 포함되어 있습니다.

라이선스 관리

VM을 가져오면 가져오기 프로세스에서 VM OS를 자동으로 탐지하고 적절한 라이선스를 기본 이미지에 적용합니다. OS 플랫폼에 따라 라이선스 유형은 다음과 같습니다.

- 라이선스 포함 - 플랫폼에 적합한 AWS 라이선스가 기본 이미지에 적용됩니다.
- 기존 보유 라이선스 사용(BYOL) - 해당하는 경우 VM의 라이선스를 유지합니다.

로 생성된 라이선스 구성을 기본 이미지 AWS License Manager 에 연결하려면 라이선스 구성 이름 목록에서를 선택합니다. License Manager에 대한 자세한 내용은 작업 단원을 참조하십시오. AWS License Manager

> **Note**
>
> - 라이선스 구성은 기업 계약 조건에 기반한 라이선스 규칙을 포함합니다.
> - Linux는 BYOL 라이선스만 지원합니다.

태그(기본 이미지)

태그는 키-값 페어를 사용하여 검색 가능한 텍스트를 Image Builder 리소스에 할당합니다. 가져온 기본 이미지에 태그를 지정하려면 키 및 값 상자에 키-값 페어를 입력합니다.

태그를 추가하려면 태그 추가를 선택합니다. 태그를 제거하려면 태그 제거를 선택합니다.

컨테이너 레시피의 새 버전 생성

이 섹션에서는 컨테이너 레시피의 새 버전을 생성하는 방법을 보여줍니다.

내용

- 콘솔을 사용하여 새 컨테이너 레시피 버전 생성
- AWS CLI(을)를 사용하여 컨테이너 레시피 생성

콘솔을 사용하여 새 컨테이너 레시피 버전 생성

컨테이너 레시피의 새 버전을 만드는 것은 새 레시피를 만드는 것과 거의 같습니다. 차이점은 대부분의 경우 기본 레시피와 일치하도록 특정 세부 정보가 미리 선택된다는 것입니다. 다음 목록은 새 레시피 생성과 기존 레시피의 새 버전 생성 간의 차이점을 설명합니다.

레시피 세부 정보

- 이름 - 편집할 수 없습니다.
- 버전 – 필수입니다. 이 세부 정보는 현재 버전이나 다른 종류의 시퀀스로 미리 채워져 있지 않습니다. 만들려는 버전 번호를 major.minor.patch 형식으로 입력합니다. 버전이 이미 있는 경우 오류가 발생합니다.

기본 이미지

- 이미지 옵션 선택 - 미리 선택되었지만 편집할 수 있습니다. 기본 이미지의 소스 선택을 변경하면 선택한 원본 옵션에 따라 달라지는 기타 세부 정보가 손실될 수 있습니다.

  Docker 컨테이너 이미지의 경우 DockerHub에서 호스팅되는 공용 이미지, Amazon ECR의 기존 컨테이너 이미지 또는 Amazon에서 관리하는 컨테이너 이미지 중에서 선택할 수 있습니다. 기본 이미지 선택과 관련된 세부 정보를 보려면 선택 항목과 일치하는 탭을 선택하십시오.

  Managed images

  - 이미지 운영 체제(OS) - 편집할 수 없습니다.
  - 이미지 이름 - 기존 레시피에 대해 선택한 기본 이미지 조합을 기반으로 미리 선택됩니다. 하지만 이미지 선택 옵션을 변경하면 미리 선택한 이미지 이름을 잃게 됩니다.
  - 자동 버전 관리 옵션 - 기본 레시피와 일치하지 않습니다. 자동 버전 관리 옵션은 기본적으로 선택한 OS 버전 사용 옵션으로 설정됩니다.

  > **Important**
  >
  > 시맨틱 버전 관리를 사용하여 파이프라인 빌드를 시작하는 경우 이 값을 사용 가능한 최신 OS 버전 사용으로 변경해야 합니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.

  ECR image

  - 이미지 운영 체제(OS) - 미리 선택되었지만 편집 가능합니다.
  - OS 버전 - 미리 선택되었지만 편집할 수 있습니다.
  - ECR 이미지 ID - 미리 입력되었지만 편집할 수 있습니다.

  Docker Hub image

  - 이미지 운영 체제(OS) - 편집할 수 없습니다.
  - OS 버전 - 미리 선택되었지만 편집할 수 있습니다.
  - 도커 이미지 ID - 미리 입력되지만 편집할 수 있습니다.

인스턴스 구성

- AMI 소스(필수) - 컨테이너 빌드 및 테스트 인스턴스의 기본 이미지로 사용할 사용자 지정 AMI를 식별합니다. AMI ID 또는 AMI ID가 포함된 AWS Systems Manager (SSM) 파라미터 스토어 파라미터일 수 있습니다.

- AMI ID -이 설정은 원래 항목으로 미리 채워지지 않습니다. 기본 이미지의 AMI ID를 입력합니다. 예시: *ami-1234567890abcdef1*.
- SSM 파라미터 - 기본 이미지의 AMI ID가 포함된 SSM 파라미터 스토어 파라미터의 이름 또는 ARN을 입력합니다. 예: */ib/test/param* 또는 `arn:aws:ssm:`*us-east-1*`:`*111122223333*`:parameter`*/ib/test/param*.

- 스토리지(볼륨)

  EBS 볼륨 1(AMI 루트) - 미리 입력되어 있습니다. 루트 볼륨 디바이스 이름, 스냅샷 또는 IOPS 선택은 편집할 수 없습니다. 하지만 크기와 같은 나머지 설정은 모두 변경할 수 있습니다. 새 볼륨을 추가할 수도 있습니다.

  > **Note**
  >
  > 다른 계정에서 공유된 기본 AMI를 지정한 경우, 지정된 보조 볼륨의 스냅샷도 계정과 공유해야 합니다.

작업 디렉터리

- 작업 디렉터리 경로 - 미리 입력되지만 편집할 수 있습니다.

Components

- 구성 요소 - 레시피에 이미 포함된 구성 요소는 각 구성 요소 목록(빌드 및 테스트) 끝에 있는 선택된 구성 요소 섹션에 표시됩니다. 필요에 맞게 선택한 구성 요소를 제거하거나 재정렬할 수 있습니다.

  CIS 강화 구성 요소는 Image Builder 레시피의 표준 구성 요소 순서 지정 규칙을 따르지 않습니다. 출력 이미지에 대해 벤치마크 테스트가 실행되도록 하기 위해 CIS 강화 구성 요소는 항상 마지막에 실행됩니다.

  > **Note**
  >
  > 빌드 및 테스트 구성 요소 목록에는 구성 요소 소유자 유형에 따라 사용 가능한 구성 요소가 표시됩니다. 구성 요소를 추가하려면 빌드 구성 요소 추가를 선택하고 적용되는 소유권 필터를 선택합니다. 예를 들어 AWS Marketplace 제품과 연결된 빌드 구성 요소를 추가하려면를 선택합니다`AWS Marketplace`. 그러면 구성 요소를 나열 AWS Marketplace 하는 콘솔 인터페이스 오른쪽에 선택 패널이 열립니다.

CIS 구성 요소에서를 선택합니다`Third party managed`.

선택한 구성 요소에 대해 다음 설정을 구성할 수 있습니다.

- 버전 관리 옵션 - 미리 선택되어 있지만 변경할 수 있습니다. 이미지 빌드가 항상 최신 버전의 구성 요소를 사용하도록 하려면 사용 가능한 최신 구성 요소 버전 사용 옵션을 선택하는 것이 좋습니다. 레시피에서 특정 구성 요소 버전을 사용해야 하는 경우 구성 요소 버전 지정을 선택하고 나타나는 구성 요소 버전 상자에 버전을 입력할 수 있습니다.
- 입력 파라미터 - 구성 요소가 받아들이는 입력 파라미터를 표시합니다. 값은 이전 버전의 레시피 값으로 미리 채워집니다. 이 레시피에서 이 구성 요소를 처음 사용하고 입력 파라미터에 대해 기본값이 정의된 경우 기본값이 값 상자에 회색으로 표시된 텍스트와 함께 나타납니다. 값을 입력하지 않으면 Image Builder에서 기본값을 사용합니다.

  입력 파라미터가 필요하지만 구성 요소에 기본값이 정의되어 있지 않은 경우 값을 제공해야 합니다. 누락된 필수 파라미터가 있고 기본값이 정의되지 않은 경우 Image Builder는 레시피 버전을 생성하지 않습니다.

  > **Important**
  >
  > 구성 요소 파라미터는 일반 텍스트 값이며 AWS CloudTrail에 기록됩니다. AWS Secrets Manager 또는 AWS Systems Manager 파라미터 스토어를 사용하여 보안 암호를 저장하는 것이 좋습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager란 무엇입니까?를 참조하십시오. AWS Systems Manager Parameter Store에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Parameter Store 섹션을 참조하십시오.

버전 관리 옵션 또는 입력 파라미터 설정을 확장하려면 설정 이름 옆에 있는 화살표를 선택하면 됩니다. 선택한 모든 구성 요소의 모든 설정을 확장하려면 모두 확장 스위치를 끄고 켤 수 있습니다.

Dockerfile 템플릿

- Dockerfile 템플릿 - 미리 채워져 있지만 편집할 수 있습니다. Image Builder가 런타임 시 빌드 정보로 대체하는 다음과 같은 컨텍스트 변수를 지정할 수 있습니다.

parentImage(필수)

빌드 시 이 변수는 레시피의 기본 이미지로 확인됩니다.

예시

```
FROM
{{{ imagebuilder:parentImage }}}
```

환경(구성 요소가 지정된 경우 필수)

이 변수는 구성 요소를 실행하는 스크립트로 확인됩니다.

예시

```
{{{ imagebuilder:environments }}}
```

구성 요소(선택 사항)

Image Builder는 컨테이너 레시피에 포함된 구성 요소에 대한 빌드 및 테스트 구성 요소 스크립트를 확인합니다. 이 변수는 환경 변수 뒤에 Dockerfile의 아무 곳에나 배치할 수 있습니다.

예시

```
{{{ imagebuilder:components }}}
```

대상 리포지토리

- 대상 리포지토리 이름 - 파이프라인이 실행되는 리전(리전 1) 에 대한 파이프라인 배포 구성에 지정된 다른 리포지토리가 없는 경우 출력 이미지가 저장되는 Amazon ECR 리포지토리입니다.

새 컨테이너 레시피 버전을 생성하려면:

1. 컨테이너 레시피 세부 정보 페이지 상단에서 새 버전 생성을 선택합니다. 컨테이너 레시피의 레시피 생성 페이지로 이동합니다.
2. 새 버전을 만들려면 변경한 다음 레시피 생성을 선택합니다.

이미지 파이프라인을 생성할 때 컨테이너 레시피를 생성하는 방법에 대한 자세한 내용은 이 가이드의 시작하기 섹션의 2단계: 레시피 선택 섹션을 참조하십시오.

AWS CLI(을)를 사용하여 컨테이너 레시피 생성

의 `imagebuilder create-container-recipe` 명령을 사용하여 Image Builder 컨테이너 레시피를 생성하려면 다음 단계를 AWS CLI따릅니다.

사전 조건

이 섹션의 Image Builder 명령을 실행하여를 사용하여 컨테이너 레시피를 생성하기 전에 레시피가 사용할 구성 요소를 생성 AWS CLI해야 합니다. 다음 단계의 컨테이너 레시피 예제는 이 가이드의 에서 사용자 지정 구성 요소 생성 AWS CLI 섹션에서 만든 예제 구성 요소를 참조합니다.

구성 요소를 만든 후 또는 기존 구성 요소를 사용하는 경우, 레시피에 포함하려는 ARN을 기록해 두십시오.

1. CLI 입력 JSON 파일 생성

   create-container-recipe 명령의 모든 입력을 인라인 명령 파라미터로 제공할 수 있습니다. 하지만 결과로 생성되는 명령은 상당히 길 수 있습니다. 명령을 간소화하기 위해 모든 컨테이너 레시피 설정이 포함된 JSON 파일을 대신 제공할 수 있습니다.

   > **Note**
   >
   > JSON 파일의 데이터 값에 대한 이름 지정 규칙은 Image Builder API 작업 요청 파라미터에 지정된 패턴을 따릅니다. API 작업 요청 파라미터를 검토하려면 EC2 Image Builder API 참조의 CreateContainerRecipe 명령을 참조하세요.
   > 데이터 값을 명령줄 파라미터로 제공하려면 AWS CLI 명령 참조에 지정된 파라미터 이름을 참조하십시오.

   다음은 이 예제의 파라미터를 요약한 것입니다.

   - 구성 요소(객체 배열, 필수) - `ComponentConfiguration` 객체 배열을 포함합니다. 빌드 구성 요소를 하나 이상 지정해야 합니다.

> **Note**
>
> Image Builder는 레시피에 지정한 순서대로 구성 요소를 설치합니다. 하지만 CIS 하드닝 구성 요소는 출력 이미지에 대해 벤치마크 테스트가 실행되도록 하기 위해 항상 마지막에 실행됩니다.

- componentARN(문자열, 필수) – 구성 요소 ARN입니다.

  > **Tip**
  >
  > 예제를 사용하여 자체 컨테이너 레시피를 생성하려면 예제 ARN을 레시피에 사용 중인 구성 요소의 ARN으로 바꾸십시오. 여기에는 각각의 AWS 리전, 이름, 버전 번호가 포함됩니다.

- 파라미터(객체 배열) - `ComponentParameter` 객체 배열을 포함합니다. 입력 파라미터가 필요하지만 구성 요소에 기본값이 정의되어 있지 않은 경우 값을 제공해야 합니다. 누락된 필수 파라미터가 있고 기본값이 정의되지 않은 경우 Image Builder는 레시피 버전을 생성하지 않습니다.

  > **Important**
  >
  > 구성 요소 파라미터는 일반 텍스트 값이며 AWS CloudTrail에 기록됩니다. AWS Secrets Manager 또는 AWS Systems Manager 파라미터 스토어를 사용하여 보안 암호를 저장하는 것이 좋습니다. Secrets Manager에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 Secrets Manager란 무엇입니까?를 참조하십시오. AWS Systems Manager Parameter Store에 대한 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager Parameter Store 섹션을 참조하십시오.

  - 이름(문자열, 필수) — 설정할 구성 요소 파라미터의 이름입니다.
  - 값(문자열 배열, 필수) - 명명된 구성 요소 파라미터의 값을 설정하는 문자열 배열을 포함합니다. 구성 요소에 대해 정의된 기본값이 있고 다른 값이 제공되지 않는 경우는 기본값을 AWSTOE 사용합니다.

- containerType(문자열, 필수) - 생성할 컨테이너의 유형입니다. 유효한 값으로는 `DOCKER`가 포함됩니다.

- dockerfileTemplateData(문자열) - 이미지를 빌드하는 데 사용되는 Dockerfile 템플릿으로, 인라인 데이터 블롭으로 표현됩니다.
- name(문자열, 필수) - 컨테이너 레시피의 이름입니다.
- description(문자열) - 컨테이너 레시피에 대한 설명입니다.
- parentImage(문자열, 필수) - 컨테이너 레시피에서 사용자 지정 이미지의 기준으로 사용할 Docker 컨테이너 이미지입니다.
  - DockerHub에서 호스팅되는 퍼블릭 이미지
  - Amazon ECR의 기존 컨테이너 이미지
  - Amazon 관리형 컨테이너 이미지
- platformOverride(문자열) – 사용자 지정 기본 이미지를 사용할 때 운영 체제 플랫폼을 지정합니다.
- semanticVersion(문자열, 필수) - 다음 형식으로 지정된 컨테이너 레시피의 의미 체계 버전으로, 각 위치에는 특정 버전 <major>.<minor>.<patch>를 나타내는 숫자 값이 있습니다. 예를 들어 `1.0.0`(이)가 됩니다. Image Builder 리소스의 시맨틱 버전 관리에 대한 자세한 내용은 Image Builder의 의미 체계 버전 관리(을)를 참조하십시오.
- tags(문자열 맵) – 컨테이너 레시피에 연결되는 태그입니다.
- instanceConfiguration(객체) – 컨테이너 이미지를 빌드하고 테스트하기 위해 인스턴스를 구성하는 데 사용할 수 있는 옵션 그룹입니다.
  - 이미지(문자열) - 컨테이너 빌드 및 테스트 인스턴스의 기본 이미지입니다. 여기에는 AMI ID가 포함되거나 접두사가 인 AWS Systems Manager (SSM) 파라미터 스토어 파라미터를 지정한 `ssm:`다음 파라미터 이름 또는 ARN을 지정할 수 있습니다. SSM 파라미터를 사용하는 경우 파라미터 값에 AMI ID가 포함되어야 합니다. 기본 이미지를 지정하지 않으면 Image Builder는 적절한 Amazon ECS 최적화 AMI를 기본 이미지로 사용합니다.
  - blockDeviceMappings(객체 배열) - image 파라미터에 지정된 이미지 빌더 AMI에서 인스턴스를 빌드하기 위해 연결할 블록 디바이스를 정의합니다.
    - deviceName(문자열) - 이러한 매핑이 적용되는 디바이스입니다.
    - ebs(객체) – 이 매핑에 대한 Amazon EBS 특정 구성을 관리하는 데 사용합니다.
      - deleteOnTermination(부울) – 연결된 디바이스의 종료 시 삭제를 구성하는 데 사용합니다.
      - encrypted(부울) – 장치 암호화를 구성하는 데 사용됩니다.
      - volumeSize(정수) - 디바이스의 볼륨 크기를 재정의하는 데 사용됩니다.
      - volumeType(문자열) - 디바이스의 볼륨 유형을 재정의하는 데 사용됩니다.

- targetRepository(객체, 필수) - 파이프라인이 실행되는 리전(리전 1) 에 대한 파이프라인 배포 구성에 지정된 다른 리포지토리가 없는 경우 컨테이너 이미지의 대상 리포지토리입니다.
  - repositoryName(문자열, 필수) – 출력 컨테이너 이미지가 저장되는 컨테이너 리포지토리의 이름입니다. 이 이름에 리포지토리 위치가 접두사로 추가됩니다.
  - service(문자열, 필수) – 이 이미지가 등록된 서비스를 지정합니다.
- workingDirectory(문자열) – 구축 및 테스트 워크플로 중에 사용할 작업 디렉터리입니다.

```
{
 "components": [
   {
   "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333:component/
helloworldal2/x.x.x"
   }
 ],
 "containerType": "DOCKER",
 "description": "My Linux Docker container image",
 "dockerfileTemplateData": "FROM
 {{{ imagebuilder:parentImage }}}\n{{{ imagebuilder:environments }}}\n{{{ imagebuilder:comp
 "name": "amazonlinux-container-recipe",
 "parentImage": "amazonlinux:latest",
 "platformOverride": "Linux",
 "semanticVersion": "1.0.2",
 "tags": {
   "sometag" : "Tag detail"
 },
 "instanceConfiguration": {
   "image": "ami-1234567890abcdef1",
   "blockDeviceMappings": [
   {
   "deviceName": "/dev/xvda",
   "ebs": {
    "deleteOnTermination": true,
    "encrypted": false,
    "volumeSize": 8,
    "volumeType": "gp2"
    }
    }
   ]
 },
 "targetRepository": {
   "repositoryName": "myrepo",
```

```
    "service": "ECR"
  },
  "workingDirectory": "/tmp"
}
```

2. 레시피 생성

   다음 명령을 사용하여 레시피를 생성합니다. 이전 단계에서 생성한 JSON 파일의 이름을 `--cli-input-json` 파라미터에 입력합니다.

```
aws imagebuilder create-container-recipe --cli-input-json file://create-container-recipe.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

리소스 정리

예상치 못한 요금이 부과되지 않도록 하려면 이 가이드의 예제에서 만든 리소스와 파이프라인을 정리하세요. Image Builder에서 리소스 삭제에 대한 자세한 내용은 오래되거나 사용되지 않는 Image Builder 리소스 삭제(을)를 참조하세요.

Image Builder 인프라 구성 관리

인프라 구성을 사용하여 Image Builder가 EC2 Image Builder 이미지를 구축하고 테스트하는 데 사용하는 Amazon EC2 인프라를 지정할 수 있습니다. 인프라 설정은 다음과 같습니다.

- 내 빌드 및 테스트 인프라의 인스턴스 유형. 인스턴스 유형을 두 개 이상 지정하는 것이 좋습니다. 이렇게 하면 Image Builder가 용량이 충분한 풀에서 인스턴스를 시작할 수 있기 때문입니다. 이렇게 하면 일시적인 빌드 실패를 줄일 수 있습니다.

  Mac 이미지의 경우 기본적으로 macOS 운영 체제를 지원하는 인스턴스 유형을 선택할 수 있습니다. 자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 Amazon EC2 Mac 인스턴스를 참조하세요.

- 이미지에서 시작하는 인스턴스가 이동해야 하는 호스트, 호스트 배치 그룹 또는 가용 영역을 지정할 수 있는 인스턴스 배치 설정입니다.
- 빌드 및 테스트 인스턴스에 사용자 지정 활동을 수행하는 데 필요한 권한을 제공하는 인스턴스 프로파일입니다. 예를 들어 Amazon S3에서 리소스를 불러오는 구성 요소가 있는 경우 그 인스턴스 프로파일에는 해당 파일에 액세스할 수 있는 권한이 필요합니다. 또한 인스턴스 프로파일에는 EC2 Image Builder가 인스턴스와 성공적으로 통신하기 위한 최소한의 권한 세트가 필요합니다. 자세한 내용은 Image Builder를 사용하여 사용자 지정 이미지를 빌드하도록 설정 단원을 참조하십시오.
- 파이프라인 빌드 및 테스트 인스턴스의 VPC, 서브넷 및 보안 그룹.
- Image Builder가 내 빌드 및 테스트의 애플리케이션 로그를 저장하는 Amazon S3 위치입니다. 로깅을 구성하는 경우 인프라 구성에 지정된 인스턴스 프로파일에 대상 버킷(`arn:aws:s3:::BucketName/*`)에 대한 `s3:PutObject` 권한이 있어야 합니다.
- 빌드가 실패하고 `terminateInstanceOnFailure`(을)를 `false`(으)로 설정한 경우 인스턴스에 로그온하여 문제를 해결할 수 있게 해주는 Amazon EC2 키 페어입니다.
- Image Builder에서 이벤트 알림을 보내는 SNS 주제입니다. Image Builder가 Amazon SNS와 어떻게 연동되는지에 관한 자세한 내용은 Image Builder에서의 Amazon SNS 통합 섹션을 참조하세요.

> **Note**
>
> SNS 주제가 암호화된 경우 이 주제를 암호화하는 키는 Image Builder 서비스가 실행되는 계정에 있어야 합니다. Image Builder는 다른 계정의 키로 암호화된 SNS 주제에 알림을 보낼 수 없습니다.

Image Builder 콘솔, Image Builder API 또는 AWS CLI의 imagebuilder 명령을 사용하여 인프라 구성을 생성하고 관리할 수 있습니다.

내용

- 인프라 구성 세부 정보 나열 및 보기
- 인프라 구성 생성
- 인프라 구성 업데이트
- Image Builder 및 AWS PrivateLink 인터페이스 VPC 엔드포인트

> **Tip**
>
> 동일한 유형의 리소스가 여러 개 있는 경우 태그를 지정하면 할당한 태그에 따라 특정 리소스를 식별하는 데 도움이 됩니다. 에서 Image Builder 명령을 사용하여 리소스에 태그를 지정하는 방법에 대한 자세한 내용은이 가이드의 리소스 태깅 섹션을 AWS CLI참조하세요.

인프라 구성 세부 정보 나열 및 보기

이 섹션에서는 EC2 Image Builder 인프라 구성에 대한 정보를 찾고 세부 정보를 볼 수 있는 다양한 방법을 설명합니다.

인프라 구성 세부 정보

- 에서 인프라 구성 나열 AWS CLI
- 에서 인프라 구성 세부 정보 가져오기 AWS CLI

에서 인프라 구성 나열 AWS CLI

다음 예제에서는 AWS CLI에서 list-infrastructure-configurations 명령을 사용하여 모든 내 인프라 구성을 모두 나열하는 방법을 보여 줍니다.

```
aws imagebuilder list-infrastructure-configurations
```

에서 인프라 구성 세부 정보 가져오기 AWS CLI

다음 예제에서는에서 get-infrastructure-configuration 명령을 사용하여 Amazon 리소스 이름(ARN)을 지정하여 인프라 구성의 세부 정보를 AWS CLI 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-infrastructure-configuration --infrastructure-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-
infrastructure-configuration
```

인프라 구성 생성

이 섹션에서는 Image Builder 콘솔 또는의 imagebuilder 명령을 사용하여 인프라 구성을 AWS CLI 생성하는 방법을 설명합니다.

Console

Image Builder 콘솔에서 인프라 구성 리소스를 생성하려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 인프라 구성을 선택합니다.
3. 인프라 구성 생성을 선택합니다.
4. 일반 섹션에서 다음 정보를 입력합니다.
   - 인프라 구성 리소스의 이름을 입력합니다.
   - 빌드 및 테스트 인스턴스에서 구성 요소 권한을 위해 인스턴스 프로파일과 연결할 IAM 역할을 선택합니다. Image Builder는 이러한 권한을 사용하여 구성 요소를 다운로드 및 실행하고, CloudWatch에 로그를 업로드하고, 내 레시피의 구성 요소가 지정하는 추가 작업을 수행합니다.
5. AWS 인프라 패널에서 사용 가능한 나머지 인프라 설정을 구성할 수 있습니다. 다음 필수 정보를 입력합니다.
   - 인스턴스 유형 - 이 빌드에 사용할 하나 이상의 인스턴스 유형을 지정할 수 있습니다. 서비스가 가용 여부에 따라 이러한 인스턴스 유형 중 하나를 선택할 것입니다.

     > **Note**
     >
     > Mac 인스턴스는 전용 호스트의 `.metal` 인스턴스 유형에서 실행됩니다. 인스턴스 유형은 인스턴스가 실행되는 호스트에 대해 정의된 유형 중 하나와 일치해야 합니다. Mac 인스턴스 및 macOS 운영 체제를 기본적으로 지원하는 인스턴스 유형 목록에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Amazon EC2 Mac 인스턴스를 참조하세요.

   - SNS 주제(선택 사항) - EC2 Image Builder에서 알림 및 경보를 받으려면 SNS 주제를 선택합니다.

   다음 설정에 값을 제공하지 않으면 해당하는 경우 서비스별 기본값이 사용됩니다.

   - VPC, 서브넷 및 보안 그룹 — Image Builder는 기본 VPC와 서브넷을 사용합니다. VPC 인터페이스 엔드포인트를 구성하는 방법에 대한 자세한 내용은 Image Builder 및 AWS PrivateLink 인터페이스 VPC 엔드포인트 섹션을 참조하세요.
   - 문제 해결 설정 섹션에서 다음 값을 구성할 수 있습니다.

- 기본적으로 실패 시 인스턴스 종료 확인란이 선택되어 있습니다. 하지만 빌드가 실패하는 경우 EC2 인스턴스에 로그온하여 문제를 해결할 수 있습니다. 빌드 실패 후에도 인스턴스를 계속 실행하려면 확인란의 선택을 취소하세요.
- 키 페어 - 빌드 실패 후에도 EC2 인스턴스가 계속 실행되는 경우, 키 페어를 생성하거나 기존 키 페어를 사용하여 인스턴스에 로그온하고 문제를 해결할 수 있습니다.
- 로그 - Image Builder가 빌드 및 테스트 문제를 해결하는 데 도움이 되는 애플리케이션 로그를 작성할 수 있는 S3 버킷을 지정할 수 있습니다. S3 버킷을 지정하지 않는 경우 Image Builder는 애플리케이션 로그를 인스턴스에 기록합니다.

- 인스턴스 메타데이터 설정 섹션에서 Image Builder가 이미지를 빌드하고 테스트하는 데 사용하는 EC2 인스턴스에 적용할 다음 값을 구성할 수 있습니다.
  - 메타데이터 버전을 선택하여 EC2에서 인스턴스 메타데이터 검색 요청에 서명된 토큰 헤더가 필요한지 여부를 결정합니다.
    - V1 및 V2(토큰 선택 사항) - 아무것도 선택하지 않은 경우의 기본값입니다.
    - V2(토큰 필요)

    > **Note**
    >
    > Image Builder가 파이프라인 빌드에서 시작하는 모든 EC2 인스턴스가 IMDSv2를 사용하도록 구성하여 인스턴스 메타데이터 검색 요청에 서명된 토큰 헤더가 필요하도록 하는 것이 좋습니다.

  - 메타데이터 토큰 응답 홉 제한 – 메타데이터 토큰이 이동할 수 있는 네트워크 홉 수입니다. 최소 홉: 1, 최대 홉: 64개, 기본값은 홉 1개입니다.
- 인스턴스 배치 설정 섹션에서 Image Builder가 이미지를 빌드하고 테스트하는 데 사용하는 EC2 인스턴스에 적용할 다음 값을 구성할 수 있습니다.
  - 이미지 생성 중에 Image Builder가 인스턴스를 시작하는 가용 영역을 선택할 수 있습니다.
  - 선택적으로 시작하는 인스턴스를 실행하는 서버에 대해 테넌시를 선택합니다. 기본적으로 EC2 인스턴스는 공유 테넌시 하드웨어에서 실행됩니다. 즉, 여러 AWS 계정 이 동일한 물리적 하드웨어를 공유할 수 있습니다. `dedicated` 테넌시가 있는 인스턴스는 단일 테넌트 하드웨어에서 실행됩니다. `host` 테넌시가 있는 인스턴스는 전용 호스트에서 실행됩니다.

    Mac 인스턴스에는 사용자 지정 이미지를 빌드하기 전에 사전 조건으로 생성된 전용 호스트가 필요합니다. macOS 이미지에 대해 `host`를 선택합니다. 그런 다음 대상 호스트 또는 호스트 리소스 그룹을 선택하여 인스턴스를 시작할 수 있지만 전용 호스트에 자동 배치가

활성화된 경우에는 필요하지 않습니다. 자세한 내용을 알아보려면 Amazon EC2 사용 설명서의 자동 배치를 참조하세요.

- 테넌시 호스트 ID - 인스턴스가 실행되는 전용 호스트의 ID입니다.
- 테넌시 호스트 리소스 그룹 - 인스턴스를 시작할 호스트 리소스 그룹의 Amazon 리소스 이름(ARN)입니다.

6. 인프라 태그 섹션(선택 사항)에서 빌드 프로세스 중에 Image Builder가 시작하는 Amazon EC2 인스턴스에 메타데이터 태그를 할당할 수 있습니다. 태그를 키 값 페어로 입력합니다.
7. 태그 섹션(선택 사항)에서 Image Builder가 출력으로 생성하는 인프라 구성 리소스에 메타데이터 태그를 할당할 수 있습니다. 태그를 키 값 페어로 입력합니다.

AWS CLI

다음 절차는 AWS CLI에서 Image Builder create-infrastructure-configuration 명령을 사용하여 이미지의 인프라를 구성하는 방법을 보여줍니다. 2단계의 명령은 1단계에서 생성한 파일을 가져옵니다. 이러한 예제에서는 1단계의 파일을 `create-infrastructure-configuration.json`이라고 합니다.

1. CLI 입력 JSON 파일 생성

   다음 예제에서는 인프라 구성을 위해 생성할 수 있는 JSON 파일의 변형을 보여줍니다. 파일 편집 도구를 사용하여 자신만의 JSON 파일을 생성합니다.

   예제 1: 실패한 빌드에서 인스턴스를 유지하기 위한 구성

   이 예제에서는 두 개의 인스턴스 유형 `m5.large` 및 `m5.xlarge`를 지정합니다. 인스턴스 유형을 두 개 이상 지정하는 것이 좋습니다. 이렇게 하면 Image Builder가 용량이 충분한 풀에서 인스턴스를 시작할 수 있기 때문입니다. 이렇게 하면 일시적인 빌드 실패를 줄일 수 있습니다.

   `instanceProfileName`(은)는 프로파일이 사용자 지정 활동을 수행하는 데 필요한 권한을 인스턴스에 제공하는 인스턴스 프로파일을 지정합니다. 예를 들어 Amazon S3에서 리소스를 불러오는 구성 요소가 있는 경우 그 인스턴스 프로파일에는 해당 파일에 액세스할 수 있는 권한이 필요합니다. 또한 인스턴스 프로파일에는 EC2 Image Builder가 인스턴스와 성공적으로 통신하기 위한 최소한의 권한 세트가 필요합니다. 자세한 내용은 Image Builder를 사용하여 사용자 지정 이미지를 빌드하도록 설정 단원을 참조하십시오.

```
{
    "name": "ExampleInfraConfigDontTerminate",
    "description": "An example that will retain instances of failed builds",
```

```
    "instanceTypes": [
        "m5.large", "m5.xlarge"
    ],
    "instanceProfileName": "myIAMInstanceProfileName",
    "securityGroupIds": [
        "sg-12345678"
    ],
    "subnetId": "sub-12345678",
    "logging": {
        "s3Logs": {
            "s3BucketName": "my-logging-bucket",
            "s3KeyPrefix": "my-path"
        }
    },
    "keyPair": "myKeyPairName",
    "terminateInstanceOnFailure": false,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:MyTopic"
}
```

예제 2: 자동 배치를 사용한 macOS 구성

이 예제에서는 전용 호스트에 자동 배치가 활성화된 Mac 인스턴스의 인스턴스 유형과 배치를 지정합니다.

```
{
   "name": "macOSInfraConfigAutoPlacement",
   "description": "An example infrastructure configuration for macOS.",
   "instanceProfileName": "EC2InstanceProfileForImageBuilder",
   "instanceTypes": ["mac1.metal, mac2.metal"],
   "terminateInstanceOnFailure": false,
   "placement": {
      "tenancy": "host"
   }
}
```

예제 3: 호스트 ID가 지정된 macOS 구성

이 예제에서는 특정 전용 호스트를 대상으로 하는 Mac 인스턴스의 인스턴스 유형과 배치를 지정합니다.

```
{
   "name": "macOSInfraConfigHostPlacement",
```

```
   "description": "An example infrastructure configuration for macOS.",
   "instanceProfileName": "EC2InstanceProfileForImageBuilder",
   "instanceTypes": ["mac2-m1ultra.metal"],
   "terminateInstanceOnFailure": false,
   "placement": {
      "tenancy": "host",
      "hostId" : "h-1234567890abcdef0"
   }
}
```

2. 다음 명령을 실행할 때 생성한 파일을 입력으로 사용합니다.

```
aws imagebuilder create-infrastructure-configuration --cli-input-json
 file://create-infrastructure-configuration.json
```

인프라 구성 업데이트

이 섹션에서는 Image Builder 콘솔 또는의 imagebuilder 명령을 사용하여 인프라 구성 리소스를 AWS CLI 업데이트하는 방법을 다룹니다. 리소스를 추적하려면 다음과 같이 태그를 적용할 수 있습니다. 태그를 키 값 페어로 입력합니다.

- 리소스 태그는 빌드 프로세스 중에 Image Builder가 시작하는 Amazon EC2 인스턴스에 메타데이터 태그를 할당합니다.
- 태그는 Image Builder가 출력으로 생성하는 인프라 구성 리소스에 메타데이터 태그를 할당합니다.

Console

Image Builder 콘솔에서 다음과 같은 인프라 구성 세부 정보를 편집할 수 있습니다.

- 인프라 구성에 관한 설명.
- 인스턴스 프로파일과 연결할 IAM 역할.
- AWS 인프라 - 인스턴스 유형 및 알림을 위한 SNS 주제 포함.
- VPC, 서브넷 및 보안 그룹.
- 장애 발생 시 인스턴스 종료, 연결을 위한 키 페어, 인스턴스 로그를 위한 선택적 S3 버킷 위치를 포함한 문제 해결 설정.

Image Builder 콘솔에서 인프라 구성 리소스를 업데이트하려면 다음 단계를 따르세요.

기존 Image Builder 인프라 구성 선택

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 계정의 인프라 구성 리소스 목록을 보려면 탐색 창에서 인프라 구성을 선택합니다.
3. 세부 정보를 보거나 인프라 구성을 편집하려면 구성 이름 링크를 선택합니다. 이렇게 하면 인프라 구성에 대한 세부 정보 보기가 열립니다.

> **Note**
>
> 또한 구성 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

4. 인프라 세부 정보 패널의 오른쪽 상단에서 편집을 선택합니다.
5. 인프라 구성에 적용한 업데이트를 저장할 준비가 되면 변경 내용 저장을 선택합니다.

AWS CLI

다음 예제는 AWS CLI에서 Image Builder update-infrastructure-configuration 명령을 사용하여 이미지의 인프라 구성을 업데이트하는 방법을 보여줍니다.

1. CLI 입력 JSON 파일 생성

   이 인프라 구성 예제는 설정을 `false`(으)로 업데이트했다는 점을 제외하면 생성 예제와 동일한 `terminateInstanceOnFailure` 설정을 사용합니다. update-infrastructure-configuration 명령을 실행하면 이 인프라 구성을 사용하는 파이프라인이 빌드를 종료하고 빌드가 실패하면 인스턴스를 테스트합니다.

   파일 편집 도구를 사용하여 다음 예제에 표시된 키와 환경에 유효한 값을 포함하는 JSON 파일을 생성합니다. 이 예제에서는 `update-infrastructure-configuration.json`(이)라는 이름의 파일이 사용됩니다.

```
{
"infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
"description": "An example that will terminate instances of failed builds",
"instanceTypes": [
    "m5.large", "m5.2xlarge"
],
"instanceProfileName": "myIAMInstanceProfileName",
```

```
"securityGroupIds": [
    "sg-12345678"
],
"subnetId": "sub-12345678",
"logging": {
    "s3Logs": {
        "s3BucketName": "my-logging-bucket",
        "s3KeyPrefix": "my-path"
    }
},
"terminateInstanceOnFailure": true,
"snsTopicArn": "arn:aws:sns:us-west-2:123456789012:MyTopic"
}
```

2. 다음 명령을 실행할 때 생성한 파일을 입력으로 사용합니다.

```
aws imagebuilder update-infrastructure-configuration --cli-input-json
 file://update-infrastructure-configuration.json
```

Image Builder 및 AWS PrivateLink 인터페이스 VPC 엔드포인트

인터페이스 VPC 엔드포인트를 생성하여 VPC와 EC2 Image Builder 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 인터넷 게이트웨이AWS PrivateLink, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 Image Builder APIs에 비공개로 액세스할 수 있는 기술로 구동됩니다. VPC의 인스턴스는 Image Builder API와 통신하는 데 퍼블릭 IP 주소를 필요로 하지 않습니다. VPC와 Image Builder 간 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다. 새 이미지를 생성할 때 인프라 구성에서 VPC 서브넷 ID를 지정할 수 있습니다.

> **Note**
>
> VPC 내에서 액세스하는 각 서비스에는 자체 엔드포인트 정책이 있는 자체 인터페이스 엔드포인트가 있습니다. Image Builder는 AWSTOE 구성 요소 관리자 애플리케이션을 다운로드하고 S3 버킷에서 관리형 리소스에 액세스하여 사용자 지정 이미지를 생성합니다. 이러한 버킷에 대한 액세스 권한을 부여하려면 이를 허용하도록 S3 엔드포인트 정책을 업데이트해야 합니다. 자세한 내용은 S3 버킷 액세스에 대한 사용자 지정 정책 섹션을 참조하세요.

VPC 엔드포인트에 대한 자세한 정보는 Amazon VPC 사용 설명서사용 설명서의 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 참조합니다.

Image Builder VPC 엔드포인트에 대한 고려 사항

Image Builder에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서에서 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다.

Image Builder는 VPC에서 모든 API 작업에 대한 호출 수행을 지원합니다.

Image Builder용 인터페이스 VPC 엔드포인트 생성하기

Image Builder 서비스에 대한 VPC 엔드포인트를 생성하려면 Amazon VPC 콘솔 또는 AWS Command Line Interface ()를 사용할 수 있습니다AWS CLI. 자세한 내용은 Amazon VPC 사용 설명서(Amazon VPC User Guide)의 인터페이스 엔드포인트 생성(Creating an interface endpoint)을 참조합니다.

다음 서비스 이름을 사용하여 Image Builder용 VPC 엔드포인트를 생성합니다.

- com.amazonaws.*region*.imagebuilder

엔드포인트에 프라이빗 DNS를 사용하도록 설정하는 경우, 리전에 대한 기본 DNS 이름(예: `imagebuilder.us-east-1.amazonaws.com`)을 사용하여 Image Builder에 API 요청을 할 수 있습니다. 대상 리전에 적용되는 엔드포인트를 조회하려면 Amazon Web Services 일반 참조의 EC2 Image Builder 엔드포인트 및 할당량을 참조합니다.

자세한 내용은 Amazon VPC 사용 설명서(Amazon VPC User Guide)의 인터페이스 엔드포인트를 통해 서비스 액세스(Accessing a service through an interface endpoint)를 참조합니다.

Image Builder에 대한 VPC 엔드포인트 정책 생성하기

Image Builder에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 위탁자.
- 수행할 수 있는 작업.
- 작업을 수행할 있는 리소스.

레시피에서 Amazon 관리 구성 요소를 사용하는 경우 Image Builder의 VPC 엔드포인트는 다음과 같은 서비스 소유 구성 요소 라이브러리에 대한 액세스를 허용해야 합니다.

```
arn:aws:imagebuilder:region:aws:component/*
```

> **Important**
>
> 기본값이 아닌 정책이 EC2 Image Builder의 인터페이스 VPC 엔드포인트에 적용되는 경우에서 실패한 요청과 같은 특정 실패한 API 요청은 AWS CloudTrail 또는 Amazon CloudWatch에 로깅되지 않을 `RequestLimitExceeded`수 있습니다.

자세한 내용은 Amazon VPC 사용 설명서(Amazon VPC User Guide)의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어(Controlling access to services with VPC endpoints)를 참조합니다.

S3 버킷 액세스에 대한 사용자 지정 정책

Image Builder는 공개적으로 사용 가능한 S3 버킷을 사용하여 구성 요소와 같은 관리형 리소스를 저장하고 액세스합니다. 또한 별도의 S3 버킷에서 AWSTOE 구성 요소 관리 애플리케이션을 다운로드합니다. 환경에서 Amazon S3용 VPC 엔드포인트를 사용하는 경우, Image Builder가 다음 S3 버킷에 액세스할 수 있도록 S3 VPC 엔드포인트 정책이 허용되는지 확인해야 합니다. 버킷 이름은 AWS 리전(*##*) 및 애플리케이션 환경(*##*)별로 고유합니다. Image Builder 및는 `prod`, `preprod`및 애플리케이션 환경을 AWSTOE 지원합니다`beta`.

- AWSTOE 구성 요소 관리자 버킷:

  ```
  s3://ec2imagebuilder-toe-region-environment
  ```

  예: s3://ec2imagebuilder-toe-us-west-2-prod/*

- Image Builder는 리소스 버킷을 관리합니다.

  ```
  s3://ec2imagebuilder-managed-resources-region-environment/components
  ```

  예: s3://ec2imagebuilder-managed-resources-us-west-2-prod/components/*

VPC 엔드포인트 정책 예제

이 섹션에는 사용자 지정 VPC 엔드포인트 정책의 예가 포함되어 있습니다.

Image Builder 작업에 대한 일반 VPC 엔드포인트 정책

Image Builder에 대한 다음 예제 엔드포인트 정책은 Image Builder 이미지 및 구성 요소 삭제 권한을 거부합니다. 또한 이 정책 예에서는 다른 모든 EC2 Image Builder 작업을 수행할 수 있는 권한을 부여합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Action": "imagebuilder:*",
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Action": [
            "imagebuilder: DeleteImage"
        ],
        "Effect": "Deny",
        "Resource": "*",
    },
    {
        "Action": [
            "imagebuilder: DeleteComponent"
        ],
        "Effect": "Deny",
        "Resource": "*",
    }]
}
```

조직별 액세스 제한, 관리되는 구성 요소 액세스 허용하기

다음 예제 엔드포인트 정책은 조직에 속한 아이덴티와 리소스에 대한 액세스를 제한하고 Amazon 관리형 Image Builder 구성 요소에 대한 액세스를 제공하는 방법을 보여줍니다. *region*, *principal-org-id* 및 *resource-org-id*를 해당 조직의 값으로 바꿉니다.

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

```
      },
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "principal-org-id",
          "aws:ResourceOrgID": "resource-org-id"
        }
      }
    },
    {
      "Sid": "AllowAccessToEC2ImageBuilderComponents",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "imagebuilder:GetComponent"
      ],
      "Resource": [
        "arn:aws:imagebuilder:region:aws:component/*"
      ]
    }
  ]
}
```

Amazon S3 버킷 액세스에 대한 VPC 엔드포인트 정책

다음 S3 엔드포인트 정책 예제는 Image Builder가 사용자 지정 이미지를 구축하는 데 사용하는 S3 버킷에 대한 액세스를 제공하는 방법을 보여줍니다. ## 및 ##을 해당 조직의 값으로 바꿉니다. 애플리케이션 요구 사항에 따라 기타 필수 권한을 정책에 추가합니다.

> **Note**
>
> Linux 이미지의 경우 이미지 레시피에 사용자 데이터를 지정하지 않으면 Image Builder는 스크립트를 추가하여 이미지의 빌드 및 테스트 인스턴스에 Systems Manager 에이전트를 다운로드하고 설치합니다. 에이전트를 다운로드하기 위해 Image Builder는 빌드 리전의 S3 버킷에 액세스합니다.
>
> Image Builder가 빌드 및 테스트 인스턴스를 부트스트랩할 수 있도록 하려면 S3 엔드포인트 정책에 다음과 같은 다른 리소스를 추가합니다.
>
> `"arn:aws:s3:::amazon-ssm-region/*"`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowImageBuilderAccessToAppAndComponentBuckets",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::ec2imagebuilder-toe-region-environment/*",
        "arn:aws:s3:::ec2imagebuilder-managed-resources-region-environment/components/
*"
      ]
    }
  ]
}
```

Image Builder 배포 설정 관리

출력 이미지에 대한 배포 설정을 구성하기 전에 배포 대상 리전의 출력 이미지에서 시작된 인스턴스에 대한 기본 인프라 또는 기타 요구 사항의 가용성을 확인하는 것이 좋습니다. 예를 들어 일부 리전은 macOS 이미지에서 인스턴스를 시작하는 데 필요한 EC2 Mac 전용 호스트를 지원하지 않습니다. 전용 호스트의 인스턴스 유형 및 요금에 대한 자세한 내용은 Amazon EC2 전용 호스트 요금 섹션을 참조하세요.

Image Builder로 배포 설정을 생성한 후 Image Builder 콘솔, Image Builder API 또는 AWS CLI의 imagebuilder 명령을 사용하여 배포 설정을 관리할 수 있습니다. 배포 설정을 사용하면 다음 작업을 수행할 수 있습니다.

AMI 배포

- 출력 AMI의 이름과 설명을 지정합니다.
- 소유자의 계정에서 AMI를 시작할 수 있도록 다른, AWS 계정조직 및 OUs에 권한을 부여합니다. AMI와 관련된 요금은 소유자 계정으로 청구됩니다.

> **Note**
>
> AMI를 공개하려면 시작 권한 승인 계정을 `all`(으)로 설정합니다. 자세한 내용 및 예제는 Amazon EC2 API 참조의 ModifyImageAttribute 섹션을 참조하세요.

- 대상 지역의 지정된 각 대상 계정, 조직 및 OU에 대해 출력 AMI의 사본을 생성합니다. 대상 계정, 조직 및 OU는 AMI 사본을 소유하며 모든 관련 요금이 청구됩니다. AMI를 AWS Organizations 및 OUs에 배포하는 방법에 대한 자세한 내용은 조직 또는 OUs.
- AMI를 다른의 소유자 계정에 복사합니다 AWS 리전.
- VM 이미지 디스크를 Amazon Simple Storage Service(S3)로 내보냅니다. 자세한 내용은 에서 출력 VM 디스크에 대한 배포 설정 생성 AWS CLI 단원을 참조하십시오.

컨테이너 이미지 배포

- Image Builder가 배포 지역에 출력 이미지를 저장하는 ECR 리포지토리를 지정합니다.

다음과 같은 방법으로 배포 설정을 사용하여 대상 리전, 계정 AWS Organizations 및 조직 단위(OUs)에 이미지를 한 번 또는 모든 파이프라인 빌드로 전송할 수 있습니다.

- 업데이트된 이미지를 지정된 리전, 계정, 조직 및 OU에 자동으로 전송하려면 일정에 따라 실행되는 Image Builder 파이프라인과 함께 배포 설정을 사용하세요.
- 새 이미지를 생성하여 지정된 리전, 계정, 조직 및 OU에 전달하려면, 작업 메뉴의 파이프라인 실행을 사용하여 Image Builder 콘솔에서 한 번 실행하는 Image Builder 파이프라인과 함께 배포 설정을 사용하세요.
- 새 이미지를 생성하여 지정된 리전, 계정, 조직 및 OU에 전달하려면 AWS CLI에서 다음 API 작업 또는 Image Builder 명령과 함께 배포 설정을 사용하세요.
  - Image Builder API에서 CreateImage 작업.
  - AWS CLI에서 create-image 명령.
- 일반 이미지 빌드 프로세스의 일환으로 가상 머신(VM) 이미지 디스크를 대상 지역의 S3 버킷으로 내보내는 것입니다.

> **Tip**
>
> 동일한 유형의 리소스가 여러 개 있는 경우, 태그를 지정하면 할당한 태그에 따라 특정 리소스를 식별하는 데 도움이 됩니다. 에서 Image Builder 명령을 사용하여 리소스에 태그를 지정하는 방법에 대한 자세한 내용은이 가이드의 리소스 태깅 섹션을 AWS CLI참조하세요.

내용

- 배포 구성에 대한 세부 정보 나열 및 보기
- AMI 배포 구성 생성 및 업데이트
- 컨테이너 이미지의 배포 설정 생성 및 업데이트
- Image Builder로 크로스 계정 AMI 배포 설정
- EC2 시작 템플릿을 사용하여 AMI 배포 구성

배포 구성에 대한 세부 정보 나열 및 보기

이 섹션에서는 EC2 Image Builder 배포 구성에 대한 정보를 찾고 세부 정보를 볼 수 있는 다양한 방법을 설명합니다.

배포 구성 세부 정보

- 콘솔에서 배포 구성 나열
- 콘솔에서 배포 구성 세부 정보 보기
- 에서 배포 나열 AWS CLI
- 에서 배포 구성 세부 정보 가져오기 AWS CLI

콘솔에서 배포 구성 나열

Image Builder 콘솔에서 사용자 계정으로 생성된 배포 구성 목록을 보려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 배포 설정을 선택합니다. 여기에는 사용자 계정에서 생성된 배포 구성 목록이 표시됩니다.
3. 세부 정보를 보거나 새 배포 구성을 만들려면 구성 이름 링크를 선택합니다. 이를 통해 배포 설정의 세부 정보 보기가 열립니다.

> **Note**
>
> 또한 구성 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

콘솔에서 배포 구성 세부 정보 보기

Image Builder 콘솔을 사용하여 특정 배포 구성의 세부 정보를 보려면 콘솔에서 배포 구성 나열에 설명된 단계를 사용하여 검토할 구성을 선택합니다.

배포 세부 정보 페이지에서 다음을 수행할 수 있습니다.

- 배포 구성을 삭제합니다. Image Builder에서 리소스 삭제에 대한 자세한 내용은 오래되거나 사용되지 않는 Image Builder 리소스 삭제(을)를 참조하세요.
- 배포 세부 정보를 편집합니다.

에서 배포 나열 AWS CLI

다음 예제에서는에서 list-distribution-configurations 명령을 사용하여 모든 배포를 나열 AWS CLI 하는 방법을 보여줍니다.

```
aws imagebuilder list-distribution-configurations
```

에서 배포 구성 세부 정보 가져오기 AWS CLI

다음 예제에서는에서 get-distribution-configuration 명령을 사용하여 Amazon 리소스 이름(ARN)을 지정하여 배포 구성의 세부 정보를 AWS CLI 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-distribution-configuration --distribution-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-
distribution-configuration
```

AMI 배포 구성 생성 및 업데이트

이 섹션에서는 Image Builder AMI의 배포 구성을 생성하고 업데이트하는 방법을 다룹니다.

내용

- SSM 출력 파라미터의 사전 조건

- AMI 배포 구성 생성
- AMI 배포 구성 업데이트
- 출력 AMI에 대한 EC2 Fast Launch를 활성화하는 배포 설정 생성
- 에서 출력 VM 디스크에 대한 배포 설정 생성 AWS CLI

SSM 출력 파라미터의 사전 조건

AWS Systems Manager Parameter Store 파라미터(SSM 파라미터)를 설정하는 새 AMI 배포 구성을 생성하기 전에 다음 사전 조건을 충족했는지 확인합니다.

실행 역할

파이프라인을 생성하거나에서 create-image 명령을 사용하는 경우 Image Builder 실행 역할을 하나만 지정할 AWS CLI수 있습니다. Image Builder 워크플로 실행 역할을 정의한 경우 해당 역할에 추가 기능 권한을 추가합니다. 그렇지 않으면 필요한 권한이 포함된 새 사용자 지정 역할을 생성합니다.

- 배포 중에 출력 AMI ID를 SSM 파라미터에 저장하려면 Image Builder 실행 역할에서 `ssm:PutParameter` 작업을 지정하고 파라미터를 리소스로 나열해야 합니다.
- 파라미터 데이터 형식을 로 설정`AWS EC2 Image`하여 파라미터 값을 AMI ID로 검증하도록 Systems Manager에 신호를 보내는 경우 `ec2:DescribeImages` 작업도 추가해야 합니다.

AMI 배포 구성 생성

배포 구성에는 출력 AMI 이름, 암호화에 대한 특정 리전 설정, 시작 권한 AWS 계정, 출력 AMI를 시작할 수 있는 조직 및 조직 단위(OUs) 및 라이선스 구성이 포함됩니다.

배포 구성을 사용하면 출력 AMI의 이름과 설명을 지정하고, 다른 AWS 계정 사용자에게 AMI를 시작하고, AMI를 다른 계정에 복사하고, AMI를 다른 AWS 리전에 복제할 수 있는 권한을 부여할 수 있습니다. 또한 AMI를 Amazon Simple Storage Service(S3)로 내보내거나 출력 Windows AMI를 위한 EC2 Fast Launch를 구성할 수 있습니다. AMI를 공개하려면 시작 권한 승인 계정을 `all`(으)로 설정합니다. EC2 ModifyImageAttribute에서 AMI를 공개하는 예를 참조하세요.

Console

AWS Management Console다음 단계에 따라에서 새 AMI 배포 구성을 생성합니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.

2. 탐색 창에서 배포 설정을 선택합니다. 여기에는 사용자 계정에서 생성된 배포 구성 목록이 표시됩니다.
3. 배포 설정 패널 상단에 있는 배포 설정 만들기를 선택합니다.
4. 이미지 유형 섹션에서 Amazon Machine Image(AMI) 출력 유형을 선택합니다.
5. 일반 섹션에서 배포 구성의 이름과 설명(선택 사항)을 입력합니다.
6. 리전 설정 섹션에서 AMI를 배포하는 각 리전에 대해 다음 세부 정보를 입력합니다.

   a. AMI는 기본적으로 현재 리전(리전 1) 에 배포됩니다. 리전 1은 배포의 소스입니다. 리전 1의 일부 설정은 편집할 수 없습니다. 추가하는 모든 리전의 경우 지역 드롭다운 목록에서 리전을 선택할 수 있습니다.

      Kms 키는 대상 리전에서 이미지의 EBS 볼륨을 암호화하는 데 AWS KMS key 사용되는를 식별합니다. 참고로, 빌드가 소스 리전(리전 1)의 사용자 계정으로 생성한 원본 AMI에는 적용되지 않습니다. 빌드의 배포 단계에서 실행되는 암호화는 다른 계정이나 리전에 배포되는 이미지에만 적용됩니다.

      계정의 소스 지역에 생성된 AMI의 EBS 볼륨을 암호화하려면 이미지 레시피 블록 디바이스 매핑(콘솔의 스토리지(볼륨))에서 KMS 키를 설정해야 합니다.

      Image Builder는 AMI를 리전에 지정한 타겟 계정 에 복사합니다.

      > **전제 조건**
      >
      > 계정 간에 이미지를 복사하려면 모든 배포 대상 계정에서 `EC2ImageBuilderDistributionCrossAccountRole` 역할을 생성하고 Ec2ImageBuilderCrossAccountDistributionAccess 정책 관리형 정책을 역할에 연결해야 합니다.

      출력 AMI 이름은 선택 사항입니다. 이름을 제공하면 최종 출력 AMI 이름에는 AMI 구축 시점의 타임스탬프가 추가됩니다. 이름을 지정하지 않으면 Image Builder에서 레시피 이름에 빌드 타임스탬프를 추가합니다. 이를 통해 각 빌드에 대해 고유한 AMI 이름을 보장합니다.

      i. AMI 공유를 사용하면 지정된 AWS 보안 주체가 AMI에서 인스턴스를 시작할 수 있는 액세스 권한을 부여할 수 있습니다. AMI 공유 섹션을 확장하면 다음 세부 정보를 입력할 수 있습니다.

- 시작 권한 - AMI를 비공개로 유지하고 특정 AWS 보안 주체가 프라이빗 AMI에서 인스턴스를 시작할 수 있도록 액세스를 허용하려면 비공개를 선택합니다. AMI를 퍼블릭으로 설정하려면 퍼블릭을 선택합니다. 모든 AWS 보안 주체는 퍼블릭 AMI에서 인스턴스를 시작할 수 있습니다.
- 보안 주체 - 인스턴스를 시작할 수 있도록 다음 유형의 AWS 보안 주체에 대한 액세스 권한을 부여할 수 있습니다.
  - AWS 계정 - 특정 AWS 계정에 대한 액세스 권한 부여
  - 조직 구성 단위(OU) - OU 및 모든 하위 엔티티에 대한 액세스 권한을 부여합니다. 하위 엔터티에는 OUs 및 AWS 계정이 포함됩니다.
  - 조직 - 사용자 AWS Organizations및 모든 하위 엔터티에 대한 액세스 권한을 부여합니다. 하위 엔터티에는 OUs 및 AWS 계정이 포함됩니다.

    먼저, 보안 주체 유형을 선택합니다. 다음 드롭다운 목록 오른쪽에 있는 상자에 액세스 권한을 부여하려는 AWS 보안 주체의 ID를 입력합니다. 다양한 종류의 ID를 다양하게 입력할 수 있습니다.

ii. 라이선스 구성 섹션을 확장하여 로 생성된 라이선스 구성을 Image Builder 이미지 AWS License Manager 에 연결할 수 있습니다. 라이선스 구성은 기업 계약 조건에 기반한 라이선스 규칙을 포함합니다. Image Builder는 기본 AMI와 연결된 라이선스 구성을 자동으로 포함합니다.

iii. 시작 템플릿 구성 섹션을 확장하여 생성한 AMI에서 인스턴스를 시작하는 데 사용할 EC2 시작 템플릿을 지정할 수 있습니다.

EC2 시작 템플릿을 사용하는 경우 Image Builder에서 빌드 완료 후 최신 AMI ID가 포함된 시작 템플릿의 새 버전을 생성하도록 지시할 수 있습니다. 시작 템플릿을 업데이트하려면 다음과 같이 설정을 구성하세요.

- 시작 템플릿 이름 - Image Builder에서 업데이트하려는 시작 템플릿의 이름을 선택합니다.
- 기본 버전 설정 - 시작 템플릿 기본 버전을 새 버전으로 업데이트하려면 이 확인란을 선택합니다.

다른 시작 템플릿 구성을 추가하려면 시작 템플릿 구성 추가를 선택합니다. 리전 당 최대 5개의 시작 템플릿 구성을 보유할 수 있습니다.

iv. SSM 파라미터 구성 섹션을 확장하여 대상 리전에 배포된 이미지의 출력 AMI ID를 저장하는 SSM 파라미터를 구성할 수 있습니다. 선택적으로 리전에서 배포 계정을 지정할 수 있습니다.

파라미터 이름 - 파라미터의 이름을 입력합니다. 예: `/output/image/param`.

데이터 유형 - 기본값()을 유지합니다`AWS EC2 Image`. 이렇게 하면 Systems Manager가 파라미터 값을 검증하여 유효한 AMI ID인지 확인하도록 지시합니다.

b. 다른 리전의 배포 설정을 추가하려면 리전 추가를 선택합니다.

7. 완료되면 설정 생성을 선택합니다.

AWS CLI

다음 예는 create-distribution-configuration 명령을 사용하여 AMI에 대한 새 배포 구성을 생성하는 방법이 AWS CLI명령을 사용하는 것을 보여줍니다.

1. CLI 입력 JSON 파일 생성

파일 편집 도구를 사용하여 다음 예 중 하나에 표시된 키와 환경에 적합한 값을 포함하는 JSON 파일을 생성합니다. 이 예제에서는 지정된 리전에 배포하는 AMI를 시작할 권한이 있는 AWS 계정 AWS Organizations 조직 단위(OUs)를 정의합니다. 다음 단계에서 사용할 수 있도록 파일 `create-ami-distribution-configuration.json` 의 이름을 지정합니다.

예제 1:에 배포 AWS 계정

이 예에서는 AMI를 두 리전에 배포하고 각 리전에서 시작 권한을 갖도록 AWS 계정 (을)를 지정합니다.

```
{
 "name": "MyExampleAccountDistribution",
 "description": "Copies AMI to eu-west-1, and specifies accounts that can launch
 instances in each Region.",
 "distributions": [
  {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "name": "Name {{imagebuilder:buildDate}}",
    "description": "An example image name with parameter references",
    "amiTags": {
     "KeyName": "Some Value"
```

```
    },
    "launchPermission": {
     "userIds": [
      "987654321012"
     ]
    }
   }
  },
  {
   "region": "eu-west-1",
   "amiDistributionConfiguration": {
    "name": "My {{imagebuilder:buildVersion}} image {{imagebuilder:buildDate}}",
    "amiTags": {
     "KeyName": "Some value"
    },
    "launchPermission": {
     "userIds": [
      "100000000001"
     ]
    }
   }
  }
 ]
}
```

예제 2: 조직 및 OUs에 배포

이 예에서는 AMI를 소스 리전에 배포하고 조직 및 OU 시작 권한을 지정합니다.

```
{
 "name": "MyExampleAWSOrganizationDistribution",
 "description": "Shares AMI with the Organization and OU",
 "distributions": [
  {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "name": "Name {{ imagebuilder:buildDate }}",
    "launchPermission": {
     "organizationArns": [
      "arn:aws:organizations::123456789012:organization/o-myorganization123"
     ],
     "organizationalUnitArns": [
```

```
      "arn:aws:organizations::123456789012:ou/o-123example/ou-1234-
myorganizationalunit"
     ]
    }
   }
  }
 ]
}
```

예제 3: SSM 파라미터에 출력 AMI ID 저장

이 예제에서는 배포 리전의 AWS Systems Manager 파라미터 스토어 파라미터에 출력 AMI ID를 저장합니다.

```
{
 "name": "SSMParameterOutputAMI",
 "description": "Updates an SSM parameter with the output AMI ID for the
 distribution.",
 "distributions": [
  {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "name": "Name {{ imagebuilder:buildDate }}"
   },
   "ssmParameterConfigurations": [
    {
     "amiAccountId": "111122223333",
     "parameterName": "/output/image/param",
     "dataType": "aws:ec2:image"
    }
   ]
  }
 ]
}
```

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder create-distribution-configuration --cli-input-json
 file://create-ami-distribution-configuration.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

자세한 내용은 AWS CLI 명령 참조의 create-distribution-configuration(을)를 참조하세요.

AMI 배포 구성 업데이트

AMI 배포 구성을 변경할 수 있습니다. 하지만 변경 사항은 Image Builder에서 이미 배포한 리소스에는 적용되지 않습니다. 예를 들어 AMI를 리전에 배포한 후 배포에서 제거한 경우 이미 배포된 AMI는 수동으로 제거할 때까지 해당 리전에 남아 있습니다.

AWS Management Console

AWS Management Console다음 단계에 따라의 AMI 배포 구성을 수행합니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 배포 설정을 선택합니다. 여기에는 사용자 계정에서 생성된 배포 구성 목록이 표시됩니다.
3. 세부 정보를 보거나 배포 구성을 업데이트하려면 구성 이름 링크를 선택합니다. 이를 통해 배포 설정의 세부 정보 보기가 열립니다.

   > **Note**
   >
   > 또한 구성 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

4. 배포 구성을 편집하려면 배포 세부 정보 섹션의 오른쪽 상단에서 편집을 선택합니다. 배포 구성 이름, 리전 1로 표시되는 기본 리전 등 일부 필드는 잠겨 있습니다. 배포 설정에 대한 자세한 내용은 AMI 배포 구성 생성 섹션을 참조하세요.
5. 작업을 마쳤으면 변경 사항 저장을 선택합니다.

AWS CLI

다음 예에서는 update-distribution-configuration 명령을 사용하여 AMI의 배포 설정을 업데이트하는 방법이 AWS CLI명령을 사용하는 것으로 보여줍니다.

1. CLI 입력 JSON 파일 생성

   파일 편집 도구를 사용하여 다음 예제에 표시된 키와 환경에 유효한 값을 사용하여 JSON 파일을 생성합니다. 이 예제에서는 `update-ami-distribution-configuration.json`라는 이름의 파일이 사용됩니다.

```
{
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/update-ami-distribution-
configuration.json",
 "description": "Copies AMI to eu-west-2, and specifies accounts that can launch
 instances in each Region.",
 "distributions": [
   {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "name": "Name {{imagebuilder:buildDate}}",
    "description": "An example image name with parameter references",
    "launchPermissions": {
     "userIds": [
      "987654321012"
     ]
    }
   }
  },
  {
   "region": "eu-west-2",
   "amiDistributionConfiguration": {
    "name": "My {{imagebuilder:buildVersion}} image {{imagebuilder:buildDate}}",
    "tags": {
     "KeyName": "Some value"
    },
    "launchPermissions": {
     "userIds": [
      "100000000001"
     ]
    }
   }
```

```
  }
 ]
}
```

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder update-distribution-configuration --cli-input-json
 file://update-ami-distribution-configuration.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

자세한 내용은 AWS CLI 명령 참조의 update-distribution-configuration(을)를 참조하세요. 배포 구성 리소스의 태그를 업데이트하려면 리소스 태깅 섹션을 참조하세요.

출력 AMI에 대한 EC2 Fast Launch를 활성화하는 배포 설정 생성

다음 예는 AWS CLI에서 create-distribution-configuration 명령을 사용하여 AMI에 대해 EC2 Fast Launch가 구성된 배포 설정을 생성하는 방법을 보여줍니다.

> **Note**
>
> Image Builder는 EC2 Fast Launch가 사전 활성화된 AMI에 대한 교차 계정 배포를 지원하지 않습니다. 대상 계정에서 EC2 Fast Launch를 활성화해야 합니다.

1. CLI 입력 JSON 파일 생성

   파일 편집 도구를 사용하여 다음 예와 같은 키와 환경에 적합한 값이 포함된 JSON 파일을 생성합니다.

이 예에서는 최대 병렬 시작 수가 대상 리소스 수보다 크기 때문에 모든 대상 리소스에 대한 인스턴스를 동시에 시작합니다. 이 파일의 이름은 다음 단계에 표시된 명령 예에서 `ami-dist-config-win-fast-launch.json`입니다.

```
{
"name": "WinFastLaunchDistribution",
"description": "An example of Windows AMI EC2 Fast Launch settings in the
 distribution configuration.",
"distributions": [
 {
  "region": "us-west-2",
  "amiDistributionConfiguration": {
   "name": "Name {{imagebuilder:buildDate}}",
   "description": "Includes Windows AMI EC2 Fast Launch settings.",
   "amiTags": {
    "KeyName": "Some Value"
   }
  },
  "fastLaunchConfigurations": [{
   "enabled": true,
   "snapshotConfiguration": {
    "targetResourceCount": 5
   },
   "maxParallelLaunches": 6,
   "launchTemplate": {
    "launchTemplateId": "lt-0ab1234c56d789012",
    "launchTemplateVersion": "1"
    }
  }],
  "launchTemplateConfigurations": [{
       "launchTemplateId": "lt-0ab1234c56d789012",
       "setDefaultVersion": true
    }]
 }]
}
```

> **Note**
>
> `launchTemplate` 섹션에 `launchTemplateId` 대신 `launchTemplateName`(을)를 지정할 수 있지만 이름과 ID를 모두 지정할 수는 없습니다.

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder create-distribution-configuration --cli-input-json file://ami-dist-config-win-fast-launch.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

자세한 내용은 AWS CLI 명령 참조의 create-distribution-configuration(을)를 참조하세요.

에서 출력 VM 디스크에 대한 배포 설정 생성 AWS CLI

다음 예는 create-distribution-configuration 명령을 사용하여 모든 이미지 빌드와 함께 VM 이미지 디스크를 Amazon S3로 내보내는 배포 설정을 생성하는 방법을 보여줍니다.

1. CLI 입력 JSON 파일 생성

   AWS CLI에서 사용하는 create-distribution-configuration 명령을 간소화할 수 있습니다. 이렇게 하려면 명령에 전달하려는 모든 내보내기 구성을 포함하는 JSON 파일을 생성하세요.

   > **Note**
   >
   > JSON 파일의 데이터 값에 대한 이름 지정 규칙은 Image Builder API 작업 요청 파라미터에 지정된 패턴을 따릅니다. API 작업 요청 파라미터를 검토하려면 EC2 Image Builder API 참조의 CreateDistributionConfiguration 명령을 참조하세요.
   > 데이터 값을 명령줄 매개변수로 제공하려면 옵션으로 create-distribution-configuration 명령에 대한 AWS CLI 명령 참조에 지정된 매개변수 이름을 참조하세요.

   다음은 이 예에서 `s3ExportConfiguration` JSON 객체에 지정하는 매개변수에 대한 요약입니다.

- roleName(문자열, 필수) - S3 버킷으로 이미지를 내보낼 수 있는 VM Import/Export 권한을 부여하는 역할의 이름입니다.
- DiskiImageFormat(문자열, 필수) - 업데이트된 디스크 이미지를 지원되는 다음 형식 중 하나로 내보냅니다.
  - 가상 하드 디스크 - Citrix Xen 및 Microsoft Hyper-V 가상화 제품과 호환됩니다.
  - 스트림 최적화 ESX 가상 머신 디스크 – VMware ESX, VMware vSphere 버전 4, 5 및 6과 호환됩니다.
  - 원시 - 원시 형식.
- S3bucket(문자열, 필수) - VM의 출력 디스크 이미지를 저장하는 S3 버킷입니다.

파일을 `export-vm-disks.json`(으)로 저장합니다. create-distribution-configuration 명령에 파일 이름을 사용합니다.

```
{
 "name": "example-distribution-configuration-with-vm-export",
 "description": "example",
 "distributions": [
  {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "description": "example-with-vm-export"

   },
   "s3ExportConfiguration": {
    "roleName": "vmimport",
    "diskImageFormat": "RAW",
    "s3Bucket": "vm-bucket-export"
   }
  }],
 "clientToken": "abc123def4567ab"
}
```

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder create-distribution-configuration --cli-input-json file://export-vm-disks.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

자세한 내용은 AWS CLI 명령 참조의 create-distribution-configuration(을)를 참조하세요.

컨테이너 이미지의 배포 설정 생성 및 업데이트

이 섹션에서는 Image Builder 컨테이너 이미지의 배포 설정 생성 및 업데이트를 다룹니다.

내용

- 에서 Image Builder 컨테이너 이미지에 대한 배포 설정 생성 AWS CLI
- 에서 컨테이너 이미지의 배포 설정 업데이트 AWS CLI

에서 Image Builder 컨테이너 이미지에 대한 배포 설정 생성 AWS CLI

배포 구성을 사용하면 출력 컨테이너 이미지의 이름과 설명을 지정하고 컨테이너 이미지를 다른 AWS 리전에 복제할 수 있습니다. 배포 구성 리소스 및 각 리전 내의 컨테이너 이미지에 별도의 태그를 적용할 수도 있습니다.

1. CLI 입력 JSON 파일 생성

   선호하는 파일 편집 도구를 사용하여 다음 예에 표시된 키와 환경에 유효한 값을 포함하는 JSON 파일을 생성하세요. 이 예제에서는 `create-container-distribution-configuration.json`(이)라는 이름의 파일이 사용됩니다.

```
{
 "name": "distribution-configuration-name",
 "description": "Distributes container image to Amazon ECR repository in two
 regions.",
 "distributions": [
  {
   "region": "us-west-2",
```

```
   "containerDistributionConfiguration": {
    "description": "My test image.",
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
    "containerTags": ["west2", "image1"]
   }
  },
  {
   "region": "us-east-1",
   "containerDistributionConfiguration": {
    "description": "My test image.",
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
      "containerTags": ["east1", "imagedist"]
   }
  }
 ],
 "tags": {
    "DistributionConfigurationTestTagKey1":
 "DistributionConfigurationTestTagValue1",
    "DistributionConfigurationTestTagKey2":
 "DistributionConfigurationTestTagValue2"
 }
}
```

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder create-distribution-configuration --cli-input-json file://create-container-distribution-configuration.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

자세한 내용은 AWS CLI 명령 참조의 create-distribution-configuration(을)를 참조하세요.

에서 컨테이너 이미지의 배포 설정 업데이트 AWS CLI

다음 예제는 update-distribution-configuration 명령을 사용하여 컨테이너 이미지의 배포 설정을 업데이트하는 방법이 AWS CLI명령을 사용하는 것으로 보여줍니다. 각 지역 내의 컨테이너 이미지에 대한 태그를 업데이트할 수도 있습니다.

1. CLI 입력 JSON 파일 생성

   즐겨 사용하는 파일 편집 도구를 사용하여 다음 예에 표시된 키와 환경에 유효한 값이 포함된 JSON 파일을 생성하세요. 이 예제에서는 `update-container-distribution-configuration.json`(이)라는 이름의 파일이 사용됩니다.

```
{
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/update-container-distribution-
configuration.json",
 "description": "Distributes container image to Amazon ECR repository in two
 regions.",
 "distributions": [
  {
   "region": "us-west-2",
   "containerDistributionConfiguration": {
    "description": "My test image.",
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
    "containerTags": ["west2", "image1"]
   }
  },
  {
   "region": "us-east-2",
   "containerDistributionConfiguration": {
    "description": "My test image.",
    "targetRepository": {
     "service": "ECR",
     "repositoryName": "testrepo"
    },
```

```
      "containerTags": ["east2", "imagedist"]
    }
   }
  ]
}
```

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder update-distribution-configuration --cli-input-json file://update-container-distribution-configuration.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

자세한 내용은 AWS CLI 명령 참조의 update-distribution-configuration(을)를 참조하세요. 배포 구성 리소스의 태그를 업데이트하려면 리소스 태깅 섹션을 참조하세요.

Image Builder로 크로스 계정 AMI 배포 설정

이 섹션에서는 Image Builder AMI를 지정한 다른 계정에 전달하도록 배포 설정을 구성하는 방법을 설명합니다.

그러면 대상 계정에서 필요에 따라 AMI를 시작하거나 수정할 수 있습니다.

> **Note**
>
> AWS CLI 이 섹션의 명령 예제에서는 이전에 이미지 레시피 및 인프라 구성 JSON 파일을 생성했다고 가정합니다. 이미지 레시피용 JSON 파일을 생성하려면 를 사용하여 이미지 레시피 생성 AWS CLI(을)를 참조하십시오. 인프라 구성을 위한 JSON 파일을 생성하려면 인프라 구성 생성(을)를 참조하십시오.

계정 간에 AMI 배포를 위한 사전 조건

대상 계정이 Image Builder 이미지에서 인스턴스를 성공적으로 시작할 수 있도록 하려면 모든 리전의 모든 대상 계정에 대해 적절한 권한을 구성해야 합니다.

AWS Key Management Service (AWS KMS)를 사용하여 AMI를 암호화하는 경우 새 이미지를 암호화하는 데 사용되는 AWS KMS key 계정의를 구성해야 합니다.

Image Builder가 암호화된 AMI에 대해 계정 간 배포를 수행하는 경우 원본 계정의 이미지가 해독되어 대상 리전으로 푸시되며, 대상 리전에서 지정된 키를 사용하여 이미지를 다시 암호화합니다. Image Builder는 대상 계정을 대신하여 작동하며 대상 리전에서 생성한 IAM 역할을 사용하기 때문에 해당 계정은 원본 및 대상 리전 모두의 키에 액세스할 수 있어야 합니다.

암호화 키

AWS KMS(을)를 사용하여 이미지를 암호화하는 경우 다음 사전 조건이 필요합니다. IAM 사전 조건은 다음 섹션에서 다룹니다.

소스 계정 요구 사항

- AMI를 구축하고 배포하는 모든 리전의 계정에서 KMS 키를 생성합니다. 기존 키를 사용할 수 있습니다.
- 대상 계정에서 내 키를 사용할 수 있도록 모든 키의 키 정책을 업데이트하십시오.

대상 계정 요구 사항

- 암호화된 AMI를 배포하는 데 필요한 작업을 해당 역할이 수행하도록 허용하는 인라인 정책을 `EC2ImageBuilderDistributionCrossAccountRole`에 추가합니다. IAM 구성 단계는 IAM 정책 사전 조건 섹션을 참조하십시오.

를 사용한 교차 계정 액세스에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 다른 계정의 사용자가 KMS 키를 사용하도록 허용을 AWS KMS참조하세요.

다음과 같이 이미지 레시피에 암호화 키를 지정합니다.

- Image Builder 콘솔을 사용하는 경우 레시피의 스토리지(볼륨) 섹션에 있는 암호화(KMS 별칭) 드롭다운 목록에서 암호화 키를 선택합니다.
- CreateImageRecipe API 작업 또는의 create-image-recipe 명령을 사용하는 경우 `blockDeviceMappings` JSON 입력의 아래 `ebs` 섹션에서 키를 AWS CLI구성합니다.

다음 JSON 스니펫은 이미지 레시피의 암호화 설정을 보여줍니다. 암호화 키를 제공하는 것 외에도 `encrypted` 플래그를 `true`(으)로 설정해야 합니다.

```
{
 ...
 "blockDeviceMappings": [
 {
  "deviceName": "Example root volume",
  "ebs": {
   "deleteOnTermination": true,
   "encrypted": true,
   "iops": 100,
   "kmsKeyId": "image-owner-key-id",
   ...
  },
  ...
 }],
 ...
}
```

IAM 정책

AWS Identity and Access Management (IAM)에서 교차 계정 배포 권한을 구성하려면 다음 단계를 따릅니다.

1. 여러 계정에 분산된 Image Builder AMI를 사용하려면 대상 계정 소유자가 `EC2ImageBuilderDistributionCrossAccountRole`(이)라는 계정에 새 IAM 역할을 생성해야 합니다.
2. 계정 간 배포를 활성화하려면 역할에 Ec2ImageBuilderCrossAccountDistributionAccess 정책(을)를 연결해야 합니다. 관리형 정책에 대한 자세한 내용을 알아보려면 AWS Identity and Access Management 사용 설명서의 관리형 정책과 인라인 정책을 참조하십시오.
3. 원본 계정 ID가 대상 계정의 IAM 역할에 연결된 신뢰 정책에 추가되었는지 확인하십시오. 다음 예제는 소스 계정의 계정 ID를 지정하는 대상 계정의 신뢰 정책을 보여줍니다.

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

```
            "AWS": "arn:aws:iam::444455556666:root"
        },
        "Action": "sts:AssumeRole"
    }]
}
```

리소스 기반 정책에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서를 참조하십시오.

4. 배포하는 AMI가 암호화된 경우 대상 계정 소유자가 KMS 키를 사용할 수 있도록 다음 인라인 정책을 그들 계정의 EC2ImageBuilderDistributionCrossAccountRole에 추가해야 합니다. Principal 섹션에는 계정 번호가 포함되어 있습니다. 이를 통해 Image Builder는를 사용하여 각 리전에 적합한 키를 사용하여 AMI를 암호화하고 복호화할 때 Image Builder를 대신하여 작업 AWS KMS 할 수 있습니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowRoleToPerformKMSOperationsOnBehalfOfTheDestinationAccount",
   "Effect": "Allow",
   "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
   ],
   "Resource": "*"
  }
 ]
}
```

인라인 정책에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서에서 인라인 정책을 참조하십시오.

5. launchTemplateConfigurations(을)를 사용하여 Amazon EC2 시작 템플릿을 지정하는 경우 각 대상 계정의 내 EC2ImageBuilderDistributionCrossAccountRole에 다음 정책도 추가해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateLaunchTemplateVersion",
                "ec2:ModifyLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeLaunchTemplates"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:launch-template/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "EC2 Image Builder"
                }
            }
        }
    ]
}
```

6. AWS Systems Manager 파라미터 스토어 파라미터를 사용하여 배포 계정 및 리전에 대한 출력 AMI의 AMI ID를 저장하는 경우 각 대상 계정의 `EC2ImageBuilderDistributionCrossAccountRole`에 다음 정책을 추가해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages"
            ],
            "Resource": "*"
        }
    ]
}
```

계정 간 분배에 대한 제한

여러 계정에 Image Builder 이미지를 배포할 때는 몇 가지 제한 사항이 있습니다.

- 대상 계정은 각 대상 리전의 동시 AMI 사본 50개로 제한됩니다.
- 반가상화(PV) 가상화 AMI를 다른 리전에 복사하려는 경우 대상 리전이 PV 가상화 AMI를 지원해야 합니다. 자세한 내용은 Linux AMI 가상화 유형을 참조하십시오.
- 암호화된 스냅샷의 암호화되지 않은 사본은 생성할 수 없습니다. `KmsKeyId` 파라미터에 AWS Key Management Service (AWS KMS) 고객 관리 키를 지정하지 않는 경우 Image Builder는 Amazon Elastic Block Store(Amazon EBS)의 기본 키를 사용합니다. 자세한 내용은 Amazon Elastic Compute Cloud 사용 설명서의 Amazon EBS 암호화를 참조하십시오.

자세한 내용은 EC2 Image Builder API 참조의 CreateDistributionConfiguration을 참조하십시오.

콘솔에서 Image Builder AMI에 대한 계정 간 배포 구성

이 섹션에서는 AWS Management Console(을)를 사용하여 Image Builder AMI의 계정 간 배포를 위한 배포 설정을 만들고 구성하는 방법을 설명합니다. 계정 간 배포를 구성하려면 특정 IAM 권한이 필요합니다. 계속하려면 먼저 이 섹션의 계정 간에 AMI 배포를 위한 사전 조건(을)를 완료해야 합니다.

Image Builder 콘솔에서 배포 설정을 생성하려면 다음 단계를 수행하십시오.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 배포 설정을 선택합니다. 여기에는 내 계정에 생성된 배포 설정 목록이 표시됩니다.
3. 배포 설정 페이지 상단에서 배포 설정 생성을 선택합니다. 그러면 배포 설정 생성 페이지로 이동합니다.
4. 이미지 유형 섹션에서 Amazon Machine Image(AMI) 출력 유형을 선택합니다. 이것이 기본 설정입니다.
5. 일반 섹션에서 생성하려는 배포 설정 리소스의 이름을 입력합니다(필수).
6. 리전 설정 섹션에서 선택한 리전의 대상 계정에 AMI를 배포하려는 12자리 계정 ID를 입력하고 엔터 키를 누릅니다. 그러면 형식이 올바른지 확인한 다음 상자 아래에 입력한 계정 ID가 표시됩니다. 이 과정을 반복하여 계정을 더 추가합니다.

   입력한 계정을 제거하려면 계정 ID 오른쪽에 표시된 X를 선택합니다.

   각 리전의 출력 AMI 이름을 입력합니다.
7. 필요한 추가 설정을 계속 지정하고 설정 생성을 선택하여 새 배포 설정 리소스를 생성합니다.

에서 Image Builder AMI에 대한 교차 계정 배포 구성 AWS CLI

이 섹션에서는 배포 설정 파일을 구성하고에서 create-image 명령을 사용하여 Image Builder AMI AWS CLI 를 빌드하고 계정 간에 배포하는 방법을 설명합니다.

계정 간 배포를 구성하려면 특정 IAM 권한이 필요합니다. create-image 명령을 실행하려면 먼저 이 섹션의 계정 간에 AMI 배포를 위한 사전 조건(을)를 완료해야 합니다.

1. 배포 설정 파일 구성

   에서 create-image 명령을 사용하여 다른 계정에 배포되는 Image Builder AMI를 AWS CLI 생성하기 전에 `AmiDistributionConfiguration` 설정에서 대상 계정 IDs를 지정하는 `DistributionConfiguration` JSON 구조를 생성해야 합니다. 원본 리전에 `AmiDistributionConfiguration`(을)를 하나 이상 지정해야 합니다.

`create-distribution-configuration.json`(으)로 이름 붙인 다음 예제 파일은 해당 원본 리전의 계정 간 이미지 배포에 대한 구성을 보여줍니다.

```
{
 "name": "cross-account-distribution-example",
 "description": "Cross Account Distribution Configuration Example",
 "distributions": [
  {
   "amiDistributionConfiguration": {
    "targetAccountIds": ["123456789012", "987654321098"],
    "name": "Name {{ imagebuilder:buildDate }}",
    "description": "ImageCopy Ami Copy Configuration"
   },
   "region": "us-west-2"
  }
 ]
}
```

2. 배포 설정 생성

의 create-distribution-configuration 명령을 사용하여 Image Builder 배포 설정 리소스를 생성하려면 명령에 다음 파라미터를 AWS CLI제공합니다.

- `--name` 파라미터에 배포 이름을 입력합니다.
- `--cli-input-json` 파라미터에서 생성한 배포 구성 JSON 파일을 첨부합니다.

```
aws imagebuilder create-distribution-configuration --name my distribution name --
cli-input-json file://create-distribution-configuration.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

`--distributions` 파라미터를 사용하여 명령에 직접 JSON을 제공할 수도 있습니다.

EC2 시작 템플릿을 사용하여 AMI 배포 구성

대상 계정 및 리전에서 Image Builder AMI를 일관되게 시작할 수 있도록 배포 설정에서 `launchTemplateConfigurations`(을)를 사용하여 Amazon EC2 시작 템플릿을 지정할 수 있습니다. 배포 프로세스 중에 `launchTemplateConfigurations`(이)가 있는 경우 Image Builder는 템플릿의 원래 설정과 빌드의 새 AMI ID를 모두 포함하는 새 버전의 시작 템플릿을 생성합니다. 시작 템플릿을 사용하여 EC2 인스턴스를 시작하는 방법에 대한 자세한 내용은 대상 운영 체제에 따라 다음 링크 중 하나를 참조하십시오.

- 시작 템플릿에서 Linux 인스턴스 시작하기
- 시작 템플릿에서 Windows 인스턴스 시작하기

> **Note**
>
> 이미지에 Windows Fast Launch를 활성화하기 위한 시작 템플릿을 포함하는 경우 Image Builder가 사용자를 대신하여 Windows Fast Launch를 활성화할 수 있도록 시작 템플릿에 다음 태그가 포함되어야 합니다.
>
> `CreatedBy: EC2 Image Builder`

콘솔에서 AMI 배포 설정에 EC2 시작 템플릿 추가

출력 AMI와 함께 시작 템플릿을 제공하려면 콘솔에서 다음 단계를 따릅니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 배포 설정을 선택합니다. 여기에는 내 계정에 생성된 배포 설정 목록이 표시됩니다.
3. 배포 설정 페이지 상단에서 배포 설정 생성을 선택합니다. 그러면 배포 설정 생성하기 페이지가 열립니다.
4. 이미지 유형 섹션에서 Amazon Machine Image(AMI) 출력 유형을 선택합니다. 이것이 기본 설정입니다.
5. 일반 섹션에서 생성하려는 배포 설정 리소스의 이름을 입력합니다(필수).
6. 리전 설정 섹션에서 목록으로부터 EC2 시작 템플릿의 이름을 선택합니다. 계정에 시작 템플릿이 없는 경우 새 시작 템플릿 생성하기를 선택합니다. 그러면 EC2 대시보드에서 시작 템플릿이 열립니다.

시작 템플릿 기본 버전을 Image Builder가 출력 AMI를 사용하여 생성한 새 버전으로 업데이트하려면 기본 버전 설정하기 확인란을 선택합니다.

선택한 리전에 다른 시작 템플릿을 추가하려면 새 템플릿 구성 추가하기를 선택합니다.

시작 템플릿을 제거하려면 제거하기를 선택합니다.

7. 필요한 추가 설정을 계속 지정하고 설정 생성을 선택하여 새 배포 설정 리소스를 생성합니다.

에서 AMI 배포 설정에 EC2 시작 템플릿 추가 AWS CLI

이 섹션에서는 시작 템플릿으로 배포 설정 파일을 구성하고 AWS CLI 의 create-image 명령을 사용하여 Image Builder AMI와 이를 사용하는 새 버전의 시작 템플릿을 빌드 및 배포하는 방법을 설명합니다.

1. 배포 설정 파일 구성

   시작 템플릿을 사용하여 Image Builder AMI를 생성하려면 먼저 `launchTemplateConfigurations` 설정을 지정하는 배포 구성 JSON 구조를 생성 AWS CLI해야 합니다. 소스 리전에 하나 이상의 `launchTemplateConfigurations` 항목을 지정해야 합니다.

   `create-distribution-config-launch-template.json`(으)로 이름이 지정된 다음 샘플 파일은 소스 리전의 시작 템플릿 구성에 대한 몇 가지 가능한 시나리오를 보여줍니다.

```
{
    "name": "NewDistributionConfiguration",
    "description": "This is just a test",
    "distributions": [
        {
            "region": "us-west-2",
            "amiDistributionConfiguration": {
                "name": "test-{{imagebuilder:buildDate}}-
{{imagebuilder:buildVersion}}",
                "description": "description"
            },
            "launchTemplateConfigurations": [
                {
                    "launchTemplateId": "lt-0a1bcde2fgh34567",
                    "accountId": "935302948087",
                    "setDefaultVersion": true
                },
```

```
                {
                    "launchTemplateId": "lt-0aaa1bcde2ff3456"
                },
                {
                    "launchTemplateId": "lt-12345678901234567",
                    "accountId": "123456789012"
                }
            ]
        }
    ],
    "clientToken": "clientToken1"
}
```

2. 배포 설정 생성

   의 create-distribution-configuration 명령을 사용하여 Image Builder 배포 설정 리소스를 생성하려면 명령에 다음 파라미터를 AWS CLI제공합니다.

   - `--name` 파라미터에 배포 이름을 입력합니다.
   - `--cli-input-json` 파라미터에서 생성한 배포 구성 JSON 파일을 첨부합니다.

```
aws imagebuilder create-distribution-configuration --name my distribution name--
cli-input-json file://create-distribution-config-launch-template.json
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

`--distributions` 파라미터를 사용하여 명령에 직접 JSON을 제공할 수도 있습니다.

와 Image Builder 리소스 공유 AWS RAM

EC2 Image Builder는 AWS Resource Access Manager (AWS RAM)와 통합되어 다음과 같은 유형의 Image Builder 리소스를 임의의 AWS 계정 또는를 통해 공유할 수 있습니다 AWS Organizations.

- Components
- 이미지
- 레시피

리소스를 공유하려면 리소스 공유를 생성 AWS RAM해야 합니다. 리소스 공유는 공유할 리소스와 공유 대상 소비자를 지정합니다. 소비자는 개인 AWS 계정, 조직 단위 또는 전체 조직일 수 있습니다 AWS Organizations. 다음 목록에는 공유할 수 있는 계정 및 조직의 유형이 포함되어 있습니다.

- 에서 조직 AWS 계정 내부 또는 외부에 특정합니다 AWS Organizations.
- AWS Organizations의 조직 내부의 조직 단위(OU).
- AWS Organizations의 전체 조직.
- AWS Organizations 또는 조직 외부의 OUs

이 모델에서는 리소스를 소유 AWS 계정 한(소유자)가 동일한 리전 내의 다른 AWS 계정 또는를 통해 AWS Organizations (소비자) 리소스를 공유합니다. 공유 리소스가 업데이트되면 소비자는 이러한 업데이트를 자동으로 받습니다.

> **Note**
>
> 공유 구성 요소, 이미지 및 이미지 레시피는 소유자의 리소스 제한에 대해서만 계산됩니다. 소비자의 리소스 제한은 소비자에게 공유된 리소스의 영향을 받지 않습니다.

주제

- 리소스 소유자
- 리소스 소비자
- Image Builder AWS RAM 리소스에 대한 리소스 공유 생성
- 에서 Image Builder 리소스 공유 해제 AWS RAM

리소스 소유자

Image Builder 리소스는 생성된 AWS 리전 에서만 공유할 수 있습니다. 이러한 리소스를 공유하면 리전 간에 복제되지 않습니다.

소유하고 공유할 수 있는 Image Builder 리소스 목록을 가져오려면 콘솔에서 소유권 필터를 지정하거나 AWS CLI에서 명령을 실행할 때 지정합니다.

- Image Builder 구성 요소 나열
- 이미지 나열
- 이미지 레시피 세부 정보 나열 및 보기
- 컨테이너 레시피 세부 정보 나열 및 보기

에 대한 자세한 내용은 AWS RAM 사용 설명서를 AWS RAM참조하세요.

Image Builder 리소스를 공유하기 위한 사전 조건

구성 요소, 이미지 또는 레시피 같은 Image Builder 리소스를 공유하려면:

- 는 공유하려는 Image Builder 리소스를 소유해야 AWS 계정 합니다. 남이 내게 공유한 리소스를 내가 다른 사람과 공유할 수는 없습니다.
- 암호화된 리소스와 연결된 AWS Key Management Service (AWS KMS) 키는 대상 계정, 조직 또는 OUs와 명시적으로 공유해야 합니다.
- 를 사용하여 Image Builder 리소스를 AWS Organizations 및 OUs와 공유하려면 공유를 활성화 AWS RAM해야 합니다. 자세한 내용은 AWS RAM 사용 설명서의 AWS Organizations과(와) 공유 활성화를 참조하세요.
- 다른 리전의 계정 AWS KMS 간에 로 암호화된 이미지를 배포하는 경우 각 대상 리전에서 KMS 키와 별칭을 생성해야 합니다. 또한 해당 리전에서 인스턴스를 시작할 사용자는 키 정책을 통해 지정된 KMS 키에 액세스해야 합니다.

Image Builder가 파이프라인 빌드에서 생성하는 다음 리소스는 Image Builder 리소스로 간주되지 않습니다. 대신 Image Builder가 계정에 배포하는 외부 리소스이며 배포 구성에서 지정한 AWS 리전, 계정, 조직 또는 조직 단위(OUs)에 배포됩니다.

- Amazon Machine Image(AMI)
- Amazon ECR에 있는 컨테이너 이미지

내 AMI의 배포 설정에 관한 자세한 내용은 AMI 배포 구성 생성 및 업데이트 섹션을 참조하세요. Amazon ECR의 컨테이너 이미지 배포 설정에 대한 자세한 내용은 컨테이너 이미지의 배포 설정 생성 및 업데이트 섹션을 참조하세요.

AMI를 AWS Organizations 및 OUs와 공유하는 방법에 대한 자세한 내용은 조직 또는 OUs.

리소스 소비자

소비자는 공유 리소스를 사용할 수 있지만 어떤 식으로든 수정할 수 없습니다. Image Builder 레시피를 생성할 때 공유 이미지를 기본 이미지로 지정하고 공유 구성 요소를 추가할 수 있습니다. 또한 Image Builder 이미지 파이프라인을 만들 때 또는 AWS CLI에서 create-image 명령을 사용할 때 공유 레시피를 지정할 수 있습니다.

의 조직에 속 AWS Organizations해 있고 조직 내 공유가 활성화된 경우 조직의 소비자에게 공유 리소스에 대한 액세스 권한이 자동으로 부여됩니다. 그렇지 않으면 소비자는 리소스 공유에 가입하라는 초대장을 받고 초대를 수락한 후 공유된 리소스의 액세스 권한을 받습니다.

Image Builder AWS RAM 리소스에 대한 리소스 공유 생성

Image Builder 구성 요소, 이미지 또는 레시피를 공유하려면 AWS Resource Access Manager 리소스 공유에 추가해야 합니다. 리소스 공유는 공유할 리소스와 공유 대상 소비자를 지정합니다.

리소스 공유에 사용할 수 있는 옵션은 다음과 같습니다.

옵션 1: RAM 리소스 공유 생성

RAM 리소스 공유를 생성하면 소유한 구성, 이미지 또는 레시피를 단번에 공유할 수 있습니다. 리소스 공유를 생성하려면 다음 방법 중 하나를 사용합니다.

- 콘솔

  AWS RAM 콘솔을 사용하여 리소스 공유를 생성하려면 AWS RAM 사용 설명서의 자신이 소유한 AWS 리소스 공유를 참조하세요.

- AWS CLI

  AWS RAM 명령줄 인터페이스를 사용하여 리소스 공유를 생성하려면에서 create-resource-share 명령을 실행합니다 AWS CLI.

옵션 2: 리소스 정책을 적용하고 기존 리소스 공유로 승격하기

리소스를 공유하는 두 번째 옵션에는 두 단계, 즉 둘 다에 AWS CLI 대해에서 명령을 실행하는 두 단계가 포함됩니다. 첫 번째 단계에서는의 Image Builder 명령을 사용하여 리소스 기반 정책을 공유 리소스에 AWS CLI 적용합니다. 두 번째 단계에서는의 promote-resource-share-created-from-policy AWS RAM 명령을 사용하여 리소스를 RAM 리소스 공유로 승격 AWS CLI 하여 리소스를 공유한 모든 보안 주체가 리소스를 볼 수 있도록 합니다.

1. 리소스 정책 적용

   리소스 정책을 성공적으로 적용하려면 공유를 진행하는 계정에 기본 리소스에 액세스할 수 있는 권한이 있는지 확인해야 합니다.

   해당 명령에 대한 리소스 유형과 일치하는 탭을 선택합니다.

   Image

   이미지에 리소스 정책을 적용하여 다른 사람들이 자기 레시피의 기본 이미지로 사용하도록 허용할 수 있습니다.

   에서 put-image-policy Image Builder 명령을 실행 AWS CLI하여 이미지를 공유할 AWS 보안 주체를 식별합니다.

   ```
   aws imagebuilder put-image-policy --image-arn arn:aws:imagebuilder:us-
   west-2:123456789012:image/my-example-image/2019.12.03/1 --policy
    '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal":
    { "AWS": [ "123456789012" ] }, "Action": ["imagebuilder:GetImage",
    "imagebuilder:ListImages"], "Resource": [ "arn:aws:imagebuilder:us-
   west-2:123456789012:image/my-example-image/2019.12.03/1" ] } ] }'
   ```

   Component

   빌드 또는 테스트 구성 요소에 리소스 정책을 적용하여 계정 간 공유를 활성화할 수 있습니다. 이 명령어는 다른 계정에 자기 레시피에서 구성 요소를 사용할 권한을 부여합니다. 리소스 정책을 성공적으로 적용하려면 공유를 진행하는 계정에 프라이빗 리포지토리에서 호스팅되는 파일 등 공유 구성 요소에서 참조하는 모든 리소스에 액세스할 권한이 있는지 확인해야 합니다.

   에서 put-component-policy Image Builder 명령을 실행 AWS CLI하여 구성 요소를 공유할 AWS 보안 주체를 식별합니다.

```
aws imagebuilder put-component-policy --component-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.03/1 --policy
 '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal":
 { "AWS": [ "123456789012" ] }, "Action": [ "imagebuilder:GetComponent",
 "imagebuilder:ListComponents" ], "Resource": [ "arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.03/1" ] } ] }'
```

Image recipe

이미지 레시피에 리소스 정책을 적용하여 계정 간 공유를 활성화할 수 있습니다. 이 명령은 다른 계정에 레시피를 사용하여 자신의 계정에 이미지를 생성할 수 있는 권한을 부여합니다. 리소스 정책을 성공적으로 적용하려면 공유를 진행하는 계정에 기본 이미지, 선택한 구성 요소 등 레시피가 참조하는 모든 리소스에 액세스할 수 있는 권한이 있는지 확인해야 합니다.

에서 put-image-recipe-policy Image Builder 명령을 실행 AWS CLI하여 이미지를 공유할 AWS 보안 주체를 식별합니다.

```
aws imagebuilder put-image-recipe-policy --image-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-
image-recipe/2019.12.03 --policy '{ "Version": "2012-10-17", "Statement":
 [ { "Effect": "Allow", "Principal": { "AWS": [ "123456789012" ] }, "Action":
 [ "imagebuilder:GetImageRecipe", "imagebuilder:ListImageRecipes" ], "Resource":
 [ "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-image-
recipe/2019.12.03" ] } ] }'
```

Container recipe

컨테이너 레시피에 리소스 정책을 적용하여 계정 간 공유를 활성화할 수 있습니다. 이 명령은 다른 계정에 레시피를 사용하여 자신의 계정에 이미지를 생성할 수 있는 권한을 부여합니다. 리소스 정책을 성공적으로 적용하려면 공유를 진행하는 계정에 기본 이미지, 선택한 구성 요소 등 레시피가 참조하는 모든 리소스에 액세스할 수 있는 권한이 있는지 확인해야 합니다.

에서 put-container-recipe-policy Image Builder 명령을 실행 AWS CLI하여 이미지를 공유할 AWS 보안 주체를 식별합니다.

```
aws imagebuilder put-container-recipe-policy --container-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-example-
container-recipe/2021.12.03 --policy '{ "Version": "2012-10-17", "Statement":
 [ { "Effect": "Allow", "Principal": { "AWS": [ "123456789012" ] }, "Action":
 [ "imagebuilder:GetContainerRecipe", "imagebuilder:ListContainerRecipes" ],
```

```
 "Resource": [ "arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-
example-container-recipe/2021.12.03" ] } ] }'
```

> **Note**
>
> 리소스 공유 및 공유 해제에 대한 올바른 정책을 설정하려면 리소스 소유자에게 `imagebuilder:put*` 권한이 있어야 합니다.

2. RAM 리소스 공유로 승격하기

   리소스를 공유한 모든 보안 주체가 리소스를 볼 수 있도록 하려면에서 promote-resource-share-created-from-policy AWS RAM 명령을 실행합니다 AWS CLI.

에서 Image Builder 리소스 공유 해제 AWS RAM

공유 구성 요소, 이미지 또는 레시피와 같이 소유한 Image Builder 리소스를 공유 해제하려면 AWS Resource Access Manager 리소스 공유에서 제거해야 합니다. 이를 위해 AWS RAM 콘솔이나 AWS CLI을(를) 사용할 수 있습니다.

> **Note**
>
> 소유자는 공유 리소스가 더 이상 공유되지 않을 때까지 공유 리소스를 삭제할 수 없습니다. 소유자는 리소스에 의존하는 소비자가 아무도 없어야만 이러한 리소스의 공유를 취소할 수 있습니다.

AWS Resource Access Manager 콘솔을 사용하여 내가 소유한 공유 구성 요소, 이미지 또는 레시피를 공유 해제하려면

AWS RAM 사용 설명서에서 리소스 공유 업데이트를 참조하세요.

AWS CLI을 사용하여 내가 소유한 공유 구성 요소, 이미지 또는 레시피를 공유 해제하려면

disassociate-resource-share 명령을 사용하여 그 리소스의 공유를 중단하세요.

Image Builder 출력 리소스 태그 지정

리소스에 태그를 지정하면 리소스 비용 또는 기타 카테고리를 필터링하고 추적하는 데 유용할 수 있습니다. 태그에 따라 액세스를 제어할 수도 있습니다. 태그 기반 권한 부여에 관한 자세한 내용은 Image Builder 태그 기반 권한 부여(을)를 참조하세요.

Image Builder는 다음과 같은 다이내믹 태그를 지원합니다.

- `- {{imagebuilder:buildDate}}`

  빌드 시점의 빌드 날짜/시간으로 확인되는 태그.
- `- {{imagebuilder:buildVersion}}`

  Image Builder Amazon 리소스 이름(ARN) 끝에 있는 번호인 빌드 버전으로 확인됩니다. 예를 들어, `"arn:aws:imagebuilder:us-west-2:123456789012:component/myexample-component/2019.12.02/1"`(은)는 빌드 버전을 1(으)로 표시합니다.

배포한 Amazon Machine Image(AMI)를 추적하는 데 도움이 되도록 Image Builder는 출력 AMI에 다음 태그를 자동으로 추가합니다.

- `"CreatedBy":"EC2 Image Builder"`
- `"Ec2ImageBuilderArn":"arn:aws:imagebuilder:`*`us-west-2`*`:`*`123456789012`*`:image/`*`simple-recipe-linux`*`/`*`1.0.0/10`*`"`. 이 태그에는 AMI를 생성하는 데 사용된 Image Builder 이미지 리소스의 ARN이 포함되어 있습니다.

내용

- 에서 리소스에 태그 지정 AWS CLI
- 에서 리소스의 태그를 해제합니다. AWS CLI
- 에서 특정 리소스에 대한 모든 태그 나열 AWS CLI

에서 리소스에 태그 지정 AWS CLI

다음 예제는 EC2 Image Builder에서 imagebuilder CLI 명령을 사용하여 리소스를 추가하고 태그를 지정하는 방법을 보여줍니다. 적용할 `resourceArn` 및 태그를 제공해야 합니다.

예제 `tag-resource.json`의 내용은 다음과 같습니다.

```
{
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-pipeline",
    "tags": {
        "KeyName": "KeyValue"
    }
}
```

이전 tag-resource.json 파일을 참조하는 다음 명령을 실행합니다.

```
aws imagebuilder tag-resource --cli-input-json file://tag-resource.json
```

에서 리소스의 태그를 해제합니다. AWS CLI

다음 예제는 imagebuilder CLI 명령을 사용하여 리소스에서 태그를 제거하는 방법을 보여줍니다. 태그를 제거하려면 resourceArn 및 키를 제공해야 합니다.

예제 untag-resource.json의 내용은 다음과 같습니다.

```
{
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-pipeline",
    "tagKeys": [
        "KeyName"
    ]
}
```

이전 untag-resource.json 파일을 참조하는 다음 명령을 실행합니다.

```
aws imagebuilder untag-resource --cli-input-json file://untag-resource.json
```

에서 특정 리소스에 대한 모든 태그 나열 AWS CLI

다음 예제는 imagebuilder CLI 명령을 사용하여 특정 리소스의 모든 태그를 나열하는 방법을 보여줍니다.

```
aws imagebuilder list-tags-for-resource --resource-arn arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-pipeline
```

오래되거나 사용되지 않는 Image Builder 리소스 삭제

Image Builder 환경도 가정과 마찬가지로 정기적인 유지 관리가 필요합니다. 그래야 혼잡함 없이 필요한 것을 찾고 작업을 완료할 수 있습니다. 테스트용으로 만든 임시 리소스를 정기적으로 정리하십시오. 그렇지 않으면 해당 리소스를 잊어버리고 나중에는 해당 리소스가 어디에 사용되었는지 기억하지 못할 수 있습니다. 그때쯤이면 안전하게 제거할 수 있을지 확실하지 않을 수도 있습니다.

리소스를 삭제해도 이미지 빌드 프로세스 중에 생성된 Amazon EC2 AMI 또는 Amazon ECR 컨테이너 이미지는 삭제되지 않습니다. 적절한 Amazon EC2 또는 Amazon ECR 콘솔 작업, API 또는 AWS CLI 명령을 사용하여 별도로 정리해야 합니다.

> **Tip**
>
> 리소스를 삭제할 때 종속성 오류를 방지하려면 다음 순서대로 리소스를 삭제하십시오.
>
> 1. 이미지 파이프라인
> 2. 이미지 레시피
> 3. 나머지 모든 리소스

에서 리소스 삭제 AWS Management Console

이미지 파이프라인 및 해당 리소스를 삭제하려면 다음 단계를 따릅니다.

파이프라인 삭제

1. 계정에서 생성된 빌드 파이프라인 목록을 보려면 탐색 창에서 이미지 파이프라인을 선택하십시오.
2. 삭제할 파이프라인을 선택하려면 파이프라인 이름 옆의 확인란을 선택합니다.
3. 이미지 파이프라인 패널 상단의 작업 메뉴에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 `Delete`을(를) 입력한 후 삭제를 선택합니다.

레시피 삭제

1. 계정에서 생성된 레시피 목록을 보려면 탐색 창에서 이미지 레시피를 선택합니다.
2. 삭제할 레시피를 선택하려면 레시피 이름 옆의 확인란을 선택합니다.

3. 이미지 레시피 패널 상단의 작업 메뉴에서 레시피 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

인프라 구성 삭제

1. 계정에서 생성된 인프라 구성 목록을 보려면 탐색 창에서 인프라 구성을 선택하십시오.
2. 구성 이름 옆의 확인란을 선택하여 삭제할 인프라 구성을 선택합니다.
3. 인프라 구성 패널 상단에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

배포 설정 삭제

1. 계정에서 생성된 배포 설정 목록을 보려면 탐색 창에서 배포 설정을 선택합니다.
2. 구성 이름 옆의 확인란을 선택하여 이 자습서를 위해 만든 배포 설정을 선택합니다.
3. 배포 설정 패널 상단에서 삭제를 선택합니다.
4. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

이미지 삭제하기

1. 계정에서 생성된 이미지 목록을 보려면 탐색 창에서 이미지를 선택합니다.
2. 제거하려는 이미지의 이미지 버전을 선택합니다. 그러면 이미지 빌드 버전 페이지가 열립니다.
3. 삭제하려는 이미지의 버전 옆 확인란을 선택합니다. 한 번에 이미지 버전을 여러 개 선택할 수 있습니다.
4. 이미지 빌드 버전 패널 상단에서 버전 삭제를 선택합니다.
5. 삭제를 확인하려면 박스에 Delete을(를) 입력한 후 삭제를 선택합니다.

에서 이미지 파이프라인 삭제 AWS CLI

다음 예제에서는 AWS CLI(을)를 사용하여 Image Builder 리소스를 삭제하는 방법을 보여줍니다. 앞서 언급했듯이 종속성 오류를 방지하려면 리소스를 다음 순서대로 삭제해야 합니다.

1. 이미지 파이프라인
2. 이미지 레시피
3. 나머지 모든 리소스

에서 이미지 파이프라인 삭제 AWS CLI

다음 예제에서는 ARN을 지정하여 이미지 파이프라인을 삭제하는 방법을 보여줍니다.

```
aws imagebuilder delete-image-pipeline --image-pipeline-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

에서 이미지 레시피 삭제 AWS CLI

다음 예제에서는 ARN을 지정하여 이미지 레시피를 삭제하는 방법을 보여줍니다.

```
aws imagebuilder delete-image-recipe --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2019.12.03
```

인프라 구성 삭제하기

다음 예제에서는 ARN을 지정하여 인프라 구성 리소스를 삭제하는 방법을 보여줍니다.

```
aws imagebuilder delete-infrastructure-configuration --infrastructure-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-
infrastructure-configuration
```

배포 설정 삭제

다음 예제에서는 ARN을 지정하여 배포 설정 리소스를 삭제하는 방법을 보여줍니다.

```
aws imagebuilder delete-distribution-configuration --distribution-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-
distribution-configuration
```

이미지 삭제하기

다음 예제에서는 ARN을 지정하여 이미지 빌드 버전을 삭제하는 방법을 보여줍니다.

```
aws imagebuilder delete-image --image-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-image/2019.12.02/1
```

구성 요소 삭제하기

다음 예제는 ARN을 지정하여 구성 요소 빌드 버전을 삭제하기 위해 imagebuilder CLI 명령을 사용하는 방법을 보여줍니다.

```
aws imagebuilder delete-component --component-build-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-
component/2019.12.02/1
```

Important

삭제하기 전에 어떤 식으로든 구성 요소 빌드 버전을 참조하는 레시피가 없는지 확인합니다. 그렇지 않으면 파이프라인 오류가 발생할 수 있습니다.

Image Builder 이미지의 빌드 및 테스트 워크플로 관리

이미지 워크플로는 EC2 Image Builder가 이미지 생성 프로세스의 빌드 및 테스트 단계에서 수행하는 일련의 단계를 정의합니다. 이는 전체 Image Builder 워크플로 프레임워크의 일부입니다.

이미지 워크플로의 이점

- 이미지 워크플로를 사용하면 이미지 생성 프로세스에서의 유연성, 가시성, 제어력이 향상됩니다.
- 워크플로 문서를 정의할 때 사용자 정의된 워크플로 단계를 추가하거나 Image Builder 기본 워크플로를 사용하도록 선택할 수 있습니다.
- 기본 이미지 워크플로에 포함된 워크플로 단계를 제외할 수 있습니다.
- 빌드 프로세스를 완전히 건너뛰는 테스트 전용 워크플로를 만들 수 있습니다. 동일한 작업으로 빌드 전용 워크플로를 만들 수 있습니다.

> **Note**
>
> 기존 워크플로는 수정할 수 없지만 복제하거나 새 버전을 만들 수 있습니다.

워크플로 주제

- 워크플로 프레임워크: 단계
- 서비스 액세스
- 이미지에 관리형 워크플로 사용
- 이미지 워크플로 나열
- 이미지 워크플로 생성
- YAML 워크플로 문서 생성

워크플로 프레임워크: 단계

이미지 워크플로를 사용자 지정하려면 이미지 생성 워크플로 프레임워크를 구성하는 워크플로 단계를 이해해야 합니다.

이미지 생성 워크플로 프레임워크에는 다음과 같은 개별 단계가 포함됩니다.

1. 빌드 단계(스냅샷 이전)-빌드 단계에서 기본 이미지를 실행하는 Amazon EC2 빌드 인스턴스를 변경하여 새 이미지의 베이스라인을 생성합니다. 예를 들어, 레시피에 애플리케이션을 설치하거나 운영 체제 방화벽 설정을 수정하는 구성 요소가 포함될 수 있습니다.

   이 단계가 성공적으로 완료되면 Image Builder는 테스트 단계 및 이후 단계에서 사용할 스냅샷 또는 컨테이너 이미지를 생성합니다.
2. 테스트 단계(스냅샷 이후)-테스트 단계에서 AMI를 생성하는 이미지와 컨테이너 이미지 사이에 약간의 차이가 있습니다. AMI 워크플로의 경우 Image Builder는 빌드 단계의 마지막 단계로 생성한 스냅샷에서 EC2 인스턴스를 시작합니다. 새 인스턴스에서 테스트를 실행하여 설정을 검증하고 인스턴스가 예상대로 작동하는지 확인합니다. 컨테이너 워크플로의 경우 테스트는 구축에 사용된 것과 동일한 인스턴스에서 실행됩니다.

워크플로 프레임워크에는 배포 단계도 포함됩니다. 하지만 Image Builder가 해당 단계의 워크플로를 처리합니다.

서비스 액세스

이미지 워크플로를 실행하려면 Image Builder에 워크플로 작업을 수행할 권한이 있어야 합니다. 다음과 같이 AWSServiceRoleForImageBuilder 서비스 연결 역할을 지정하거나 서비스 액세스를 위한 사용자 지정 역할을 지정할 수 있습니다.

- 콘솔-파이프라인 마법사 3단계 이미지 생성 프로세스 정의에서 서비스 액세스 패널의 IAM 역할 목록에서 서비스 연결 역할 또는 사용자 지정 역할을 선택합니다.
- Image Builder API-CreateImage 작업 요청에서 서비스 연결 역할 또는 사용자 지정 역할을 `executionRole` 파라미터 값으로 지정합니다.

서비스 역할을 생성하는 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 서비스에 권한을 위임할 역할 생성을 참조하세요.

이미지에 관리형 워크플로 사용

관리형 워크플로는에서 생성하고 유지 관리합니다 AWS. 이미지 파이프라인에서 관리형 워크플로를 사용하거나 일회성 이미지 생성을 위해 사용할 관리형 워크플로의 Amazon 리소스 이름(ARN)을 선택할 수 있습니다. Amazon은 패치 및 기타 업데이트가 적용된 최신 버전을 제공합니다. 관리형 워크플로 목록을 가져오려면 섹션을 참조이미지 워크플로 나열하고 소유자 = Amazon(콘솔)을 기준으로 필터링합니다.

이미지 워크플로 나열

Image Builder 콘솔의 이미지 워크플로 목록 페이지에서 소유하고 있거나 액세스할 수 있는 이미지 워크플로 리소스의 목록과 이러한 리소스에 대한 몇 가지 주요 세부 정보를 확인할 수 있습니다. Image Builder API, SDKs 또는에서 명령 또는 작업을 사용하여 계정의 이미지 워크플로를 나열 AWS CLI 할 수도 있습니다.

다음 방법 중 하나를 사용하여 소유하고 있거나 엑세스 권한이 있는 이미지 워크플로 리소스를 나열할 수 있습니다. API 작업에 대한 내용은 EC2 Image Builder API 참조의 ListWorkflows를 참조하세요. 연결된 SDK 요청의 경우, 같은 페이지의 참고 항목 링크를 참조하세요.

다음 세부 정보를 기준으로 필터링하여 결과 목록을 간소화할 수 있습니다. 예를 들어 콘솔에서 소유자 = Amazon을 기준으로 필터링하면 목록에 AWS 관리형 워크플로만 표시됩니다.

- 워크플로
- 버전
- 유형
- 소유자

Console

워크플로 세부 정보

Image Builder 콘솔의 이미지 워크플로 목록 페이지에 있는 세부 정보는 다음과 같습니다.

- 워크플로-이미지 워크플로 리소스의 최신 버전 이름입니다. Image Builder 콘솔에서 워크플로 열은 워크플로 세부 정보 페이지로 연결됩니다.
- 버전-이미지 워크플로 리소스의 최신 버전입니다.
- 유형-워크플로 유형 `BUILD` 또는`TEST`입니다.
- 소유자-워크플로 리소스의 소유자입니다.
- 생성 시간-Image Builder에서 이미지 워크플로 리소스의 최신 버전을 생성한 날짜와 시간입니다.
- ARN-이미지 워크플로 리소스 최신 버전의 Amazon 리소스 이름(ARN)입니다.

이미지 워크플로 나열

Image Builder 콘솔에서 이미지 워크플로 리소스를 나열하려면 다음 단계를 수행하세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지 워크플로를 선택합니다.

필터링 결과

이미지 워크플로 목록 페이지에서 특정 이미지 워크플로를 검색하여 결과를 필터링할 수 있습니다. 이미지 워크플로에서 사용할 수 있는 필터는 다음과 같습니다.

`Workflow`

결과를 간소화하기 위해 워크플로 이름 전체 또는 일부를 입력할 수 있습니다. 기본값은 목록의 모든 워크플로를 표시하는 것입니다.

`Version`

결과를 간소화하기 위해 버전 번호 전체 또는 일부를 입력할 수 있습니다. 기본값은 목록의 모든 버전을 표시하는 것입니다.

`Type`

워크플로 유형별로 필터링하거나 모든 유형을 볼 수 있습니다. 기본값은 목록의 모든 워크플로 유형을 표시하는 것입니다.

- BUILD
- 테스트

`Owner`

검색 창에서 소유자 필터를 선택하면 Image Builder에서 사용자 계정의 이미지 워크플로 소유자 목록을 표시합니다. 목록에서 소유자를 선택하여 결과를 간소화할 수 있습니다. 기본값은 목록의 모든 소유자를 표시하는 것입니다.

- AWS 계정-워크플로 리소스를 소유한 계정입니다.
- Amazon-Amazon이 소유하고 관리하는 워크플로 리소스입니다.

AWS CLI

에서 list-workflows 명령을 실행하면 소유하거나 액세스할 AWS CLI수 있는 이미지 워크플로 목록을 가져올 수 있습니다.

다음 명령 예제는 필터 없이 list-workflows 명령을 사용하여 소유하고 있거나 액세스 권한이 있는 Image Builder 이미지 워크플로 리소스를 모두 나열하는 방법을 보여줍니다.

예: 모든 이미지 워크플로 나열

```
aws imagebuilder list-workflows
```

출력:

```
{
 "workflowVersionList": [
  {
    "name": "example-test-workflow",
    "dateCreated": "2023-11-21T22:53:14.347Z",
    "version": "1.0.0",
    "owner": "111122223333",
    "type": "TEST",
    "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/test/example-test-
workflow/1.0.0"
  },
  {
    "name": "example-build-workflow",
    "dateCreated": "2023-11-20T12:26:10.425Z",
    "version": "1.0.0",
    "owner": "111122223333",
    "type": "BUILD",
    "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/example-
build-workflow/1.0.0"
  }
 ]
}
```

list-workflows 명령을 실행할 때 다음 예제와 같이 필터를 적용하여 결과를 간소화할 수 있습니다. 결과를 필터링하는 방법에 대한 자세한 내용은AWS CLI 명령 참조의 list-workflows 명령을 참조하세요.

예: 빌드 워크플로용 필터

```
aws imagebuilder list-workflows --filters name="type",values="BUILD"
```

출력:

```
{
 "workflowVersionList": [
  {
    "name": "example-build-workflow",
    "dateCreated": "2023-11-20T12:26:10.425Z",
    "version": "1.0.0",
    "owner": "111122223333",
    "type": "BUILD",
    "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/example-
build-workflow/1.0.0"
  }
 ]
}
```

이미지 워크플로 생성

이미지 워크플로를 생성할 때 이미지 생성 프로세스를 더 많이 제어할 수 있습니다. Image Builder에서 이미지를 빌드할 때 실행할 워크플로와 이미지를 테스트할 때 실행할 워크플로를 지정할 수 있습니다. 워크플로 리소스를 암호화하기 위해 고객 관리형 키를 지정할 수도 있습니다. 워크플로 리소스 암호화에 대한 자세한 내용은 Image Builder의 암호화 및 키 관리 섹션을 참조하세요.

이미지 생성의 경우 빌드 단계 워크플로 하나와 테스트 단계 워크플로를 하나 이상 지정할 수 있습니다. 필요에 따라 빌드 또는 테스트 단계를 완전히 건너뛸 수도 있습니다. 워크플로에서 사용하는 YAML 정의 문서에서 워크플로가 수행하는 작업을 구성합니다. YAML 문서의 구문에 대한 자세한 내용은 YAML 워크플로 문서 생성 섹션을 참조하세요.

새 빌드 또는 테스트 워크플로 생성 단계를 보려면 사용할 환경과 일치하는 탭을 선택합니다.

AWS Management Console

다음 프로세스를 사용하여 Image Builder 콘솔에서 새 워크플로를 만들 수 있습니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 이미지 워크플로를 선택합니다. 그러면 계정이 소유하거나 액세스 권한이 있는 이미지 워크플로의 목록이 표시됩니다.

> **Note**
>
> Image Builder가 기본 워크플로에 사용하는 Amazon 관리형 워크플로 리소스가 항상 목록에 표시됩니다. 이러한 워크플로의 세부 정보를 보려면 워크플로 링크를 선택하면 됩니다.

3. 이미지 워크플로 생성을 선택하여 새 워크플로를 생성합니다. 그러면 이미지 워크플로 생성 페이지가 표시됩니다.
4. 새 워크플로의 세부 정보를 구성합니다. 빌드 워크플로를 생성하려면 양식 상단에 있는 빌드 옵션을 선택합니다. 테스트 워크플로를 생성하려면 양식 상단에 있는 테스트 옵션을 선택합니다. Image Builder는 이 옵션에 따라 템플릿 목록을 채웁니다. 빌드 및 테스트 워크플로의 다른 모든 단계는 동일합니다.

   일반

   일반 섹션에는 워크플로 리소스에 적용되는 설정(예: 이름, 설명)이 포함됩니다. 일반 설정에는 다음이 포함됩니다.

   - 이미지 워크플로 이름(필수)-이미지 워크플로의 이름입니다. 이름은 계정에서 고유해야 합니다. 각 이름의 최대 길이는 128자입니다. 유효한 문자에는 문자, 숫자, 공백, -, _ 등이 포함됩니다.
   - 버전(필수)-생성할 워크플로 리소스의 의미 체계 버전(major.minor.patch)입니다.
   - 설명(선택 사항)-필요에 따라 워크플로에 대한 설명을 추가할 수 있습니다.
   - KMS 키(선택 사항)-고객 관리형 키를 사용하여 워크플로 리소스를 암호화할 수 있습니다. 자세한 내용은 고객 관리형 키로 이미지 워크플로 암호화 단원을 참조하십시오.

   정의 문서

   YAML 워크플로 문서에는 워크플로의 모든 구성이 포함되어 있습니다.

   시작

   - Image Builder 기본 템플릿을 워크플로의 기준으로 사용하려면 템플릿에서 시작 옵션을 선택합니다. 이 옵션은 기본적으로 설정되어 있습니다. 템플릿 목록에서 사용할 템플릿을 선택하면 선택한 템플릿의 기본 구성이 새 워크플로 문서의 콘텐츠에 복사되어 변경 작업을 수행할 수 있습니다.

- 워크플로 문서를 처음부터 정의하려면 처음부터 시작 옵션을 선택합니다. 그러면 문서 형식의 몇 가지 중요한 부분에 대한 간략한 개요가 콘텐츠에 채워져 시작하는 데 도움이 됩니다.

콘텐츠 패널 하단에는 YAML 문서에 대한 경고 또는 오류를 표시하는 상태 표시줄이 있습니다. YAML 워크플로 문서 생성 방법에 대한 자세한 내용은 YAML 워크플로 문서 생성 섹션을 참조하세요.

5. 워크플로를 완료했거나 진행 상황을 저장하고 나중에 다시 돌아오려면 워크플로 생성을 선택합니다.

AWS CLI

에서 create-workflow 명령을 실행 AWS CLI하기 전에 워크플로에 대한 모든 구성이 포함된 YAML 문서를 생성해야 합니다. 자세한 내용은 YAML 워크플로 문서 생성 단원을 참조하십시오.

다음 예에서는 create-workflow AWS CLI 명령을 사용하여 빌드 워크플로를 생성하는 방법을 보여줍니다. `--data` 파라미터는 생성한 워크플로의 빌드 구성이 포함된 YAML 문서를 참조합니다.

예: 워크플로 생성

```
aws imagebuilder create-workflow --name example-build-workflow --semantic-
version 1.0.0 --type BUILD --data file://example-build-workflow.yml
```

출력:

```
{
"workflowBuildVersionArn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
build/example-build-workflow/1.0.0/1",
"clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

다음 예에서는 create-workflow AWS CLI 명령을 사용하여 테스트 워크플로를 생성하는 방법을 보여줍니다. `--data` 파라미터는 생성한 워크플로의 빌드 구성이 포함된 YAML 문서를 참조합니다.

예: 테스트 워크플로 생성

```
aws imagebuilder create-workflow --name example-test-workflow --semantic-
version 1.0.0 --type TEST --data file://example-test-workflow.yml
```

출력:

```
{
"workflowBuildVersionArn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
test/example-test-workflow/1.0.0/1",
"clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

YAML 워크플로 문서 생성

YAML 형식 정의 문서는 이미지 빌드 프로세스의 빌드 및 테스트 단계에 대한 입력, 출력 및 워크플로 단계를 구성합니다. 표준화된 단계가 포함된 템플릿에서 시작하거나 처음부터 시작하여 고유한 워크플로를 정의할 수 있습니다. 템플릿을 사용하든 처음부터 시작하든 관계없이 필요에 따라 워크플로를 사용자 정의할 수 있습니다.

YAML 워크플로 문서의 구조

Image Builder가 이미지 빌드 및 테스트 작업에 사용하는 YAML 워크플로 문서는 다음과 같이 구성되어 있습니다.

- 워크플로 문서 식별
- 워크플로 문서 입력 파라미터
- 워크플로 문서 단계
- 워크플로 문서 출력

워크플로 문서 식별

워크플로를 고유하게 식별합니다. 이 섹션에는 다음과 같은 속성이 포함될 수 있습니다.

필드	설명	형식	필수
name	워크플로 문서의 이름입니다.	String	No
설명	문서에 대한 설명입니다.	String	No

필드	설명	형식	필수
schemaVersion	문서의 스키마 버전, 현재 1.0입니다.	String	예

예제

```
---
name: sample-test-image
description: Workflow for a sample image, with extra configuration options exposed
 through workflow parameters.
schemaVersion: 1.0
```

워크플로 문서 입력 파라미터

워크플로 문서의 이 부분에서는 호출자가 지정할 수 있는 입력 파라미터를 정의합니다. 파라미터가 없으면 이 섹션을 생략해도 됩니다. 파라미터를 지정하는 경우 각 파라미터에 다음 속성이 포함될 수 있습니다.

필드	설명	형식	필수	제약 조건
name	파라미터의 이름입니다.	String	예	
설명	파라미터 설명입니다.	String	No	
기본값	값이 제공되지 않은 경우 파라미터의 값이 기본값으로 적용됩니다. 파라미터 정의에 기본값을 포함하지 않는 경우 런	파라미터 데이터 유형과 일치합니다.	아니요	

필드	설명	형식	필수	제약 조건
	타임에 파라미터 값이 필요합니다.			
유형	파라미터의 데이터 유형입니다. 파라미터 정의에 데이터 유형을 포함하지 않는 경우 파라미터 유형의 기본값은 런타임에 필요한 문자열 값입니다.	String	예	파라미터의 데이터 유형이며 다음 중 하나여야 합니다. • `string` • `integer` • `boolean` • `stringList`

예제

워크플로 문서에 파라미터를 지정합니다.

```
parameters:
  - name: waitForActionAtEnd
    type: boolean
    default: true
    description: "Wait for an external action at the end of the workflow"
```

워크플로 문서의 파라미터 값을 사용합니다.

```
$.parameters.waitForActionAtEnd
```

워크플로 문서 단계

워크플로에 대해 최대 15 단계의 작업을 지정합니다. 단계는 워크플로 문서에 정의된 순서대로 실행됩니다. 실패할 경우 롤백은 실패한 단계부터 시작하여 이전 단계를 거꾸로 진행하면서 역순으로 실행됩니다.

각 단계는 이전 단계 작업의 출력을 참조할 수 있습니다. 이를 체이닝 또는 참조라고합니다. JSONPath 선택기를 사용하여 이전 단계 작업의 결과를 참조할 수 있습니다. 예시:

```
$.stepOutputs.step-name.output-name
```

자세한 내용은 워크플로 문서에 동적 변수 사용 단원을 참조하십시오.

> **Note**
>
> 단계 자체에는 출력 속성이 없더라도 각 단계 작업의 모든 출력이 해당 단계의 `stepOutput`에 포함됩니다.

각 단계에 다음 속성이 포함될 수 있습니다.

필드	설명	형식	필수	기본값	제약 조건
작업	이 단계에서 수행하는 워크플로 작업입니다.	String	예		Image Builder 워크플로 문서에서 지원되는 단계의 작업이어야 합니다.
`if`의 경우 `if` 연산자를 수정하는 일련의 조건문이 뒤따릅니다.	조건문은 워크플로 단계 본문에 제어 결정 지점의 흐름을 추가합니다.	Dict	아니요		Image Builder는 다음과 같은 조건문을 `if` 연산자의 수정자로 지원합니다. • 분기 조건 및 수정자는 `if`, `and`, `or`, `not`입

필드	설명	형식	필수	기본값	제약 조건
					니다. 분기 조건은 한 줄에 자체적으로 지정됩니다. • 비교 연산자는 `booleanEquals` , `numberEquals` , `numberGreaterThan` , `numberGreaterThanEquals` , `numberLessThan` , `numberLessThanEquals` , `stringEquals` 입니다.
설명	단계에 대한 설명입니다.	String	No		빈 문자열은 허용되지 않습니다. 포함할 경우 길이는 1~1024자여야 합니다.

필드	설명	형식	필수	기본값	제약 조건
입력	해당 단계의 작업을 실행하는 데 필요한 파라미터가 들어 있습니다. 키 값을 정적 값으로 지정하거나 올바른 데이터 유형으로 확인되는 JSONPath 변수를 사용하여 지정할 수 있습니다.	Dict	예		
name	단계의 이름입니다. 이 이름은 워크플로 문서 내에서 고유해야 합니다.	String	예		길이는 3~128자여야 합니다. 영숫자 문자 및 _를 포함할 수 있습니다. 공백은 사용할 수 없습니다.

필드	설명	형식	필수	기본값	제약 조건
onFailure	단계가 실패할 경우 수행할 작업을 다음과 같이 구성합니다. 동작 • Abort-단계가 실패하고 워크플로에도 실패하며 실패한 단계 이후의 나머지 단계는 실행하지 않습니다. 롤백이 활성화된 경우 롤백은 실패한 단계부터 시작하여 롤백을 허용하는 모든 단계가 롤백될 때까지 계속됩니다. • Continue-단계는 실	String	No	Abort	Abort | Continue

필드	설명	형식	필수	기본값	제약 조건
	패하지만 실패한 단계 이후 나머지 단계는 계속 실행합니다. 이 경우에는 롤백이 없습니다.				
rollbackEnabled	실패 발생 시 해당 단계를 롤백할지 여부를 구성합니다. 정적 부울 값 또는 부울 값으로 해석되는 동적 JSONPath 변수를 사용할 수 있습니다.	불린(Boolean)	아니요	`true`	`true` | `false` true 또는 false로 확인되는 | 또는 JSONPath 변수입니다.
timeoutSeconds	재시도가 적용되는 경우 실패 및 재시도까지 단계가 실행되는 최대 시간(초)입니다.	Integer	아니요	해당하는 경우 단계 작업에 정의된 기본값에 따라 달라집니다.	1~86,400초 사이(최대 24시간)

예제

```
steps:
```

```
  - name: LaunchTestInstance
    action: LaunchInstance
    onFailure: Abort
    inputs:
      waitFor: "ssmAgent"

  - name: ApplyTestComponents
    action: ExecuteComponents
    onFailure: Abort
    inputs:
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"

  - name: TerminateTestInstance
    action: TerminateInstance
    onFailure: Continue
    inputs:
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"

  - name: WaitForActionAtEnd
    action: WaitForAction
    if:
      booleanEquals: true
      value: "$.parameters.waitForActionAtEnd"
```

워크플로 문서 출력

워크플로의 출력을 정의합니다. 각 출력은 출력의 이름과 값을 지정하는 키 값 페어입니다. 출력을 사용하여 런타임 시 후속 워크플로에서 사용할 수 있는 데이터를 내보낼 수 있습니다. 이 섹션은 선택 사항입니다.

정의하는 각 출력에는 다음 속성이 포함됩니다.

필드	설명	형식	필수
name	출력의 이름입니다. 이 이름은 파이프라인에서 포함하는 워크플로에서 고유해야 합니다.	String	예
값		String	예

필드	설명	형식	필수
	출력의 값입니다. 문자열 값은 동적 변수(예: 단계 작업의 출력 파일)일 수도 있습니다. 자세한 내용은 워크플로 문서에 동적 변수 사용 단원을 참조하십시오.		

예제

`createProdImage` 단계의 단계 출력을 사용하여 워크플로 문서의 출력 이미지 ID를 생성합니다.

```
outputs:
  - name: 'outputImageId'
    value: '$.stepOutputs.createProdImage.imageId'
```

다음 워크플로의 워크플로 출력을 참조하세요.

```
$.workflowOutputs.outputImageId
```

워크플로 문서에 지원되는 단계 작업

이 섹션에는 Image Builder에서 지원하는 단계 작업에 대한 세부 정보가 포함되어 있습니다.

이 섹션에서 사용되는 용어

AMI

Amazon Machine Image

ARN

Amazon 리소스 이름

지원되는 작업

- BootstrapInstanceForContainer
- CollectImageMetadata
- CollectImageScanFindings
- CreateImage
- ExecuteComponents
- LaunchInstance
- RunCommand
- RunSysPrep
- SanitizeInstance
- TerminateInstance
- WaitForAction

BootstrapInstanceForContainer

이 단계 작업은 서비스 스크립트를 실행하여 컨테이너 워크플로를 실행하기 위한 최소 요구 사항으로 인스턴스를 부트스트랩합니다. Image Builder는 Systems Manager API의 sendCommand를 사용하여 이 스크립트를 실행합니다. 자세한 내용은 AWS Systems Manager Run Command를 참조하세요.

> **Note**
>
> 부트스트랩 스크립트는 Image Builder가 Docker 컨테이너를 성공적으로 빌드하기 위한 사전 조건인 AWS CLI 및 Docker 패키지를 설치합니다. 이 단계 작업을 포함하지 않으면 이미지 빌드가 실패할 수 있습니다.

기본 제한 시간: 60분

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	부트스트랩할 인스턴스의 ID입니다.	String	예		이 워크플로의 인스턴스를 시작한 워

입력 이름	설명	형식	필수	기본값	제약 조건
					크플로 단계의 출력 인스턴스 ID여야 합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
runCommandId	인스턴스에서 부트스트랩 스크립트를 실행한 Systems Manager sendCommand의 ID입니다.	String
status	Systems Manager sendCommand에서 반환된 상태입니다.	String
output	Systems Manager sendCommand에서 반환된 출력입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: ContainerBootstrapStep
  action: BootstrapInstanceForContainer
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

워크플로 문서에 있는 단계 작업 값의 출력을 사용합니다.

```
$.stepOutputs.ContainerBootstrapStep.status
```

CollectImageMetadata

이 단계 작업은 빌드 워크플로에만 유효합니다.

EC2 Image Builder는 시작한 EC2 인스턴스에서 AWS Systems Manager (Systems Manager) Agent를 실행하여 이미지를 빌드 및 테스트합니다. Image Builder는 Systems Manager Inventory를 사용하여 빌드 단계에서 사용된 인스턴스에 대한 추가 정보를 수집합니다. 이 정보에는 운영 체제(OS)이름 및 버전 뿐만 아니라 운영 체제에서 보고한 패키지 및 해당 버전 목록이 포함됩니다.

> **Note**
> 이 단계 작업은 AMI를 생성하는 이미지에만 적용됩니다.

기본 제한 시간: 30분

롤백: Image Builder는 이 단계에서 생성된 모든 Systems Manager 리소스를 롤백합니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	메타데이터 설정을 적용할 빌드 인스턴스입니다.	String	예		이 워크플로의 빌드 인스턴스를 시작한 워크플로 단계의 출력 인스턴스 ID여야 합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
osVersion	빌드 인스턴스에서 수집된 운영 체제의 이름과 버전입니다.	String

출력 이름	설명	형식
associationId	인벤토리 수집에 사용하는 Systems Manager 연결 ID입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: CollectMetadataStep
  action: CollectImageMetadata
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

워크플로 문서의 단계 작업 출력을 사용합니다.

```
$.stepOutputs.CollectMetadataStep.osVersion
```

CollectImageScanFindings

계정에서 Amazon Inspector를 활성화하고 파이프라인에서 이미지 스캔을 활성화한 경우, 이 단계 작업은 테스트 인스턴스에 대해 Amazon Inspector에서 보고한 이미지 스캔 결과를 수집합니다. 빌드 워크플로에는 이 단계 작업을 사용할 수 없습니다.

기본 제한 시간: 120분

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	스캔을 실행한 인스턴스의 ID입니다.	String	예		이 워크플로의 인스턴스를 시작한 워크플로 단계

입력 이름	설명	형식	필수	기본값	제약 조건
					의 출력 인스턴스 ID여야 합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
runCommandId	검색 결과를 수집하기 위해 스크립트를 실행한 Systems Manager sendCommand의 ID입니다.	String
status	Systems Manager sendCommand에서 반환된 상태입니다.	String
output	Systems Manager sendCommand에서 반환된 출력입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: CollectFindingsStep
  action: CollectImageScanFindings
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

워크플로 문서에 있는 단계 작업 값의 출력을 사용합니다.

```
$.stepOutputs.CollectFindingsStep.status
```

CreateImage

이 단계 작업은 Amazon EC2 `CreateImage` API를 이용하여 실행 중인 인스턴스에서 이미지를 생성합니다. 생성 프로세스 중에 단계 작업은 리소스가 올바른 상태에 도달했는지 확인하기 위해 필요한 만큼 대기한 후 작업을 계속합니다.

기본 제한 시간: 720분

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	새 이미지를 만들 때 사용할 인스턴스입니다.	String	예		제공된 인스턴스 ID의 인스턴스는 이 단계가 시작될 때 `running` 상태여야 합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
imageId	생성한 이미지의 AMI ID입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: CreateImageFromInstance
  action: CreateImage
  onFailure: Abort
  inputs:
```

```
      instanceId.$: "i-1234567890abcdef0"
```

워크플로 문서에 있는 단계 작업 값의 출력을 사용합니다.

```
$.stepOutputs.CreateImageFromInstance.imageId
```

ExecuteComponents

이 단계 동작은 빌드 중인 현재 이미지의 레시피에 지정된 구성 요소를 실행합니다. 빌드 워크플로는 빌드 인스턴스에서 빌드 구성 요소를 실행합니다. 테스트 워크플로는 테스트 인스턴스에서만 테스트 구성 요소를 실행합니다.

Image Builder는 Systems Manager API의 sendCommand를 사용하여 구성 요소를 실행합니다. 자세한 내용은 AWS Systems Manager Run Command를 참조하세요.

기본 제한 시간: 720분

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	구성 요소를 실행해야 하는 인스턴스의 ID입니다.	String	예		이 워크플로의 인스턴스를 시작한 워크플로 단계의 출력 인스턴스 ID여야 합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
runCommandId	인스턴스에서 구성 요소를 실행한 Systems Manager sendCommand의 ID입니다.	String

출력 이름	설명	형식
status	Systems Manager sendCommand에서 반환된 상태입니다.	String
output	Systems Manager sendCommand에서 반환된 출력입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: ExecComponentsStep
  action: ExecuteComponents
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

워크플로 문서의 단계 작업 출력을 사용합니다.

```
$.stepOutputs.ExecComponentsStep.status
```

LaunchInstance

이 단계 작업은에서 인스턴스를 시작하고 다음 단계로 넘어가기 전에 Systems Manager 에이전트가 인스턴스에서 실행될 때까지 AWS 계정 기다립니다. 시작 작업에는 레시피의 설정과 이미지와 관련된 인프라 구성 리소스가 사용됩니다. 예를 들어, 시작할 인스턴스 유형은 인프라 구성에서 가져옵니다. 출력은 시작한 인스턴스의 인스턴스 ID입니다.

`waitFor` 입력은 단계 완료 요구 사항을 충족하는 조건을 구성합니다.

기본 제한 시간: 60분

롤백: 빌드 인스턴스의 경우 롤백은 인프라 구성 리소스에서 구성한 작업을 수행합니다. 기본적으로 이미지 생성에 실패하면 빌드 인스턴스가 종료됩니다. 하지만 인프라 구성에는 문제 해결을 위해 빌드 인스턴스를 유지하는 설정이 있습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
imageIdOverride	인스턴스를 시작하는 데 사용할 이미지	String	No	빌드 단계: 이미지 레시피 기본 이미지 테스트 단계: 빌드 단계에서 AMI 출력	유효한 AMI ID여야 합니다.
instanceTypesOverride	Image Builder는 성공적으로 시작되는 인스턴스 유형을 찾을 때까지 목록의 각 인스턴스 유형을 시도합니다.	문자열 목록	아니요	인프라 구성에 지정된 인스턴스 유형	유효한 인스턴스 유형이어야 합니다.
waitFor	워크플로 단계를 완료하고 다음 단계로 넘어가기 전에 기다릴 조건입니다.	String	예		Image Builder는 `ssmAgent`(을)를 지원합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
instanceId	시작한 인스턴스의 인스턴스 ID입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: LaunchStep
  action: LaunchInstance
  onFailure: Abort
  inputs:
    waitFor: ssmAgent
```

워크플로 문서의 단계 작업 출력을 사용합니다.

```
$.stepOutputs.LaunchStep.instanceId
```

RunCommand

이 단계 동작은 워크플로의 명령 문서를 실행합니다. Image Builder는 Systems Manager API의 sendCommand를 사용하여 대신 실행합니다. 자세한 내용은 AWS Systems Manager Run Command를 참조하세요.

기본 제한 시간: 12시간

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	명령 문서를 실행할 인스턴스의 ID입니다.	String	예		이 워크플로의 인스턴스를 시작한 워크플로 단계의 출력 인스턴스 ID여야 합니다.
documentName	실행할 Systems Manager 명령 문서의 이름입니다.	String	예		

입력 이름	설명	형식	필수	기본값	제약 조건
parameters	명령 문서에 필요한 파라미터의 키 값 페어의 목록입니다.	dictionary<string, list<string>>	조건		
documentVersion	실행할 명령 문서 버전입니다.	String	No	$DEFAULT	

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
runCommandId	인스턴스에서 명령 문서를 실행한 Systems Manager sendCommand의 ID입니다.	String
status	Systems Manager sendCommand에서 반환된 상태입니다.	String
output	Systems Manager sendCommand에서 반환된 출력입니다.	문자열 목록

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: RunCommandDoc
  action: RunCommand
  onFailure: Abort
  inputs:
    documentName: SampleDocument
    parameters:
```

```
        osPlatform:
          - "linux"
    instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

워크플로 문서에 있는 단계 작업 값의 출력을 사용합니다.

```
$.stepOutputs.RunCommandDoc.status
```

RunSysPrep

이 단계 작업은 스냅샷을 위해 빌드 인스턴스가 종료되기 전에 Systems Manager API의 sendCommand를 사용하여 Windows 인스턴스용 AWSEC2-RunSysprep 문서를 실행합니다. 이러한 작업은 AWS 이미지 강화 및 정리 모범 사례를 따릅니다.

기본 제한 시간: 60분

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	AWSEC2-RunSysprep 문서를 실행할 인스턴스의 ID입니다.	String	예		이 워크플로의 인스턴스를 시작한 워크플로 단계의 출력 인스턴스 ID여야 합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
runCommandId	인스턴스에서 AWSEC2-RunSysprep 문서를 실행한 Systems Manager sendCommand의 ID입니다.	String

출력 이름	설명	형식
status	Systems Manager sendCommand에서 반환된 상태입니다.	String
output	Systems Manager sendCommand에서 반환된 출력입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: RunSysprep
  action: RunSysPrep
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

워크플로 문서에 있는 단계 작업 값의 출력을 사용합니다.

```
$.stepOutputs.RunSysprep.status
```

SanitizeInstance

이 단계 작업은 스냅샷을 위해 빌드 인스턴스를 종료하기 전에 Linux 인스턴스용 권장 삭제 스크립트를 실행합니다. 삭제 스크립트는 최종 이미지가 보안 모범 사례를 따르는지 확인하고 스냅샷으로 전달해서는 안 되는 빌드 아티팩트 또는 설정을 제거하는 데 도움이 됩니다. 스크립트에 대한 자세한 내용은 빌드 후 정리 필요 섹션을 참조하세요. 이 단계 작업은 컨테이너 이미지에 적용되지 않습니다.

Image Builder는 Systems Manager API의 sendCommand를 사용하여 이 스크립트를 실행합니다. 자세한 내용은 AWS Systems Manager Run Command를 참조하세요.

기본 제한 시간: 60분

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	삭제할 인스턴스의 ID입니다.	String	예		이 워크플로의 인스턴스를 시작한 워크플로 단계의 출력 인스턴스 ID여야 합니다.

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
runCommandId	인스턴스에서 삭제 스크립트를 실행한 Systems Manager sendCommand의 ID입니다.	String
status	Systems Manager sendCommand에서 반환된 상태입니다.	String
output	Systems Manager sendCommand에서 반환된 출력입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: SanitizeStep
  action: SanitizeInstance
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

워크플로 문서에 있는 단계 작업 값의 출력을 사용합니다.

```
$.stepOutputs.SanitizeStep.status
```

TerminateInstance

이 단계 작업은 입력으로 전달된 인스턴스 ID를 사용하여 인스턴스를 종료합니다.

기본 제한 시간: 30분

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
instanceId	종료할 인스턴스의 ID입니다.	String	예		

출력: 이 단계 작업에 대한 출력이 없습니다.

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: TerminateInstance
  action: TerminateInstance
  onFailure: Continue
  inputs:
      instanceId.$: i-1234567890abcdef0
```

WaitForAction

이 단계 작업은 실행 중인 워크플로를 일시 중지하고 Image Builder SendWorkflowStepAction API 작업으로부터 외부 작업을 수신할 때까지 기다립니다. 이 단계에서는 EventBridge 이벤트를 세부 정보 유형 `EC2 Image Builder Workflow Step Waiting`과 함께 기본 EventBridge 이벤트 버스에 게시합니다. SNS 주제 ARN을 제공하는 경우 이 단계에서 SNS 알림을 보낼 수도 있습니다.

기본 제한 시간: 3일

롤백: 이 단계 작업에는 롤백이 없습니다.

입력: 다음 표에는 이 단계 작업에 지원되는 입력이 포함되어 있습니다.

입력 이름	설명	형식	필수	기본값	제약 조건
snsTopicArn	워크플로 단계가 보류 중일 때 알림을 보내는 선택적 SNS 주제 ARN입니다.	String	No		

출력: 다음 표에는 이 단계 작업의 출력이 포함되어 있습니다.

출력 이름	설명	형식
작업	SendWorkflowStepAction API 작업이 반환하는 작업입니다.	문자열(RESUME 또는 STOP)
reason	작업이 반환된 이유입니다.	String

예제

워크플로 문서에 단계 작업을 지정합니다.

```
- name: SendEventAndWait
  action: WaitForAction
  onFailure: Abort
  inputs:
    snsTopicArn: arn:aws:sns:us-west-2:111122223333:ExampleTopic
```

워크플로 문서에 있는 단계 작업 값의 출력을 사용합니다.

```
$.stepOutputs.SendEventAndWait.reason
```

워크플로 문서에 동적 변수 사용

워크플로 문서에서 동적 변수를 사용하여 이미지 생성 프로세스의 런타임에 변하는 값을 표현할 수 있습니다. 동적 변수 값은 대상 변수를 고유하게 식별하는 구조적 노드가 있는 JSONPath 선택기로 표시됩니다.

JSONPath 동적 워크플로 변수 구조

```
$.<document structure>.[<step name>.]<variable name>
```

루트($) 뒤의 첫 번째 노드는 워크플로 문서의 구조(예: `stepOutputs` 또는 Image Builder 시스템 변수의 경우 `imageBuilder`)를 나타냅니다. 다음 목록에는 지원되는 JSONPath 워크플로 문서 구조 노드가 포함되어 있습니다.

문서 구조 노드

- 파라미터-워크플로 파라미터
- stepOutputs-동일한 워크플로 문서에 있는 단계의 출력
- workflowOutputs-이미 실행된 워크플로 문서의 출력
- imagebuilder - Image Builder 시스템 변수

`parameters` 및 `stepOutputs` 문서 구조 노드에는 단계의 이름에 대한 선택적 노드가 포함됩니다. 이렇게 하면 모든 단계에서 고유한 변수 이름을 지정할 수 있습니다.

JSONPath의 마지막 노드는 대상 변수의 이름(예: `instanceId`)입니다.

각 단계에서 이러한 JSONPath 동적 변수를 사용하여 이전 단계 작업의 출력을 참조할 수 있습니다. 이를 체이닝 또는 참조라고도 합니다. 이전 단계 작업의 결과를 참조하려면 다음 동적 변수를 사용할 수 있습니다.

```
$.stepOutputs.step-name.output-name
```

입력 파라미터가 동적 변수를 참조하는 경우 다음 예제와 같이 연결 표시기(`.$`)를 파라미터 이름 끝에 연결해야 합니다.

예제

```
  - name: ApplyTestComponents
    action: ExecuteComponents
```

```
    onFailure: Abort
    inputs:
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"
```

Image Builder 시스템 변수 사용

Image Builder는 워크플로 문서에서 사용할 수 있는 다음과 같은 시스템 변수를 제공합니다.

변수 이름	설명	형식	예시 값
cloudWatchLogGroup	출력 로그에서 CloudWatch Logs 그룹의 이름입니다. 형식: `/aws/imagebuilder/`*<recipe-name>*	String	`/aws/imagebuilder/`*sampleImageRecipe*
cloudWatchLogStream	출력 로그에서 CloudWatch Logs 스트림의 이름입니다.	String	*1.0.0/1*
collectImageMetadata	인스턴스 메타데이터를 수집할지 여부를 Image Builder에 지시하는 설정입니다.	불	`true` | `false`
collectImageScanFindings	Image Builder가 이미지 스캔 결과를 수집할 수 있도록 지원하는 설정의 현재 값입니다.	불	`true` | `false`
imageBuildNumber	이미지의 빌드 버전 번호입니다.	Integer	*1*

변수 이름	설명	형식	예시 값
imageId	기본 이미지의 AMI id입니다.	String	*ami-1234567890abcdef1*
imageName	이미지의 이름입니다.	String	*sampleImage*
imageType	이미지 출력 유형입니다.	String	`AMI | Docker`
imageVersionNumber	이미지의 버전 번호입니다.	String	*1.0.0*
instanceProfileName	Image Builder가 빌드 및 테스트 인스턴스를 시작하는 데 사용하는 인스턴스 프로파일 역할의 이름입니다.	String	*SampleImageBuilderInstanceProfileRole*
platform	빌드된 이미지의 운영 체제 플랫폼입니다.	String	`Linux | Windows | MacOS`
s3Logs	Image Builder가 작성하는 S3 로그의 구성을 포함하는 JSON 객체입니다.	JSON 객체	{'s3Logs': {'s3BucketName': '*sample-bucket* ', 's3KeyPrefix': '*ib-logs*'}}
securityGroups	빌드 및 테스트 인스턴스에 적용되는 보안 그룹 ID입니다.	목록[문자열]	*[sg-1234567890abcdef1, sg-11112222333344445]*

변수 이름	설명	형식	예시 값
sourceImageARN	워크플로가 빌드 및 테스트 단계에 사용하는 Image Builder 이미지 리소스의 Amazon 리소스 이름(ARN)입니다.	String	arn:aws:imagebuilder:*us-east-1*:*111122223333*:image/*sampleImage*/*1.0.0/1*
subnetId	빌드 및 테스트 인스턴스를 시작할 서부넷의 ID입니다.	String	*subnet-1234567890abcdef1*
terminateInstanceOnFailure	실패 시 Image Builder에서 인스턴스를 종료하거나 문제 해결을 위해 인스턴스를 유지하도록 지시하는 설정의 현재 값입니다.	불	`true` | `false`
workflowPhase	워크플로 실행을 위해 실행 중인 현재 단계입니다.	String	`Build` | `Test`
workingDirectory	작업 디렉터리에 대한 경로입니다.	String	`/tmp`

워크플로 단계에서 조건문을 사용합니다.

조건문은 `if`문 문서 속성으로 시작합니다. `if`문의 궁극적인 목적은 단계 작업을 실행할지 아니면 건너뛰는지 결정하는 것입니다. `if`문이 `true`로 확인되면 단계 작업이 실행됩니다. `false`로 확인되면 Image Builder는 단계 작업을 건너뛰고 로그에 `SKIPPED` 단계 상태를 기록합니다.

if문은 분기 명령문(and, or)과 조건부 수정자(not)를 지원합니다. 또한 비교하는 데이터 유형(문자열 또는 숫자)에 따라 값을 비교(같음, 보다 작음, 보다 큼)하는 다음 조건문도 지원합니다.

지원되는 비교 연산자

- booleanEquals
- numberEquals
- numberGreaterThan
- numberGreaterThanEquals
- numberLessThan
- numberLessThanEquals
- stringEquals

분기 명령문 및 조건부 수정자 규칙

분기 명령문(and, or) 및 조건부 수정자(not)에는 다음 규칙이 적용됩니다.

- 분기문과 조건부 수정자는 한 줄에 단독으로 나타나야 합니다.
- 분기문과 조건부 수정자는 레벨 규칙을 따라야 합니다.
  - 상위 수준에는 문이 하나만 있을 수 있습니다.
  - 각 하위 분기 또는 수정자는 새 레벨을 시작합니다.

  레벨에 대한 자세한 내용은 조건문의 중첩 수준 섹션을 참조하세요.
- 각 분기문에는 하위 조건문이 1개 이상 10개 이하로 있어야 합니다.
- 조건부 수정자는 하위 조건문 하나에만 사용할 수 있습니다.

조건문의 중첩 수준

조건문은 자체 섹션의 여러 수준에서 작동합니다. 예를 들어 if문 속성은 워크플로 문서에서 단계 이름 및 작업과 동일한 수준에 나타납니다. 이것이 조건문의 기본입니다.

조건문 수준은 최대 4개 지정할 수 있지만 상위 수준에는 문 하나만 나타날 수 있습니다. 다른 모든 분기문, 조건부 수정자 또는 조건 연산자는 여기에서 수준당 하나씩 들여쓰기됩니다.

다음 개요는 조건문의 최대 중첩 레벨 수를 보여줍니다.

```
base:
```

```
  parent:
    - child (level 2)
        - child (level 3)
            child (level 4)
```

if 속성

if 속성은 조건문을 문서 속성으로 지정합니다. 이 레벨은 0입니다.

상위 수준

조건문의 첫 번째 중첩 레벨입니다. 이 수준에는 문이 하나만 있을 수 있습니다. 분기 또는 수정자가 필요하지 않은 경우 하위 명령문이 없는 조건부 연산자를 사용할 수 있습니다. 이 수준에서는 조건 연산자를 제외하고 대시 표기법을 사용하지 않습니다.

하위 수준

레벨 2~4는 하위 수준으로 간주합니다. 하위 명령문에는 분기문, 조건부 수정자 또는 조건 연산자가 포함될 수 있습니다.

예: 중첩 수준

다음 예제에서는 조건문 내 최대 레벨 수를 보여줍니다.

```
if:
  and:                              #first level
    - stringEquals: 'my_string'     #second level
      value: 'my_string'
    - and:                          #also second level
        - numberEquals: '1'         #third level
          value: 1
        - not:                      #also third level
            stringEquals: 'second_string'      #fourth level
            value: "diff_string"
```

중첩 규칙

- 하위 수준의 각 분기 또는 수정자는 새 레벨을 시작합니다.
- 각 레벨이 들여쓰기됩니다.
- 명령문, 수정자 또는 상위 수준의 연산자 하나를 포함하여 최대 4개의 수준이 있을 수 있으며, 추가 수준은 최대 3개까지 가능합니다.

조건문 예제

이 예제 그룹은 조건문의 다양한 측면을 보여줍니다.

브랜치: and

and 분기문은 분기의 하위 표현식 목록에서 연산을 수행하며, 이들 표현식은 모두 true로 평가되어야 합니다. Image Builder는 목록에 나타나는 순서대로 표현식을 평가합니다. false로 평가되는 표현식이 있으면 처리가 중지되고 분기가 false로 고려됩니다.

다음 예제에서는 두 표현식 모두 true로 평가되기 때문에 true로 평가합니다.

```
if:
  and:
    - stringEquals: 'test_string'
      value: 'test_string'
    - numberEquals: 1
      value: 1
```

브랜치: or

or 분기문은 분기의 하위 표현식 목록에서 연산을 수행하며, 이들 표현식 중 1개 이상은 true로 평가되어야 합니다. Image Builder는 목록에 나타나는 순서대로 표현식을 평가합니다. true로 평가되는 표현식이 있으면 처리가 중지되고 분기가 true로 고려됩니다.

다음 예제에서는 첫 번째 표현식이 true임에도 false로 평가합니다.

```
if:
  or:
    - stringEquals: 'test_string'
      value: 'test_string_not_equal'
    - numberEquals: 1
      value: 1
```

조건부 수정자: not

not 조건부 수정자는 분기의 하위 항목인 조건문을 무효화합니다.

다음 예제에서는 not 수정자가 stringEquals 조건문을 부정하는 경우 true로 평가합니다.

```
if:
```

```
  not:
    - stringEquals: 'test_string'
      value: 'test_string_not_equal'
```

조건문: booleanEquals

booleanEquals 조건문 연산자는 부울 값을 비교하고 부울 값이 정확히 일치하면 true를 반환합니다.

다음 예제에서는 collectImageScanFindings의 활성화 여부를 확인합니다.

```
if:
  - booleanEquals: true
    value: '$.imagebuilder.collectImageScanFindings'
```

조건문: stringEquals

stringEquals 조건문 연산자는 두 문자열을 비교하고 문자열이 정확히 일치하면 true를 반환합니다. 두 값 중 하나가 문자열이 아닌 경우 Image Builder는 비교하기 전에 해당 값을 문자열로 변환합니다.

다음 예제에서는 플랫폼 시스템 변수를 비교하여 워크플로가 Linux 플랫폼에서 실행되고 있는지 확인합니다.

```
if:
  - stringEquals: 'Linux'
    value: '$.imagebuilder.Platform'
```

조건문: numberEquals

numberEquals 비교 연산자는 두 숫자를 비교하고 숫자가 같으면 true를 반환합니다. 비교할 숫자는 다음 형식 중 하나여야 합니다.

- Integer
- Float
- 다음 정규식 패턴과 일치하는 문자열: ^-?[0-9]+(\.)?[0-9]+$

다음 예제 비교는 모두 true로 평가됩니다.

```
if:
  # Value provider as a number
  numberEquals: 1
```

```
  value: '1'

  # Comparison value provided as a string
  numberEquals: '1'
  value: 1

  # Value provided as a string
  numberEquals: 1
  value: '1'

  # Floats are supported
  numberEquals: 5.0
  value: 5.0

  # Negative values are supported
  numberEquals: -1
  value: -1
```

반복 가능한 파이프라인 프로세스를 통해 Image Builder에서 사용자 지정 이미지 생성 관리

Image Builder 이미지 파이프라인은 사용자 지정 AMI 및 컨테이너 이미지를 생성하고 유지 관리하기 위한 자동화 프레임워크를 제공합니다. 파이프라인은 다음과 같은 기능을 제공합니다.

- 기본 이미지, 구축 및 테스트용 구성 요소, 인프라 구성, 배포 설정을 조합하는 방법을 설명합니다.
- 콘솔 내 마법사의 `Schedule builder`(을)를 사용하거나 이미지 반복 업데이트에 대한 cron 표현식을 입력하여 자동화된 유지 관리 프로세스를 쉽게 예약할 수 있습니다.
- 기본 이미지 및 구성 요소에 대한 변경 감지를 활성화하여 변경 사항이 없을 경우 예약된 빌드를 자동으로 건너뛸 수 있습니다.
- Amazon EventBridge를 통해 규칙 기반 자동화를 활성화하세요.

> **Note**
>
> EventBridge API를 사용하여 규칙을 보거나 변경하는 방법에 대한 자세한 내용은 Amazon EventBridge API 참조를 참조하세요. 에서 EventBridge events 명령을 사용하여 규칙을 보거나 변경하는 AWS CLI 방법에 대한 자세한 내용은 AWS CLI 명령 참조의 이벤트를 참조하세요.

내용

- 파이프라인 세부 정보 나열 및 보기
- AMI 이미지 파이프라인 생성 및 업데이트
- 컨테이너 이미지 파이프라인 생성 및 업데이트
- Image Builder의 이미지 파이프라인 워크플로 구성
- 이미지 파이프라인 실행
- Image Builder에서 cron 표현식 사용
- Image Builder 파이프라인과 함께 EventBridge 규칙 사용

파이프라인 세부 정보 나열 및 보기

이 섹션에서는 EC2 Image Builder 이미지 파이프라인에 대한 정보를 찾고 세부 정보를 볼 수 있는 다양한 방법을 설명합니다.

파이프라인 세부 정보

- 에서 이미지 파이프라인 나열 AWS CLI
- 에서 이미지 파이프라인 세부 정보 가져오기 AWS CLI

에서 이미지 파이프라인 나열 AWS CLI

다음 예제에서는에서 list-image-pipelines 명령을 사용하여 모든 이미지 파이프라인을 나열 AWS CLI 하는 방법을 보여줍니다.

```
aws imagebuilder list-image-pipelines
```

에서 이미지 파이프라인 세부 정보 가져오기 AWS CLI

다음 예제에서는에서 get-image-pipeline 명령을 사용하여 ARN을 통해 이미지 파이프라인에 대한 세부 정보를 AWS CLI 가져오는 방법을 보여줍니다.

```
aws imagebuilder get-image-pipeline --image-pipeline-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

AMI 이미지 파이프라인 생성 및 업데이트

Image Builder API 또는 AWS CLI의 imagebuilder 명령을 사용하여 Image Builder 콘솔의 AMI 이미지 파이프라인을 설정, 구성, 관리할 수 있습니다. 이미지 파이프라인 생성 콘솔 마법사가 다음 단계를 안내합니다.

- 이름, 설명, 리소스 태그와 같은 파이프라인 세부 정보를 지정합니다.
- 빠른 시작 Amazon 관리형 이미지, 생성했거나 공유한 이미지 또는를 통해 구독한 이미지의 기본 이미지가 포함된 AMI 이미지 레시피를 선택합니다 AWS Marketplace. 레시피에는 Image Builder가 이미지를 빌드하는 데 사용하는 EC2 인스턴스에서 다음 작업을 수행하는 구성 요소도 포함되어 있습니다.

  - 소프트웨어 추가 및 제거
  - 설정 및 스크립트 사용자 지정
  - 선택한 테스트 실행
- 워크플로를 지정하여 파이프라인에서 실행하는 이미지 빌드와 테스트 단계를 구성합니다.
- 기본 설정 또는 직접 구성한 설정으로 파이프라인의 인프라 구성을 정의합니다. 구성에는 이미지, 보안 및 네트워크 설정, 로그 스토리지 및 문제 해결 설정, SNS 알림에 사용할 인스턴스 유형 및 키 쌍이 포함됩니다.

  이 단계는 선택 사항입니다. 구성을 직접 정의하지 않는 경우 Image Builder는 인프라 구성에 기본 설정을 사용합니다.
- 대상 AWS 리전 및 계정에 이미지를 전송하기 위한 배포 설정을 정의합니다. 암호화를 위한 KMS 키를 지정하거나, AMI 공유 또는 라이선스 구성을 구성하거나, 배포하는 AMI에 대한 시작 템플릿을 구성할 수 있습니다.

  이 단계는 선택 사항입니다. 구성을 직접 정의하지 않는 경우 Image Builder는 출력 AMI에 기본 이름 지정을 사용하고 AMI를 소스 리전에 배포합니다. 소스 리전은 파이프라인을 실행하는 리전입니다.

기본값을 사용하는 이미지 파이프라인 생성 콘솔 마법사 사용에 대한 자세한 내용 및 단계별 자습서는 자습서: Image Builder 콘솔 마법사에서 출력 AMI를 사용하여 이미지 파이프라인 생성 섹션을 참조하세요.

내용

- 에서 AMI 이미지 파이프라인 생성 AWS CLI
- 콘솔에서 AMI 이미지 파이프라인 업데이트
- 에서 AMI 이미지 파이프라인 업데이트 AWS CLI

에서 AMI 이미지 파이프라인 생성 AWS CLI

AWS CLI에서 create-image-pipeline 명령에 대한 입력으로 구성 세부 정보가 들어 있는 JSON 파일을 사용하여 AMI 이미지 파이프라인을 생성할 수 있습니다.

파이프라인이 기본 이미지와 구성 요소의 보류 중인 업데이트를 통합하기 위해 새 이미지를 빌드하는 빈도는 구성한 `schedule`에 따라 다릅니다. `schedule`(에)는 다음 속성이 있습니다.

- `scheduleExpression` - 파이프라인 실행 일정을 설정하여 `pipelineExecutionStartCondition`(을)를 평가하고 빌드를 시작할지 여부를 결정합니다. 일

정은 cron 표현식으로 구성됩니다. Image Builder에서 cron 표현식의 형식을 지정하는 방법에 대한 자세한 내용은 Image Builder에서 cron 표현식 사용 섹션을 참조하세요.

- `pipelineExecutionStartCondition` - 파이프라인에서 빌드를 시작할지 여부를 결정합니다. 유효한 값으로는 다음이 포함됩니다.
  - `EXPRESSION_MATCH_ONLY` - 파이프라인은 cron 표현식이 현재 시간과 일치할 때마다 새 이미지를 생성합니다.
  - `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE` - 기본 이미지나 구성 요소에 보류 중인 변경 사항이 없는 한 파이프라인에서 새 이미지 빌드를 시작하지 않습니다.

에서 create-image-pipeline 명령을 실행하면 AWS CLI많은 구성 리소스가 선택 사항입니다. 그러나 일부 리소스에는 파이프라인이 생성하는 이미지 유형에 따라 조건부 요구 사항이 있습니다. AMI 이미지 파이프라인에는 다음과 같은 리소스가 필요합니다.

- 이미지 레시피 ARN
- 인프라 구성 ARN

1. CLI 입력 JSON 파일 생성

   자주 사용하는 파일 편집 도구를 사용하여 다음 키와 환경에 적합한 값을 포함하는 JSON 파일을 만드세요. 이 예제에서는 `create-image-pipeline.json`(이)라는 이름의 파일이 사용됩니다.

```
{
 "name": "MyWindows2019Pipeline",
 "description": "Builds Windows 2019 Images",
 "enhancedImageMetadataEnabled": true,
 "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-recipe/2020.12.03",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-infrastructure-configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-distribution-configuration",
 "imageTestsConfiguration": {
  "imageTestsEnabled": true,
  "timeoutMinutes": 60
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * SUN *)",
```

```
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "ENABLED"
}
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

2. 입력으로 생성한 파일을 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder create-image-pipeline --cli-input-json file://create-image-
pipeline.json
```

콘솔에서 AMI 이미지 파이프라인 업데이트

AMI 이미지용 Image Builder 이미지 파이프라인을 생성한 후, 이미지 빌더 콘솔에서 인프라 구성 및 배포 설정을 변경할 수 있습니다.

새 이미지 레시피로 이미지 파이프라인을 업데이트하려면 AWS CLI(을)를 사용해야 합니다. 자세한 내용은 이 가이드의 에서 AMI 이미지 파이프라인 업데이트 AWS CLI 섹션을 참조하세요.

기존 이미지 빌더 파이프라인 선택

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 계정에서 생성된 이미지 파이프라인 목록을 보려면 탐색 창에서 이미지 파이프라인을 선택합니다.

> **Note**
>
> 이미지 파이프라인 목록에는 파이프라인에서 생성된 출력 이미지 유형(AMI 또는 Docker)에 대한 표시기가 포함됩니다.

3. 세부 정보를 보거나 파이프라인을 편집하려면 파이프라인 이름 링크를 선택합니다. 파이프라인의 세부 정보 보기가 열립니다.

> **Note**
>
> 또한 파이프라인 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

파이프라인 세부 정보

파이프라인 세부 정보 페이지에는 다음 섹션이 포함되어 있습니다.

요약

페이지 상단의 섹션에는 세부 정보 탭이 열려 있는 상태에서 볼 수 있는 파이프라인의 주요 세부 정보가 요약되어 있습니다. 이 섹션에 표시된 세부 정보는 해당 세부 정보 탭에서만 편집할 수 있습니다.

세부 정보 탭

- 출력 이미지 - 파이프라인에서 생성한 출력 이미지를 표시합니다.
- 이미지 레시피 - 레시피 세부 정보를 보여줍니다. 레시피를 생성한 후에는 편집할 수 없습니다. 이미지 빌더 콘솔의 이미지 레시피 페이지에서 또는 AWS CLI의 이미지 빌더 명령을 사용하여 새 버전의 레시피를 만들어야 합니다. 자세한 내용은 Image Builder에서 레시피 관리 단원을 참조하십시오.
- 인프라 구성 - 빌드 파이프라인 인프라 구성을 위한 편집 가능한 정보를 표시합니다.
- 배포 설정 - AMI 배포에 대한 편집 가능한 정보를 표시합니다.
- EventBridge 규칙 - 선택한 이벤트 버스에 대해 현재 파이프라인을 대상으로 하는 EventBridge 규칙을 표시합니다. EventBridge 콘솔에 연결되는 이벤트 버스 생성 및 규칙 생성 작업을 포함합니다. 이 탭에 대한 자세한 내용은 EventBridge 규칙 사용 섹션을 참조하세요.

파이프라인의 인프라 구성 편집

인프라 구성에는 파이프라인을 생성한 후 편집할 수 있는 다음과 같은 세부 정보가 포함됩니다.

- 인프라 구성에 관한 설명.
- 인스턴스 프로파일과 연결할 IAM 역할.
- AWS 인프라 - 인스턴스 유형 및 알림을 위한 SNS 주제 포함.

- VPC, 서브넷 및 보안 그룹.
- 장애 발생 시 인스턴스 종료, 연결을 위한 키 페어, 인스턴스 로그를 위한 선택적 S3 버킷 위치를 포함한 문제 해결 설정.

파이프라인 세부 정보 페이지에서 인프라 구성을 편집하려면 다음 단계를 따르세요.

1. 인프라 구성 탭을 선택합니다.
2. 구성 세부 정보 패널의 오른쪽 상단에서 편집을 선택합니다.
3. 인프라 구성에 적용한 업데이트를 저장할 준비가 되면 변경 내용 저장을 선택합니다.

파이프라인의 배포 설정 편집

배포 설정에는 파이프라인을 생성한 후 편집할 수 있는 다음과 같은 세부 정보가 포함됩니다.

- 이 배포 구성에 대한 설명입니다.
- 이미지를 배포하는 리전의 리전 설정. 리전 1은 파이프라인을 생성한 리전을 기본 리전으로 설정합니다. 리전 추가 버튼을 사용하여 배포할 리전을 추가할 수 있으며 리전 1을 제외한 모든 리전을 제거할 수 있습니다.

  리전 설정에는 다음이 포함됩니다.
  - 대상 리전
  - 출력 AMI 이름
  - 시작 권한 및 이를 공유할 계정
  - 관련 라이선스(연결 라이선스 구성)

> **Note**
>
> License Manager 설정은 계정에서 활성화해야 하는 AWS 리전, 예를 들어 `ap-east-1` (홍콩) 리전과 `me-south-1` (바레인) 리전 간에 복제되지 않습니다.

파이프라인 세부 정보 페이지에서 배포 설정을 편집하려면 다음 단계를 따르세요.

1. 배포 설정 탭을 선택합니다.
2. 배포 세부 정보 패널의 오른쪽 상단에서 편집을 선택합니다.
3. 업데이트를 저장할 준비가 되면 변경 내용 저장을 선택합니다.

파이프라인의 빌드 일정 편집

파이프라인 편집 페이지에는 파이프라인을 생성한 후 편집할 수 있는 다음과 같은 세부 정보가 포함되어 있습니다.

- 파이프라인 설명.
- 향상된 메타데이터 수집. 이 옵션은 기본적으로 활성화되어 있습니다. 이 기능을 끄려면 향상된 메타데이터 수집 활성화 확인란의 선택을 취소하세요.
- 파이프라인의 빌드 일정. 여기에서 스케줄 옵션과 모든 설정을 변경할 수 있습니다.

파이프라인 세부 정보 페이지에서 파이프라인을 편집하려면 다음 단계를 따르세요.

1. 파이프라인 세부 정보 페이지의 오른쪽 상단에서 작업을 선택한 다음 파이프라인 편집을 선택합니다.
2. 업데이트를 저장할 준비가 되면 변경 내용 저장을 선택합니다.

> **Note**
>
> cron 표현식을 사용하여 빌드 일정 설정에 대한 자세한 내용은 Image Builder에서 cron 표현식 사용 섹션을 참조하세요.

에서 AMI 이미지 파이프라인 업데이트 AWS CLI

JSON 파일을 AWS CLI의 update-image-pipeline 명령의 입력으로 사용하여 AMI 이미지 파이프라인을 업데이트할 수 있습니다. JSON 파일을 구성하려면 다음과 같은 기존 리소스를 참조할 수 있는 Amazon 리소스 이름(ARN)이 있어야 합니다.

- 업데이트할 이미지 파이프라인
- 이미지 레시피
- 인프라 구성
- 배포 설정

다음과 AWS CLI 같이에서 update-image-pipeline 명령을 사용하여 AMI 이미지 파이프라인을 업데이트할 수 있습니다.

> **Note**
>
> UpdateImagePipeline은 파이프라인에 대한 선택적 업데이트를 지원하지 않습니다. 업데이트 요청에는 변경된 속성뿐만 아니라 모든 필수 속성을 지정해야 합니다.

1. CLI 입력 JSON 파일 생성

   자주 사용하는 파일 편집 도구를 사용하여 다음 키와 환경에 적합한 값을 포함하는 JSON 파일을 만드세요. 이 예제에서는 `create-component.json`(이)라는 이름의 파일이 사용됩니다.

```
 {
 "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-
pipeline/my-example-pipeline",
 "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-
example-recipe/2019.12.08",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-
configuration",
 "imageTestsConfiguration": {
  "imageTestsEnabled": true,
  "timeoutMinutes": 120
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * MON *)",
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "DISABLED"
}
```

   > **Note**
   >
   > - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.

- JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

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

컨테이너 이미지 파이프라인 생성 및 업데이트

Image Builder 콘솔, Image Builder 컨테이너 또는 AWS CLI의 imagebuilder 명령을 사용하여 AMI 이미지 파이프라인을 설정, 구성 및 관리할 수 있습니다. 이미지 파이프라인 생성 콘솔 마법사는 시작 아티팩트를 제공하고 다음 단계를 안내합니다.

- 퀵 스타트 관리 이미지, Amazon ECR 또는 Docker Hub 리포지토리에서 기본 이미지를 선택
- 소프트웨어 추가 및 제거
- 설정 및 스크립트 사용자 지정
- 선택한 테스트 실행
- 사전 구성된 빌드 타임 변수를 사용하여 Dockerfile을 생성합니다.
- AWS 리전에 이미지 배포

이미지 파이프라인 생성 콘솔 마법사 사용에 대한 자세한 내용 및 단계별 자습서는 자습서: Image Builder 콘솔 마법사에서 출력 Docker 컨테이너 이미지를 사용하여 이미지 파이프라인 생성 섹션을 참조하세요.

내용

- 에서 컨테이너 이미지 파이프라인 생성 AWS CLI
- 콘솔에서 컨테이너 이미지 파이프라인 업데이트
- 에서 컨테이너 이미지 파이프라인 업데이트 AWS CLI

에서 컨테이너 이미지 파이프라인 생성 AWS CLI

JSON 파일을 AWS CLI의 create-image-pipeline 명령에 대한 입력으로 사용하여 컨테이너 이미지 파이프라인을 만들 수 있습니다.

파이프라인이 기본 이미지와 구성 요소의 보류 중인 업데이트를 통합하기 위해 새 이미지를 빌드하는 빈도는 구성한 `schedule`에 따라 다릅니다. `schedule`(에)는 다음 속성이 있습니다.

- `scheduleExpression` - 파이프라인 실행 일정을 설정하여 `pipelineExecutionStartCondition`(을)를 평가하고 빌드를 시작할지 여부를 결정합니다. 일정은 cron 표현식으로 구성됩니다. Image Builder에서 cron 표현식의 형식을 지정하는 방법에 대한 자세한 내용은 Image Builder에서 cron 표현식 사용 섹션을 참조하세요.
- `pipelineExecutionStartCondition` - 파이프라인에서 빌드를 시작할지 여부를 결정합니다. 유효한 값으로는 다음이 포함됩니다.
  - `EXPRESSION_MATCH_ONLY` - 파이프라인은 cron 표현식이 현재 시간과 일치할 때마다 새 이미지를 생성합니다.
  - `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE` - 기본 이미지나 구성 요소에 보류 중인 변경 사항이 없는 한 파이프라인에서 새 이미지 빌드를 시작하지 않습니다.

에서 create-image-pipeline 명령을 실행하면 AWS CLI많은 구성 리소스가 선택 사항입니다. 그러나 일부 리소스에는 파이프라인이 생성하는 이미지 유형에 따라 조건부 요구 사항이 있습니다. 컨테이너 이미지 파이프라인에는 다음과 같은 리소스가 필요합니다.

- 컨테이너 레시피ARN
- 인프라 구성 ARN

명령을 실행할 때 배포 구성 리소스를 포함하지 않으면 create-image-pipeline 명령을 실행하는 리전의 컨테이너 레시피에 대상 리포지토리로 지정한 ECR 리포지토리에 출력 이미지가 저장됩니다. 파이프라인의 배포 구성 리소스를 포함하면 배포의 첫 번째 리전에 대해 지정한 대상 리포지토리가 사용됩니다.

1. CLI 입력 JSON 파일 생성

   자주 사용하는 파일 편집 도구를 사용하여 다음 키와 환경에 적합한 값을 포함하는 JSON 파일을 만드세요. 이 예제에서는 `create-image-pipeline.json`(이)라는 이름의 파일이 사용됩니다.

   ```
   {
   ```

```
 "name": "MyWindows2019Pipeline",
 "description": "Builds Windows 2019 Images",
 "enhancedImageMetadataEnabled": true,
 "containerRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:container-
recipe/my-example-recipe/2020.12.03",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-
configuration",
 "imageTestsConfiguration": {
  "imageTestsEnabled": true,
  "timeoutMinutes": 60
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * SUN *)",
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "ENABLED"
}
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

```
aws imagebuilder create-image-pipeline --cli-input-json file://create-image-
pipeline.json
```

콘솔에서 컨테이너 이미지 파이프라인 업데이트

도커 이미지용 Image Builder 컨테이너 이미지 파이프라인을 생성한 후 Image Builder 콘솔에서 인프라 구성 및 배포 설정을 변경할 수 있습니다.

새 컨테이너 레시피로 컨테이너 이미지 파이프라인을 업데이트하려면 AWS CLI(을)를 사용해야 합니다. 자세한 내용은 이 가이드의 에서 컨테이너 이미지 파이프라인 업데이트 AWS CLI 섹션을 참조하세요.

기존 Image Builder 도커 이미지 이미지 파이프라인 선택

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 계정에서 생성된 이미지 파이프라인 목록을 보려면 탐색 창에서 이미지 파이프라인을 선택합니다.

> **Note**
>
> 이미지 파이프라인 목록에는 파이프라인에서 생성된 출력 이미지 유형(AMI 또는 Docker)에 대한 표시기가 포함됩니다.

3. 세부 정보를 보거나 파이프라인을 편집하려면 파이프라인 이름 링크를 선택합니다. 파이프라인의 세부 정보 보기가 열립니다.

> **Note**
>
> 또한 파이프라인 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

파이프라인 세부 정보

EC2 Image Builder 파이프라인 세부 정보 페이지에는 다음 단원이 포함되어 있습니다.

요약

페이지 상단의 섹션에는 세부 정보 탭이 열려 있는 상태에서 볼 수 있는 파이프라인의 주요 세부 정보가 요약되어 있습니다. 이 섹션에 표시된 세부 정보는 해당 세부 정보 탭에서만 편집할 수 있습니다.

세부 정보 탭

- 출력 이미지 - 파이프라인에서 생성한 출력 이미지를 표시합니다.
- 컨테이너 레시피 – 레시피 세부 정보를 표시합니다. 레시피를 생성한 후에는 편집할 수 없습니다. 컨테이너 레시피 페이지에서 레시피의 새 버전을 생성해야 합니다. 자세한 내용은 컨테이너 레시피의 새 버전 생성 섹션을 참조하세요.

- 인프라 구성 - 빌드 파이프라인 인프라 구성을 위한 편집 가능한 정보를 표시합니다.
- 배포 설정 - 도커 이미지 배포에 대한 편집 가능한 정보를 표시합니다.
- EventBridge 규칙 - 선택한 이벤트 버스에 대해 현재 파이프라인을 대상으로 하는 EventBridge 규칙을 표시합니다. EventBridge 콘솔에 연결되는 이벤트 버스 생성 및 규칙 생성 작업을 포함합니다. 이 탭에 대한 자세한 내용은 EventBridge 규칙 사용 섹션을 참조하세요.

파이프라인의 인프라 구성 편집

인프라 구성에는 파이프라인을 생성한 후 편집할 수 있는 다음과 같은 세부 정보가 포함됩니다.

- 인프라 구성에 관한 설명.
- 인스턴스 프로파일과 연결할 IAM 역할.
- AWS 인프라 - 인스턴스 유형 및 알림을 위한 SNS 주제 포함.
- VPC, 서브넷 및 보안 그룹.
- 장애 발생 시 인스턴스 종료, 연결을 위한 키 페어, 인스턴스 로그를 위한 선택적 S3 버킷 위치를 포함한 문제 해결 설정.

파이프라인 세부 정보 페이지에서 인프라 구성을 편집하려면 다음 단계를 따르세요.

1. 인프라 구성 탭을 선택합니다.
2. 구성 세부 정보 패널의 오른쪽 상단에서 편집을 선택합니다.
3. 인프라 구성에 적용한 업데이트를 저장할 준비가 되면 변경 내용 저장을 선택합니다.

파이프라인의 배포 설정 편집

배포 설정에는 파이프라인을 생성한 후 편집할 수 있는 다음과 같은 세부 정보가 포함됩니다.

- 이 배포 설정에 대한 설명입니다.
- 이미지를 배포하는 리전의 리전 설정. 리전 1은 파이프라인을 생성한 리전을 기본 리전으로 설정합니다. 리전 추가 버튼을 사용하여 배포할 리전을 추가할 수 있으며 리전 1을 제외한 모든 리전을 제거할 수 있습니다.

  리전 설정에는 다음이 포함됩니다.
  - 대상 리전
  - 이 서비스의 기본값은 'ECR'이며 편집할 수 없습니다.

- 리포지토리 이름 — 대상 리포지토리의 이름(Amazon ECR 위치 제외). 예를 들어 리포지토리 이름과 해당 위치는 다음과 같은 패턴으로 표시됩니다.

  `<account-id>.dkr.ecr.<region>.amazonaws.com/<repository-name>`

  > **Note**
  >
  > 리포지토리 이름을 변경하려면 변경 후 생성된 이미지만 새 이름 아래에 추가됩니다. 파이프라인에서 이전에 생성한 모든 이미지는 원래 리포지토리에 그대로 남아 있습니다.

파이프라인 세부 정보 페이지에서 배포 설정을 편집하려면 다음 단계를 따르세요.

1. 배포 설정 탭을 선택합니다.
2. 배포 세부 정보 패널의 오른쪽 상단에서 편집을 선택합니다.
3. 배포 설정에 적용한 업데이트를 저장할 준비가 되면 변경 내용 저장을 선택합니다.

파이프라인의 빌드 일정 편집

파이프라인 편집 페이지에는 파이프라인을 생성한 후 편집할 수 있는 다음과 같은 세부 정보가 포함되어 있습니다.

- 파이프라인 설명.
- 향상된 메타데이터 수집. 이 옵션은 기본적으로 활성화되어 있습니다. 이 기능을 끄려면 향상된 메타데이터 수집 활성화 확인란의 선택을 취소하세요.
- 파이프라인의 빌드 일정. 이 섹션에서 내 일정 옵션과 모든 설정을 변경할 수 있습니다.

파이프라인 세부 정보 페이지에서 파이프라인을 편집하려면 다음 단계를 따르세요.

1. 파이프라인 세부 정보 페이지의 오른쪽 상단에서 작업을 선택한 다음 파이프라인 편집을 선택합니다.
2. 업데이트를 저장할 준비가 되면 변경 내용 저장을 선택합니다.

> **Note**
>
> cron 표현식을 사용하여 빌드 일정 설정에 대한 자세한 내용은 Image Builder에서 cron 표현식 사용 섹션을 참조하세요.

에서 컨테이너 이미지 파이프라인 업데이트 AWS CLI

JSON 파일을 AWS CLI의 update-image-pipeline 명령에 대한 입력으로 사용하여 컨테이너 이미지 파이프라인을 업데이트할 수 있습니다. JSON 파일을 구성하려면 다음과 같은 기존 리소스를 참조할 수 있는 Amazon 리소스 이름(ARN)이 있어야 합니다.

- 업데이트할 이미지 파이프라인
- 컨테이너 레시피
- 인프라 구성
- 배포 설정(현재 파이프라인에 포함된 경우)

> **Note**
>
> 배포 설정 리소스가 포함된 경우 명령이 실행되는 리전(리전 1)의 배포 설정에서 대상 리포지토리로 지정된 ECR 리포지토리가 컨테이너 레시피에 지정된 대상 리포지토리보다 우선합니다.

다음 단계에 따라 AWS CLI의 update-image-pipeline 명령을 사용하여 컨테이너 이미지 파이프라인을 업데이트하세요.

> **Note**
>
> UpdateImagePipeline은 파이프라인에 대한 선택적 업데이트를 지원하지 않습니다. 업데이트 요청에는 변경된 속성뿐만 아니라 모든 필수 속성을 지정해야 합니다.

1. CLI 입력 JSON 파일 생성

   자주 사용하는 파일 편집 도구를 사용하여 다음 키와 환경에 적합한 값을 포함하는 JSON 파일을 만드세요. 이 예제에서는 `create-component.json`(이)라는 이름의 파일이 사용됩니다.

```
{
 "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-
pipeline/my-example-pipeline",
 "containerRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:container-
recipe/my-example-recipe/2020.12.08",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-
configuration",
 "imageTestsConfiguration": {
  "imageTestsEnabled": true,
  "timeoutMinutes": 120
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * MON *)",
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "DISABLED"
}
```

> **Note**
>
> - JSON 파일 경로의 시작 부분에 `file://` 표기법을 포함시켜야 합니다.
> - JSON 파일의 경로는 명령을 실행하는 기본 운영 체제에 대한 적절한 규칙을 따라야 합니다. 예를 들어 Windows에서는 디렉터리 경로를 참조하기 위해 백슬래시(\)를 사용하고 Linux 및 macOS에서는 슬래시(/)를 사용합니다.

2. 작성한 파일을 입력으로 사용하여 다음 명령을 실행합니다.

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

Image Builder의 이미지 파이프라인 워크플로 구성

이미지 워크플로를 사용하면 파이프라인이 실행하는 워크플로를 사용자 지정하여 필요에 따라 이미지를 빌드하고 테스트할 수 있습니다. 정의한 워크플로는 Image Builder 워크플로 프레임워크의 컨텍스트 내에서 실행됩니다. 워크플로 프레임워크를 구성하는 단계에 대한 자세한 내용은 Image Builder 이미지의 빌드 및 테스트 워크플로 관리 섹션을 참조하세요.

빌드 워크플로

빌드 워크플로는 워크플로 프레임워크의 `Build` 단계에서 실행됩니다. 파이프라인에는 하나의 빌드 워크플로만 지정할 수 있습니다. 또는 빌드를 완전히 건너뛰고 테스트 전용 파이프라인을 구성할 수도 있습니다.

테스트 워크플로

테스트 워크플로는 워크플로 프레임워크의 `Test` 단계에서 실행됩니다. 파이프라인에 최대 10개의 테스트 워크플로를 지정할 수 있습니다. 파이프라인 빌드만 하려는 경우 테스트를 완전히 건너뛸 수도 있습니다.

테스트 워크플로를 위한 테스트 그룹 정의

테스트 워크플로는 테스트 그룹 내에서 정의됩니다. 파이프라인에 최대 10개의 테스트 워크플로를 실행할 수 있습니다. 테스트 워크플로를 특정 순서로 실행할지 아니면 동시에 최대한 많이 실행할지 결정합니다. 테스트 그룹을 어떻게 정의하느냐에 따라 실행 방식이 달라집니다. 다음 시나리오는 테스트 워크플로를 정의할 수 있는 여러 가지 방법을 보여줍니다.

> **Note**
>
> 콘솔을 사용하여 워크플로를 만드는 경우 테스트 그룹을 정의하기 전에 테스트 워크플로를 실행할 방법을 계획하는 것이 좋습니다. 콘솔에서 테스트 워크플로와 그룹을 추가하거나 제거할 수 있지만 순서를 바꿀 수는 없습니다.

시나리오 1: 한 번에 하나의 테스트 워크플로 실행

모든 테스트 워크플로를 한 번에 하나씩 실행할 경우 테스트 그룹을 최대 10개까지 구성할 수 있으며, 각 테스트 그룹에는 단일 테스트 워크플로가 포함됩니다. 테스트 그룹은 파이프라인에 추가하는 순서대로 한 번에 하나씩 실행됩니다. 이는 테스트 워크플로를 특정 순서로 한 번에 하나씩 실행하게 하는 한 가지 방법입니다.

시나리오 2: 여러 테스트 워크플로를 동시에 실행

순서가 중요하지 않고 최대한 많은 테스트 워크플로를 동시에 실행하려는 경우 단일 테스트 그룹을 구성하고 여기에 최대 개수의 테스트 워크플로를 넣을 수 있습니다. Image Builder는 동시에 최대 5개의 테스트 워크플로를 시작하고 다른 워크플로가 완료되면 추가 테스트 워크플로를 시작합니다. 테스트 워크플로를 최대한 빨리 실행하는 것이 목표라면 이 방법으로 달성할 수 있습니다.

시나리오 3: 믹스 앤 매치

동시에 실행할 수 있는 테스트 워크플로와 한 번에 하나씩 실행해야 하는 테스트 워크플로가 혼합된 시나리오의 경우 이 목표를 달성하도록 테스트 그룹을 구성할 수 있습니다. 테스트 그룹을 구성하는 방법에서 유일한 한도는 파이프라인에서 실행할 수 있는 테스트 워크플로의 최대 수입니다.

콘솔에서 Image Builder 파이프라인의 워크플로 파라미터 설정

워크플로 파라미터는 빌드 워크플로와 테스트 워크플로에서 동일한 방식으로 작동합니다. 파이프라인을 만들거나 업데이트할 때 포함할 빌드 및 테스트 워크플로를 선택합니다. 선택한 워크플로의 워크플로 문서에서 파라미터를 정의한 경우 Image Builder가 파라미터 패널에 해당 파라미터를 표시합니다. 파라미터가 정의되지 않은 워크플로에서는 패널이 숨겨집니다.

각 파라미터에는 워크플로 문서에서 정의한 다음 속성이 표시됩니다.

- 이름(편집 불가)-파라미터 이름입니다.
- 유형(편집 불가) – 파라미터 값의 데이터 유형입니다.
- 값 – 파라미터의 값입니다. 파이프라인에 맞게 파라미터 값을 편집하여 설정할 수 있습니다.

Image Builder가 워크플로 작업을 실행하는 데 사용하는 IAM 서비스 역할 지정

이미지 워크플로를 실행하려면 Image Builder에 워크플로 작업을 수행할 권한이 있어야 합니다. 다음과 같이 AWSServiceRoleForImageBuilder 서비스 연결 역할을 지정하거나 서비스 액세스를 위한 사용자 지정 역할을 지정할 수 있습니다.

- 콘솔-파이프라인 마법사 3단계 이미지 생성 프로세스 정의에서 서비스 액세스 패널의 IAM 역할 목록에서 서비스 연결 역할 또는 사용자 지정 역할을 선택합니다.
- Image Builder API-CreateImage 작업 요청에서 서비스 연결 역할 또는 사용자 지정 역할을 `executionRole` 파라미터 값으로 지정합니다.

서비스 역할을 생성하는 방법에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 AWS 서비스에 권한을 위임할 역할 생성을 참조하세요.

이미지 파이프라인 실행

파이프라인의 수동 일정 옵션을 선택한 경우 수동으로 빌드를 시작할 때만 파이프라인이 실행됩니다. 자동 예약 옵션 중 하나를 선택한 경우 정기적으로 예약된 실행 사이에 수동으로 실행할 수도 있습니다. 예를 들어 보통 한 달에 한 번 실행되는 파이프라인이 있는데 이전 실행 후 2주 후에 구성 요소 중 하나에 업데이트를 통합해야 하는 경우 파이프라인을 수동으로 실행하도록 선택할 수 있습니다.

Console

이미지 빌더 콘솔의 파이프라인 세부 정보 페이지에서 파이프라인을 실행하려면 페이지 상단의 작업 메뉴에서 파이프라인 실행을 선택합니다. 파이프라인 시작 또는 오류 발생 시 페이지 상단에 상태 메시지가 표시됩니다.

1. 파이프라인 세부 정보 페이지의 왼쪽 상단에 있는 작업 메뉴에서 파이프라인 실행을 선택합니다.
2. 출력 이미지 탭의 상태 열에서 파이프라인의 현재 상태를 확인할 수 있습니다.

AWS CLI

다음 예제는 AWS CLI 에서 start-image-pipeline-execution 명령을 사용하여 이미지 파이프라인을 수동으로 시작하는 방법을 보여줍니다. 이 명령을 실행하면 파이프라인이 새 이미지를 빌드하고 배포합니다.

```
aws imagebuilder start-image-pipeline-execution --image-pipeline-arn
 arn:aws:imagebuilder:us-west-2:111122223333:image-pipeline/my-example-pipeline
```

빌드 파이프라인 실행 시 생성되는 리소스를 확인하려면 생성할 리소스 섹션을 참조하세요.

Image Builder에서 cron 표현식 사용

EC2 Image Builder용 cron 표현식을 사용하여 파이프라인의 기본 이미지 및 구성 요소에 적용되는 업데이트로 이미지를 새로 고치는 기간을 설정합니다. 파이프라인 새로 고침 기간은 cron 표현식에서 설정한 시간부터 시작됩니다. cron 표현식의 시간을 분 단위까지 설정할 수 있습니다. 파이프라인 빌드는 시작 시간에 또는 그 이후에 실행될 수 있습니다.

빌드가 실행되기 시작하는 데 몇 초 또는 최대 1분이 소요될 수 있습니다.

> **Note**
>
> Cron 표현식은 기본적으로 UTC(Universal Coordinated Time) 시간대를 사용하지만 시간대를 지정할 수도 있습니다. 협정 세계시(UTC)에 대한 자세한 내용과 시간대에 대한 오프셋을 찾으려면 전 세계 시간대 약어 목록을 참조하십시오.

Image Builder에서 Cron 표현식에 대해 지원되는 값

EC2 Image Builder는 6개의 필수 필드로 구성된 cron 형식을 사용합니다. 각 필드는 선행 공백 또는 후행 공백 없이 사이에 공백을 두고 다른 필드와 구분합니다.

<Minute> <Hour> <Day> <Month> <Day of the week> <Year>

다음 표에는 필수적인 cron 항목에 대해 지원되는 값이 나와 있습니다.

Cron 표현식에 대해 지원되는 값

필드	값	와일드카드
분	0-59	, - * /
시간	0-23	, - * /
일	1-31	, - * ? / L W
월	1-12 또는 jan-dec	, - * /
요일	1-7 또는 sun-sat	, - * ? L #
연도	1970-2199	, - * /

와일드카드

다음 표에서는 Image Builder가 cron 표현식에 와일드카드를 사용하는 방법을 설명합니다. 빌드를 시작하는 데 지정한 시간이 지난 후 최대 1분이 소요될 수 있음을 유의하세요.

Cron 표현식에 대해 지원되는 와일드카드

와일드카드	설명
,	,(쉼표) 와일드카드는 추가 값을 포함합니다. 예컨대, '월' 필드에서 `jan,feb,mar` 는 1월, 2월, 3월을 포함한다는 의미입니다.
-	-(대시) 와일드카드는 범위를 지정합니다. 예컨대, '일' 필드에서 `1-15`는 지정된 달의 1일에서 15일까지 포함한다는 의미입니다.
*	*(별표) 와일드카드는 필드의 모든 유효한 값을 포함합니다.
?	?(물음표) 와일드카드는 필드 값이 다른 설정에 종속되도록 지정합니다. '일' 및 '요일' 필드의 경우 하나를 지정하거나 가능한 모든 값(*)을 포함하는 경우 다른 필드는 ?(이)여야 합니다. 둘 다 지정할 수는 없습니다. 예를 들어, '일' 필드에 7을 입력하면(매월 7일에 빌드 실행) '요일' 위치에는 ?가 포함되어야 합니다.
/	/(슬래시) 와일드카드로 증분을 지정합니다. 예를 들어, 이틀에 한 번씩 빌드를 실행하려면 '일' 필드에 `*/2`(을)를 입력합니다.
L	날짜 필드 중 하나에 있는 L 와일드카드는 마지막 날을 지정합니다. 즉, 월에 따라 일의 경우 28~31일로, 요일의 경우 일요일을 지정합니다.
W	'일' 필드에서는 W 와일드카드로 어떤 한 평일을 지정할 수 있습니다. '일' 필드에서 `W` 앞에 숫자를 입력하면 해당 날짜와 가장 가까운 평일을 대상으로 지정하려는 것입니다. 예를 들어, `3W`를 지정하는 경우 해당 월의 3일과 가장 가까운 평일에 빌드가 실행되도록 할 수 있습니다.

와일드카드	설명
#	#(해시)는 요일 필드에만 허용되며 그 뒤에 1에서 5 사이의 숫자가 와야 합니다. 숫자는 지정된 월에서 빌드 실행을 적용할 주를 지정합니다. 예를 들어, 매월 두 번째 금요일에 빌드를 실행하려면 요일 필드에 `fri#2`(을)를 사용합니다.

제한 사항

- 같은 cron 표현식에서 '날짜' 및 '요일' 필드를 지정할 수 없습니다. 이들 필드 중 하나에 값 또는 *(을)를 지정하는 경우에는 다른 필드에서 반드시 ?(을)를 사용해야 합니다.
- 1분보다 빠른 속도로 이어지는 cron 표현식은 지원되지 않습니다.

Image Builder의 cron 표현식 예제

Image Builder 콘솔에서는 cron 표현식이 API 또는 CLI와 다르게 입력됩니다. 예제를 보려면 해당하는 탭을 선택하세요.

Image Builder console

다음 예제는 빌드 일정에 맞게 콘솔에 입력할 수 있는 cron 표현식을 보여줍니다. 협정 세계시(UTC)는 24시간제를 사용하여 지정합니다.

매일 오전 10시(UTC)에 실행

```
0 10 * * ? *
```

매일 오후 12시 15분(UTC)에 실행

```
15 12 * * ? *
```

매일 자정(UTC)에 실행

```
0 0 * * ? *
```

매주 평일 오전 10시(UTC)에 실행

```
0 10 ? * 2-6 *
```

매주 평일 오후 6시(UTC)에 실행

```
0 18 ? * mon-fri *
```

매월 1일 오전 8시(UTC)에 실행

```
0 8 1 * ? *
```

매월 두 번째 화요일 오후 10시 30분(UTC)에 실행

```
30 22 ? * tue#2 *
```

> **Tip**
> 파이프라인 작업이 실행되는 동안 다음 날까지 연장되지 않도록 하려면, 시작 시간을 지정할 때 빌드 시간을 고려해야 합니다.

API/CLI

다음 예제는 CLI 명령 또는 API 요청을 사용하여 빌드 일정에 입력할 수 있는 cron 표현식을 보여줍니다. cron 표현식만 표시됩니다.

매일 오전 10시(UTC)에 실행

```
cron(0 10 * * ? *)
```

매일 오후 12시 15분(UTC)에 실행

```
cron(15 12 * * ? *)
```

매일 자정(UTC)에 실행

```
cron(0 0 * * ? *)
```

매주 평일 오전 10시(UTC)에 실행

```
cron(0 10 ? * 2-6 *)
```

매주 평일 오후 6시(UTC)에 실행

`cron(0 18 ? * mon-fri *)`

매월 1일 오전 8시(UTC)에 실행

`cron(0 8 1 * ? *)`

매월 두 번째 화요일 오후 10시 30분(UTC)에 실행

`cron(30 22 ? * tue#2 *)`

> **Tip**
>
> 파이프라인 작업이 실행되는 동안 다음 날까지 연장되지 않도록 하려면, 시작 시간을 지정할 때 빌드 시간을 고려해야 합니다.

Image Builder의 rate 표현식

rate 표현식은 예약된 이벤트 규칙을 생성할 때 시작되며, 정의된 예약 일정에 따라 실행됩니다.

rate 표현식에는 필수 필드가 2개 있습니다. 각 필드는 공백으로 구분됩니다.

구문

```
rate(value unit)
```

값

양수.

unit

시간 단위. 1의 값(예: `minute`)과 1을 초과하는 값(예: `minutes`)은 서로 다른 단위가 필요합니다.

유효값: 분 | 분 | 시간 | 시간 | 일 | 일

제한 사항

값이 1(와)과 같을 경우에는 단위가 단수여야 합니다. 마찬가지로, 1보다 큰 값에 대해서는 단위가 복수여야 합니다. 예를 들어 `rate(1 hours)`(와)과 `rate(5 hour)`은(는) 잘못된 식이며 `rate(1 hour)`(와)과 `rate(5 hours)`은(는) 유효한 식입니다.

Image Builder 파이프라인과 함께 EventBridge 규칙 사용

다양한 AWS 및 파트너 서비스의 이벤트는 거의 실시간으로 Amazon EventBridge 이벤트 버스로 스트리밍됩니다. 또한 사용자 정의 이벤트를 생성하고 자체 애플리케이션에서 EventBridge로 이벤트를 전송할 수 있습니다. 이벤트 버스는 규칙을 사용하여 이벤트 데이터를 라우팅할 위치를 결정합니다.

Image Builder 파이프라인은 EventBridge 규칙 대상으로 사용할 수 있습니다. 즉, 버스 또는 일정에 따라 이벤트에 응답하기 위해 생성한 규칙을 기반으로 Image Builder 파이프라인을 실행할 수 있습니다.

Image Builder가 EventBridge에 전송하는 시스템 생성 이벤트의 요약은 Image Builder에서 보내는 이벤트 메시지 섹션을 참조하세요.

> **Note**
>
> 이벤트 버스는 리전별로 다릅니다. 규칙과 대상은 같은 리전에 있어야 합니다.

내용

- EventBridge 용어
- Image Builder 파이프라인에 대한 EventBridge 규칙 보기
- EventBridge 규칙을 사용하여 파이프라인 빌드를 예약하세요.

EventBridge 용어

이 섹션에는 EventBridge가 Image Builder 파이프라인과 통합되는 방식을 이해하는 데 도움이 되는 용어 요약이 포함되어 있습니다.

Event

하나 이상의 애플리케이션 리소스에 영향을 미칠 수 있는 환경 변화에 대해 설명합니다. 환경은 AWS 환경, SaaS 파트너 서비스 또는 애플리케이션, 애플리케이션 또는 서비스 중 하나일 수 있습니다. 타임라인에서 예정된 이벤트를 설정할 수도 있습니다.

이벤트 버스

애플리케이션과 서비스로부터 이벤트 데이터를 수신하는 파이프라인

소스

이벤트를 이벤트 버스로 보낸 서비스 또는 애플리케이션

대상

EventBridge가 규칙과 일치할 때 호출하여 이벤트의 데이터를 대상으로 전달하는 리소스 또는 엔드포인트

규칙

규칙은 들어오는 이벤트에서 일치하는 것을 찾아서 대상으로 전송하여 처리합니다. 단일 규칙으로 이벤트를 여러 대상으로 전송한 다음, 병렬로 실행할 수 있습니다. 규칙은 이벤트 패턴 또는 일정을 기반으로 합니다.

패턴

이벤트 패턴은 대상 작업을 시작하기 위해 규칙이 일치시키는 이벤트 구조와 필드를 정의합니다.

일정

일정 규칙은 Image Builder 파이프라인을 실행하여 분기별로 이미지를 새로 고치는 등의 작업을 일정에 따라 수행합니다. 일정 표현식에는 두 가지 유형이 있습니다.

- cron 표현식 - 간단한 기준(예: 특정 날짜에 매주 실행)을 설명하는 cron 구문을 사용하여 특정 일정 기준과 일치시킵니다. 분기별로 매월 5일인 오전 2시에서 오전 4시 사이에 실행하는 것과 같이 더 복잡한 기준을 설정할 수도 있습니다.
- rate 표현식 - 대상을 호출할 때 일정한 간격(예: 12시간마다)을 지정합니다.

Image Builder 파이프라인에 대한 EventBridge 규칙 보기

Image Builder 이미지 파이프라인 세부 정보 페이지의 EventBridge 규칙 탭에는 사용자 계정이 액세스할 수 있는 EventBridge 이벤트 버스와 현재 파이프라인에 적용되는 선택된 이벤트 버스에 대한 규칙이 표시됩니다. 또한 이 탭은 새 리소스를 생성할 수 있는 EventBridge 콘솔로 직접 연결됩니다.

EventBridge 콘솔에 연결된 액션

- 이벤트 버스 생성
- 규칙 생성

EventBridge에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 다음 주제를 참조하세요.

- Amazon EventBridge란 무엇인가요?
- Amazon EventBridge 이벤트 버스
- Amazon EventBridge 이벤트

- Amazon EventBridge 규칙

EventBridge 규칙을 사용하여 파이프라인 빌드를 예약하세요.

이 예제에서는 rate 표현식을 사용하여 기본 이벤트 버스에 대한 새 일정 규칙을 생성합니다. 이 예제의 규칙은 90일마다 이벤트 버스에 이벤트를 생성합니다. 이벤트는 이미지를 새로 고치는 파이프라인 빌드를 시작합니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 계정에서 생성된 이미지 파이프라인 목록을 보려면 탐색 창에서 이미지 파이프라인을 선택합니다.

   > **Note**
   >
   > 이미지 파이프라인 목록에는 파이프라인에서 생성된 출력 이미지 유형(AMI 또는 Docker)에 대한 표시기가 포함됩니다.

3. 세부 정보를 보거나 파이프라인을 편집하려면 파이프라인 이름 링크를 선택합니다. 파이프라인의 세부 정보 보기가 열립니다.

   > **Note**
   >
   > 또한 파이프라인 이름 옆에 있는 확인란을 선택한 다음 세부 정보 보기를 선택할 수 있습니다.

4. EventBridge 규칙 탭을 엽니다.
5. 이벤트 버스 패널에서 미리 선택한 기본 이벤트 버스는 그대로 둡니다.
6. 규칙 생성을 선택합니다. 그러면 Amazon EventBridge 콘솔의 규칙 생성 페이지로 이동합니다.
7. 규칙에 대해 이름과 설명을 입력하세요. 규칙 이름은 선택한 리전에 대한 이벤트 버스 내에서 고유해야 합니다.
8. 패턴 정의 패널에서 일정 옵션을 선택합니다. 이렇게 하면 패널이 확장되고 모든 고정 요금 옵션이 선택됩니다.
9. 첫 번째 상자에 90을(를) 입력하고, 드롭다운 목록에서 일수를 선택합니다.
10. 대상 선택 패널에서 다음 작업을 수행합니다.

    a. 대상 드롭다운 목록에서 EC2 Image Builder을(를) 선택합니다.

b. Image Builder 파이프라인에 규칙을 적용하려면 이미지 파이프라인 드롭다운 목록에서 대상 파이프라인을 선택합니다.

c. EventBridge에는 선택한 파이프라인의 빌드를 시작할 권한이 필요합니다. 이 예제에서는 이 특정 리소스에 대해 새 역할 생성을 기본 옵션으로 설정합니다.

d. 대상 추가를 선택합니다.

11. 생성을 선택합니다.

> **Note**
>
> 이 예제에서 다루지 않은 rate 표현식 규칙의 설정에 대해 자세히 알아보려면 Amazon EventBridge 사용 설명서의 rate 표현식을 참조하세요.

Image Builder에서 제품 및 서비스 통합

EC2 Image Builder는 AWS Marketplace 및 기타 AWS 서비스 및 애플리케이션과 통합되어 강력하고 안전한 사용자 지정 머신 이미지를 생성하는 데 도움이 됩니다.

Products

Image Builder 레시피는 AWS Marketplace 및 Image Builder 관리형 구성 요소의 이미지 제품을 통합하여 다음과 같이 특수한 빌드 및 테스트 기능을 제공할 수 있습니다.

- AWS Marketplace 이미지 제품 - CIS Hardening과 같은 조직 표준을 충족하기 위해 레시피의 기본 이미지 AWS Marketplace 로의 이미지 제품을 사용합니다. Image Builder 콘솔에서 레시피를 생성할 때 기존 구독에서 선택하거나 AWS Marketplace의 특정 제품을 검색할 수 있습니다. Image Builder API, CLI 또는 SDK에서 레시피를 생성할 때 기본 이미지로 사용할 이미지 제품 Amazon 리소스 이름(ARN)을 지정할 수 있습니다.
- Image Builder 구성 요소 - 레시피에 지정하는 구성 요소는 빌드 및 테스트 작업(예: 소프트웨어 설치 또는 규정 준수 검증 수행)을 수행할 수 있습니다. AWS Marketplace 에서 구독하는 일부 이미지 제품에는 레시피에 사용할 수 있는 보조 구성 요소가 포함될 수 있습니다. CIS 강화 이미지에는 레시피에서 구성에 CIS 벤치마크 레벨 1 지침을 적용하는 데 사용할 수 있는 일치하는 AWSTOE 구성 요소가 포함되어 있습니다.

> **Note**
>
> 규정 준수 관련 제품에 대한 자세한 내용은 Image Builder 이미지를 위한 규정 준수 제품 섹션을 참조하세요.

서비스

Image Builder는 다음과 통합되어 자세한 이벤트 지표, 로깅 및 모니터링을 AWS 서비스 제공합니다. 이 정보는 활동을 추적하고, 이미지 빌드 문제를 해결하고, 이벤트 알림을 기반으로 자동화를 생성하는 데 도움이 됩니다.

- AWS Organizations - 조직의 계정에 서비스 제어 정책(SCP)을 AWS Organizations 적용할 수 있습니다. 개별 정책을 생성, 관리, 활성화 및 비활성화할 수 있습니다. 다른 모든 AWS 아티팩트 및 서비스와 마찬가지로 Image Builder는 승인된 AMIs만 있는 인스턴스를 시작하도록 멤버 계정에 제약 조건을 적용하는 등 일반적인 시나리오에 대해 템플릿 SCPs를 AWS Organizations AWS 제공합니다.

- AWS CloudTrail - CloudTrail로 전송되는 Image Builder 이벤트를 모니터링합니다. CloudTrail 통합과 Image Builder에 대한 자세한 내용은 CloudTrail을 사용하여 Image Builder API 직접 호출 로깅 섹션을 참조하세요.

  설정 방법 및 로그 파일을 찾는 방법을 비롯한 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

- Amazon CloudWatch Logs – CloudWatch를 사용하여 Image Builder 로그 파일을 모니터링, 저장 및 액세스합니다. 선택적으로 S3 버킷으로 로그를 저장할 수도 있습니다. CloudWatch와 Image Builder 통합에 대한 자세한 내용은 Amazon CloudWatch Logs를 사용하여 Image Builder 로그 모니터링 섹션을 참조하세요.

  CloudWatch Logs에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 Amazon CloudWatch Logs란 무엇입니까?를 참조하세요.

- Amazon Elastic Container Registry(Amazon ECR) – Amazon ECR은 안전하고 확장 가능하고 신뢰할 수 있는 관리형 AWS 컨테이너 이미지 레지스트리 서비스입니다. Image Builder로 생성한 컨테이너 이미지는 소스 리전(빌드가 실행되는 리전)의 Amazon ECR 및 컨테이너 이미지를 배포하는 모든 리전에 저장됩니다. Amazon ECR에 대한 자세한 내용은 Amazon Elastic Container Registry 사용 설명서를 참조합니다.

- Amazon EventBridge - 계정의 Image Builder 활동에서 수집한 실시간 이벤트 데이터 스트림에 연결합니다. Eventbridge에 대한 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge란 무엇입니까?를 참조하세요.

- Amazon Inspector-Image Builder가 새 이미지 생성을 시작하는 EC2 테스트 인스턴스를 자동으로 스캔하여 소프트웨어 및 네트워크 설정의 취약성을 발견합니다. Image Builder는 출력 이미지 리소스의 결과를 저장하므로 테스트 인스턴스가 종료된 후 조사하고 수정할 수 있습니다. 스캔 및 가격 책정에 대한 자세한 내용은 Amazon Inspector 사용 설명서의 Amazon Inspector란 무엇입니까?를 참조하세요.

  또한 Amazon Inspector는 향상된 스캔을 구성한 경우 ECR 리포지토리를 스캔할 수 있습니다. 자세한 내용은 Amazon Inspector 사용 설명서의 Amazon ECR 컨테이너 이미지 스캔을 참조하세요.

  > **Note**
  >
  > Amazon Inspector는 유료 기능입니다.

- AWS License Manager - 배포 프로세스 중에 License Manager 자체 관리형 라이선스를 출력 AMI에 연결할 수 있습니다. 대상 리전에 대해 지정하는 라이선스는 해당 리전에 이미 있어야 합니다. 자체 관리형 라이선스에 대한 자세한 내용은 License Manager의 자체 관리형 라이선스를 참조하세요.

- AWS Marketplace - 현재 AWS Marketplace 제품 구독 목록을 확인하고 Image Builder에서 직접 이미지 제품을 검색할 수 있습니다. 구독한 이미지 제품을 Image Builder 레시피의 기본 이미지로 사용할 수도 있습니다. AWS Marketplace 구독 관리에 대한 자세한 내용은 AWS Marketplace 구매자 안내서의 제품 구매를 참조하세요.
- AWS Resource Access Manager (AWS RAM) - AWS RAM를 사용하면 AWS 계정 또는를 통해 리소스를 공유할 수 있습니다 AWS Organizations. 가 여러 개 있는 경우 리소스를 중앙에서 생성하고 AWS RAM 를 사용하여 해당 리소스를 다른 계정과 공유할 AWS 계정수 있습니다. EC2 Image Builder를 사용하면 구성 요소, 이미지 및 이미지 레시피와 같은 리소스를 공유할 수 있습니다. 에 대한 자세한 내용은 AWS Resource Access Manager 사용 설명서를 AWS RAM참조하세요. Image Builder 리소스 공유에 대한 자세한 내용은 와 Image Builder 리소스 공유 AWS RAM(을)를 참조합니다.
- Amazon Simple Notification Service(SNS) - 구성된 경우 구독하는 SNS 주제에 이미지 상태에 대한 자세한 메시지를 게시할 수 있습니다. Amazon SNS에 대한 자세한 내용은 Amazon Simple Notification Service 개발자 설명서의 Amazon SNS란 무엇입니까?를 참조하세요.

제품 및 서비스 통합 주제

- Image Builder에서의 Amazon EventBridge 통합
- Image Builder에 Amazon Inspector 통합
- AWS Marketplace Image Builder의 통합
- Image Builder에서의 Amazon SNS 통합
- Image Builder 이미지를 위한 규정 준수 제품

Image Builder에서의 Amazon EventBridge 통합

Amazon EventBridge는 Image Builder 애플리케이션을 다른 AWS 서비스의 관련 데이터와 연결하는 데 사용할 수 있는 서버리스 이벤트 버스 서비스입니다. EventBridge에서 규칙은 들어오는 이벤트에서 일치하는 것을 찾아서 대상으로 전송하여 처리합니다. 단일 규칙으로 이벤트를 여러 대상으로 전송할 수 있으며 이러한 이벤트는 병렬로 실행됩니다.

EventBridge를 사용하면를 자동화 AWS 서비스 하고 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. AWS 서비스 의 이벤트는 거의 실시간으로 EventBridge로 전송됩니다. 수신 이벤트에 반응하여 작업을 시작하는 규칙을 설정할 수 있습니다. 예를 들어, EC2 인스턴스의 상태가 보류 중에서 실행 중으로 변경될 때 Lambda 함수에 이벤트를 보내는 등의 작업을 시작할 수 있습니다. 이를 패턴이라고 합니다. 이벤트 패턴에 기반을 둔 규칙을 생성

하는 방법에 대한 자세한 정보는 Amazon EventBridge 사용 설명서의 이벤트에 대응하는 Amazon EventBridge 규칙 생성을 참조하세요.

자동으로 실행할 수 있는 작업은 다음과 같습니다.

- AWS Lambda 함수 호출
- Amazon EC2 Run Command 호출
- Amazon Kinesis Data Streams로 이벤트 릴레이
- AWS Step Functions 상태 시스템 활성화
- Amazon SNS 주제 또는 Amazon SQS 대기열 알림

또한 Image Builder 파이프라인을 실행하여 분기별로 이미지를 새로 고치는 등의 작업을 정기적으로 수행하도록 기본 이벤트 버스에 대한 스케줄링 규칙을 설정할 수 있습니다. 일정 표현식에는 두 가지 유형이 있습니다.

- cron 표현식 - 다음은 UTC+0 정오에 작업이 실행되도록 스케줄을 지정하는 cron 표현식 예제입니다.

  ```
  cron(0 12 * * ? *)
  ```

  EventBridge에서의 cron 표현식에 대한 자세한 정보는 Amazon EventBridge 사용 설명서의 Cron 표현식을 참조하세요.
- rate 표현식 - rate 표현식의 다음 예는 12시간마다 작업이 실행되도록 스케줄링합니다.

  ```
  rate(12 hour)
  ```

  EvenBridge에서의 rate 표현식 사용에 대한 자세한 정보는 Amazon EventBridge 사용 설명서의 rate 표현식을 참조하세요.

EventBridge 규칙이 Image Builder 이미지 파이프라인과 통합되는 방법에 대한 자세한 내용은 Image Builder 파이프라인과 함께 EventBridge 규칙 사용 섹션을 참조하세요.

Image Builder에서 보내는 이벤트 메시지

Image Builder는 Image Builder 리소스의 상태가 크게 변경되면 EventBridge로 이벤트 메시지를 전송합니다. 예를 들어 이미지에 대한 상태가 변경된 경우입니다. 다음 예제에서는 Image Builder가 보낼 수 있는 일반적인 JSON 이벤트 메시지를 보여줍니다.

EC2 Image Builder 이미지 상태 변경

Image Builder는 이미지 생성 중에 이미지 리소스의 상태가 변경될 때 이 이벤트를 전송합니다. 예를 들어 이미지 상태가 다음과 같이 한 상태에서 다른 상태로 변경되는 경우:

- `building`에서 `testing`으로
- `testing`에서 `distribution`으로
- `testing`에서 `failed`으로
- `integrating`에서 `available`으로

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder Image State Change",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2024-01-18T17:50:56Z",
    "region": "us-west-2",
    "resources": ["arn:aws:imagebuilder:us-west-2:111122223333:image/
cmkencryptedworkflowtest-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/1.0.0/1"],
    "detail": {
        "previous-state": {
            "status": "TESTING"
        },
        "state": {
            "status": "AVAILABLE"
        }
    }
}
```

EC2 Image Builder CVE 감지됨

이미지에 대해 CVE 감지를 활성화한 경우 이미지 스캔이 완료될 때마다 Image Builder가 결과와 함께 메시지를 보냅니다.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder CVE Detected",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2023-03-01T16:59:09Z",
```

```
    "region": "us-east-1",
    "resources": [
        "arn:aws:imagebuilder:us-east-1:111122223333:image/test-image/1.0.0/1",
        "arn:aws:imagebuilder:us-east-1:111122223333:image-pipeline/test-pipeline"
    ],
    "detail": {
        "resource-id": "i-1234567890abcdef0",
        "finding-severity-counts": {
            "all": 0,
            "critical": 0,
            "high": 0,
            "medium": 0
        }
    }
}
```

EC2 Image Builder 워크플로 단계 대기

Image Builder는 `WaitForAction` 워크플로 단계가 일시 중지되어 비동기 작업이 완료될 때까지 기다리는 메시지를 보냅니다.

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder Workflow Step Waiting",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2024-01-18T16:54:44Z",
    "region": "us-west-2",
    "resources": ["arn:aws:imagebuilder:us-west-2:111122223333:image/
workflowstepwaitforactionwithvalidsnstopictest-a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222/1.0.0/1", "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
build-workflow-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333/1.0.0/1"],
    "detail": {
        "workflow-execution-id": "wf-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "workflow-step-execution-id": "step-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "workflow-step-name": "TestAutoSNSStop"
    }
}
```

Image Builder에 Amazon Inspector 통합

Amazon Inspector로 보안 스캔을 활성화하면 계정의 시스템 이미지와 실행 중인 인스턴스를 지속적으로 스캔하여 운영 체제 및 프로그래밍 언어 취약성을 확인합니다. 활성화된 경우 보안 검색이 자동으로 수행되며 Image Builder는 새 이미지를 생성할 때 테스트 인스턴스를 통한 결과의 스냅샷을 저장할 수 있습니다. Amazon Inspector는 유료 서비스입니다.

Amazon Inspector는 소프트웨어 또는 네트워크 설정에서 취약성을 발견하면 다음 조치를 취합니다.

- 결과가 있었음을 알려줍니다.
- 결과의 심각도를 평가합니다. 심각도 등급은 발견의 우선 순위를 정하는 데 도움이 되는 취약성을 분류하며 다음 값을 포함합니다.
  - 분류되지 않음
  - 정보
  - 낮음
  - 중간
  - 높음
  - 심각
- 조사 결과에 대한 정보를 제공하고 자세한 내용을 확인할 수 있는 추가 리소스 링크를 제공합니다.
- 결과를 생성한 문제를 해결하는 데 도움이 되는 수정 지침을 제공합니다.

보안 스캔 구성

계정에 대해 Amazon Inspector를 활성화한 경우, Amazon Inspector는 Image Builder가 시작하는 EC2 인스턴스를 자동으로 스캔하여 새 이미지를 구축하고 테스트합니다. 이러한 인스턴스는 구축 및 테스트 프로세스 동안 수명이 짧으며 일반적으로 해당 인스턴스가 종료되는 즉시 결과가 만료됩니다. 새 이미지에 대한 결과를 조사하고 수정하는 데 도움이 되도록 Image Builder는 Amazon Inspector가 빌드 프로세스 중에 테스트 인스턴스에서 확인한 모든 결과를 스냅샷으로 저장할 수 있습니다.

파이프라인의 보안 스캔을 구성하려면 에서 Image Builder 이미지에 대한 보안 스캔 구성 AWS Management Console(을)를 참조하세요.

보안 결과 검토

Image Builder 콘솔에서는 모든 Image Builder 리소스에 대한 보안 결과를 한 곳에서 볼 수 있습니다. 보안 개요 섹션의 보안 결과 페이지에서 모든 결과를 보거나 취약성, 이미지 파이프라인 또는 이미지별로 결과를 그룹화할 수 있습니다. 콘솔에는 기본적으로 모든 보안 탐지 결과가 표시됩니다. 모든 보안

결과 옵션의 요약 패널에는 각 심각도 수준에 해당하는 검색 결과 수가 표시됩니다. 자세한 내용은 에서 Image Builder 이미지의 보안 조사 결과 관리 AWS Management Console 단원을 참조하십시오.

Amazon Inspector 취약성 발견에 대해 자세히 알아보려면 Amazon Inspector 사용 설명서의 Amazon Inspector 조사 결과 이해를 참조하세요.

AWS Marketplace Image Builder의 통합

AWS Marketplace 는 비즈니스 요구 사항에 맞는 솔루션을 구축하는 데 도움이 되는 타사 소프트웨어, 데이터 및 서비스를 찾고 구독할 수 있는 엄선된 디지털 카탈로그입니다.는 보안, 네트워킹, 스토리지, 기계 학습 등과 같은 인기 있는 범주의 소프트웨어 목록과 함께 인증된 구매자 및 등록된 판매자를 AWS Marketplace 제공합니다.

AWS Marketplace 판매자는 독립 소프트웨어 공급업체(ISV), 리셀러 또는 AWS 제품 및 서비스와 함께 사용할 제품을 제공하는 개인이 될 수 있습니다. 판매자는 제품을 제출할 때 제품 가격과 이용 약관을 AWS Marketplace정의합니다. 구매자는 제공되는 제품에 설정된 요금 및 이용 약관에 동의합니다. 자세한 내용은 란 무엇입니까?를 AWS Marketplace참조하십시오. AWS Marketplace

AWS Marketplace 통합 기능

Image Builder는와 통합되어 Image Builder 콘솔에서 직접 다음 기능을 AWS Marketplace 제공합니다.

- 에서 사용할 수 있는 이미지 제품을 검색합니다 AWS Marketplace.
- 구성 요소를 제공하는 AWS Marketplace 이미지 제품을 검색합니다.
- 현재 AWS Marketplace 제품 구독 목록을 확인합니다.
- Image Builder 레시피의 기본 AWS Marketplace 이미지로 구독한 이미지 제품을 사용합니다.
- Image Builder 레시피에서 구독한 AWS Marketplace 구성 요소를 사용합니다.

Image Builder는와 통합되어 구독한 이미지 제품 및 구성 요소를 AWS Marketplace 표시합니다. AWS Marketplace Image Builder 콘솔을 벗어나지 않고 제품 검색 페이지에서 이미지 제품 및 구성 요소를 검색할 수도 있습니다.

Image Builder가 생성하는 출력 AMI에는 AWS Marketplace 이미지 제품 및 구성 요소의 제품 코드가 포함됩니다. 최종 사용자 지정 이미지에 대해 최대 4개의 제품 코드를 보유할 수 있습니다.

AWS Marketplace Image Builder의 구독

Image Builder 콘솔의 AWS Marketplace 섹션에 있는 구독 페이지에는 현재 구독 중인 AWS Marketplace 제품 목록이 표시됩니다. 구독한 각 제품에는 다음과 같은 세부 정보가 표시됩니다.

- 제품 이름. 이는의 제품 세부 정보 페이지에 연결됩니다 AWS Marketplace. 구독한 제품의 제품 세부 정보 페이지가 브라우저의 새 탭에서 열립니다.
- 게시자입니다. 이는의 게시자 세부 정보 페이지에 연결됩니다 AWS Marketplace. 브라우저의 새 탭에서 게시자 세부 정보 페이지가 열립니다.
- 구독한 버전입니다.
- 구독한 제품에 연결된 구성 요소가 포함된 경우 Image Builder는 구성 요소 세부 정보에 대한 링크를 표시합니다.

페이지 상단에서 이름별로 특정 제품을 검색하거나 페이지 매김 컨트롤을 사용하여 결과 페이지를 넘길 수 있습니다. 새 레시피에서 구독한 이미지 제품을 사용하려면 구독한 제품을 선택하고 새 레시피 생성을 선택합니다. Image Builder는 기본적으로 목록에서 첫 번째 제품을 미리 선택합니다.

> **Note**
>
> 방금 구독한 제품을 찾고 있는데 목록에 표시되지 않는 경우 탭 상단의 새로 고침 버튼을 사용하여 결과를 새로 고치십시오. 목록에 새 구독이 표시될 때까지 몇 분 정도 걸릴 수 있습니다.

AWS Marketplace Image Builder 콘솔에서 이미지 제품 검색

이 섹션에서는 레시피에서 기본 AWS Marketplace 이미지로 사용할 이미지 제품에 중점을 둡니다. 연결된 소프트웨어 구성 요소가 포함된 제품의 경우 콘솔과 API, SDK 및 CLI에서 제품 소유자를 기준으로 필터링할 수 있습니다. 자세한 내용은 Image Builder 구성 요소 나열 단원을 참조하십시오. AWS Marketplace 구성 요소 검색, 구독 및 사용에 대한 자세한 내용은 섹션을 참조하세요AWS Marketplace 구성 요소를 사용하여 이미지 사용자 지정.

제품 검색

Image Builder 콘솔에서 AWS Marketplace 이미지 제품을 찾으려면 다음 단계를 따르세요.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창의 AWS Marketplace 섹션에서 제품 검색을 선택합니다.
3. 제품 검색 페이지의 이미지 제품 탭에서 이미지 제품을 검색할 수 있습니다.

   Image Builder는에서 제품을 사전 필터링 AWS Marketplace 하여 Image Builder 레시피에 사용할 수 있는 머신 이미지에 초점을 맞춥니다. Image Builder와의 AWS Marketplace 통합에 대한 자세한 내용을 보려면 보려는 것과 일치하는 탭을 선택합니다.

이 탭에는 두 개의 패널이 있습니다. 왼쪽에 있는 결과 수정 패널을 사용하면 결과를 필터링하여 구독하려는 제품을 찾을 수 있습니다. 오른쪽의 제품 검색 패널에는 필터 기준에 맞는 제품이 표시되며 제품 이름별로 검색하는 옵션도 있습니다.

결과 수정

다음 목록은 제품 검색에 적용할 수 있는 몇 가지 필터만 보여줍니다.

- 인프라 소프트웨어 또는 기계 학습과 같은 제품 범주를 하나 이상 선택합니다.
- 이미지 제품의 운영 체제를 선택하거나 특정 운영 체제 플랫폼에 맞는 모든 제품(예: 모든 Linux/Unix)을 선택합니다.
- 사용 가능한 제품을 표시할 게시자를 한 명 이상 선택하세요. 모두 보기 링크를 선택하면 적용한 필터에 맞는 제품을 보유한 모든 게시자가 표시됩니다.

> **Note**
>
> 게시자 이름은 알파벳 순서로 되어 있지 않습니다. 예를 들어 **`Center for Internet Security`** 같은 특정 게시자를 찾는 경우 모든 게시자 대화 상자 상단의 검색 상자에 이름의 일부를 입력할 수 있습니다. 이름을 약어로 입력해야 합니다. 예를 들어 `CIS` 같이 원하는 결과가 나오지 않을 수 있습니다.
> 게시자 이름을 페이지별로 찾아볼 수도 있습니다.

필터 선택은 동적입니다. 선택한 각 항목은 다른 모든 카테고리의 옵션에 영향을 줍니다. 에는 수천 개의 제품이 AWS Marketplace있으므로 필터링할 수 있을수록 원하는 제품을 찾을 가능성이 높아집니다.

제품 검색

이름으로 특정 제품을 찾으려면 이 패널 상단의 검색 창에 이름의 일부를 입력할 수 있습니다. 각 제품 결과에는 다음과 같은 세부 정보가 포함됩니다.

- 제품 이름 및 로고. 이 두 페이지 모두 AWS Marketplace의 제품 세부 정보 페이지에 연결되어 있습니다. 자세한 내용이 포함된 페이지는 기본 브라우저에서 새 탭으로 열립니다. Image Builder 레시피에 사용하려는 경우 여기에서 이미지 제품을 구독할 수 있습니다. 자세한 내용은 AWS Marketplace 구매자 가이드에서 제품 구매를 참조하세요.

에서 이미지 제품을 구독하는 경우 브라우저의 Image Builder 탭으로 다시 AWS Marketplace전환하고 구독한 이미지 제품 목록을 새로 고쳐 확인합니다.

> **Note**
>
> 새 구독이 가능해지기까지 몇 분 정도 걸릴 수 있습니다.

- 게시자 이름. 이는의 게시자 세부 정보 페이지에 연결됩니다 AWS Marketplace. 브라우저의 새 탭에서 게시자 세부 정보 페이지가 열립니다.
- 제품 버전.
- 제품 별점 및 AWS Marketplace내 제품 세부 정보 페이지의 리뷰 섹션으로 연결되는 직접 링크 자세한 내용이 포함된 페이지는 기본 브라우저에서 새 탭으로 열립니다.
- 제품 설명의 처음 몇 줄.

검색 창 바로 아래에서 검색 결과 수와 해당 결과 중 현재 표시된 결과 중 하위 집합을 확인할 수 있습니다. 패널 오른쪽에 있는 추가 컨트롤을 사용하여 한 번에 표시할 제품 수 및 결과에 적용할 정렬 순서에 대한 설정을 조정할 수 있습니다. 또한 페이지 매김 컨트롤을 사용하여 결과를 페이지로 이동할 수도 있습니다.

AWS Marketplace Image Builder 레시피에서 이미지 제품 사용

레시피 생성 페이지를 열고 다음과 같이 기본 AWS Marketplace 이미지로 사용할 이미지 제품을 선택합니다.

1. https://console.aws.amazon.com/imagebuilder/에서 EC2 Image Builder 콘솔을 엽니다.
2. 탐색 창에서 AWS Marketplace 섹션의 이미지 레시피를 선택합니다. 그러면 생성한 이미지 레시피 목록이 표시됩니다.
3. 이미지 레시피 생성을 선택합니다. 그러면 레시피 생성 페이지가 열립니다.
4. 레시피 세부 정보 섹션에서 평소와 같이 이름과 버전을 입력합니다.
5. 기본 이미지 섹션에서 AWS Marketplace 이미지 옵션을 선택합니다. 구독 탭에서 구독한 AWS Marketplace 이미지 제품의 목록을 보여줍니다. 목록에서 기본 이미지를 선택할 수 있습니다.

   AWS Marketplace 탭에서 AWS Marketplace 직접에서 사용할 수 있는 다른 이미지 제품을 검색할 수도 있습니다. 제품 추가를 선택하거나 AWS Marketplace 탭을 직접 엽니다. 에서 필터를 설정하

고 검색하는 방법에 대한 자세한 내용은 섹션을 AWS Marketplace참조하세요AWS Marketplace Image Builder 콘솔에서 이미지 제품 검색.

6. 평소와 같이 나머지 세부 정보를 입력합니다. 또는 제품 구독에 빌드 구성 요소가 포함된 경우 빌드 구성 요소 목록에서 해당 구성 요소를 선택할 수 있습니다. 구성 `AWS Marketplace` 요소 소유자 유형 목록에서 선택하여 보거나 CIS 구성 요소에 `Third party managed` 대해를 선택합니다.
7. 레시피 생성을 선택합니다.

   최종 이미지에는 AWS Marketplace 이미지 제품 및 구성 요소의 제품 코드가 최대 4개까지 포함될 수 있습니다. 선택한 기본 이미지와 구성 요소에 4개 이상의 제품 코드가 포함된 경우 레시피를 생성하려고 하면 Image Builder가 오류를 반환합니다.

Image Builder에서의 Amazon SNS 통합

Amazon Simple Notification Service(SNS)는 게시자에서 구독자(생산자 및 소비자라고도 함)로 비동기 메시지를 전송하는 관리형 서비스입니다.

인프라 구성에서 SNS 주제를 지정할 수 있습니다. 이미지를 생성하거나 파이프라인을 실행하면 Image Builder에서 이미지 상태에 대한 자세한 메시지를 이 주제에 게시할 수 있습니다. 이미지 상태가 다음 상태 중 하나에 도달하면 Image Builder에서 메시지를 게시합니다.

- `AVAILABLE`
- `FAILED`

Image Builder의 SNS 메시지 예제는 SNS 메시지 형식(을)를 참조하세요. SNS 주제를 새로 생성하고 싶다면, Amazon Simple Notification Service 개발자 설명서의 Amazon SNS 시작하기 섹션을 참조하세요.

암호화된 SNS 주제

SNS 주제가 암호화된 경우 Image Builder 서비스 역할이 다음 작업을 수행할 수 있도록 AWS KMS key 정책에서 권한을 부여해야 합니다.

- `kms:Decrypt`
- `kms:GenerateDataKey`

> **Note**
>
> SNS 주제가 암호화된 경우 이 주제를 암호화하는 키는 Image Builder 서비스가 실행되는 계정에 있어야 합니다. Image Builder는 다른 계정의 키로 암호화된 SNS 주제에 알림을 보낼 수 없습니다.

KMS 키 정책 추가 예제

다음 예는 KMS 키 정책에 추가하는 추가 섹션을 보여줍니다. Image Builder 이미지를 처음 생성할 때 Image Builder가 사용자 계정에서 생성한 IAM 서비스 연결 역할에는 Amazon 리소스 이름(ARN)을 사용하세요. Image Builder 서비스 연결 역할에 대한 자세한 내용은 Image Builder의 IAM 서비스 연결 역할 사용 섹션을 참조하세요.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::123456789012:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

다음 방법 중 하나를 사용하여 ARN을 가져올 수 있습니다.

AWS Management Console

Image Builder가 계정에서 생성한 서비스 연결 역할의 ARN을에서 가져오려면 다음 단계를 AWS Management Console따릅니다.

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. `ImageBuilder`을(를) 검색한 후 결과에서 다음 역할 이름을 선택합니다. `AWSServiceRoleForImageBuilder` 그러면 역할 세부 정보 페이지가 표시됩니다.

4. ARN을 클립보드에 복사하여 ARN 이름 옆의 아이콘을 선택하세요.

AWS CLI

Image Builder가 계정에서 생성한 서비스 연결 역할의 ARN을에서 가져오려면 다음과 같이 IAM get-role 명령을 AWS CLI사용합니다.

```
aws iam get-role --role-name AWSServiceRoleForImageBuilder
```

일부 샘플 출력:

```
{
    "Role": {
        "Path": "/aws-service-role/imagebuilder.amazonaws.com/",
        "RoleName": "AWSServiceRoleForImageBuilder",
        ...
        "Arn": "arn:aws:iam::123456789012:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder",
        ...
}
```

SNS 메시지 형식

Image Builder가 Amazon SNS 주제에 메시지를 게시하고 나면 해당 주제를 구독하는 다른 서비스에서 메시지 형식을 필터링하여 메시지 형식이 추가 조치 기준을 충족하는지 확인할 수 있습니다. 예를 들어, 성공 메시지는 AWS Systems Manager 매개 변수 저장소를 업데이트하거나 출력 AMI에 대한 외부 규정 준수 테스트 워크플로를 시작하는 작업을 시작할 수 있습니다.

다음 예제는 파이프라인 빌드가 완료될 때 Image Builder에서 게시하는 일반적인 메시지에 대한 JSON 페이로드를 보여 주고 Linux 이미지를 생성합니다.

```
{
  "versionlessArn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image",
  "semver": 1237940039285380274899124227,
  "arn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image/1.0.0/3",
  "name": "example-linux-image",
  "version": "1.0.0",
  "type": "AMI",
```

```
  "buildVersion": 3,
  "state": {
    "status": "AVAILABLE"
  },
  "platform": "Linux",
  "imageRecipe": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:image-recipe/example-linux-
image/1.0.0",
    "name": "amjule-barebones-linux",
    "version": "1.0.0",
    "components": [
      {
        "componentArn": "arn:aws:imagebuilder:us-west-1:123456789012:component/update-
linux/1.0.2/1"
      }
    ],
    "platform": "Linux",
    "parentImage": "arn:aws:imagebuilder:us-west-1:987654321098:image/amazon-linux-2-
x86/2022.6.14/1",
    "blockDeviceMappings": [
      {
        "deviceName": "/dev/xvda",
        "ebs": {
          "encrypted": false,
          "deleteOnTermination": true,
          "volumeSize": 8,
          "volumeType": "gp2"
        }
      }
    ],
    "dateCreated": "Feb 24, 2021 12:31:54 AM",
    "tags": {
      "internalId": "1a234567-8901-2345-bcd6-ef7890123456",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:image-recipe/example-
linux-image/1.0.0"
    },
    "workingDirectory": "/tmp",
    "accountId": "462045008730"
  },
  "sourcePipelineArn": "arn:aws:imagebuilder:us-west-1:123456789012:image-pipeline/
example-linux-pipeline",
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:infrastructure-configuration/
example-linux-infra-config-uswest1",
```

```
    "name": "example-linux-infra-config-uswest1",
    "instanceProfileName": "example-linux-ib-baseline-admin",
    "tags": {
      "internalId": "234abc56-d789-0123-a4e5-6b789d012c34",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:infrastructure-
configuration/example-linux-infra-config-uswest1"
    },
    "logging": {
      "s3Logs": {
        "s3BucketName": "amzn-s3-demo-bucket"
      }
    },
    "keyPair": "example-linux-key-pair-uswest1",
    "terminateInstanceOnFailure": true,
    "snsTopicArn": "arn:aws:sns:us-west-1:123456789012:example-linux-ibnotices-
uswest1",
    "dateCreated": "Feb 24, 2021 12:31:55 AM",
    "accountId": "123456789012"
  },
  "imageTestsConfigurationDocument": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 720
  },
  "distributionConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:distribution-configuration/
example-linux-distribution",
    "name": "example-linux-distribution",
    "dateCreated": "Feb 24, 2021 12:31:56 AM",
    "distributions": [
      {
        "region": "us-west-1",
        "amiDistributionConfiguration": {}
      }
    ],
    "tags": {
      "internalId": "345abc67-8910-12d3-4ef5-67a8b90c12de",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:distribution-
configuration/example-linux-distribution"
    },
    "accountId": "123456789012"
  },
  "dateCreated": "Jul 28, 2022 1:13:45 AM",
  "outputResources": {
    "amis": [
```

```
      {
        "region": "us-west-1",
        "image": "ami-01a23bc4def5a6789",
        "name": "example-linux-image 2022-07-28T01-14-17.416Z",
        "accountId": "123456789012"
      }
    ]
  },
  "buildExecutionId": "ab0cd12e-34fa-5678-b901-2c3456d789e0",
  "testExecutionId": "6a7b8901-cdef-234a-56b7-8cd89ef01234",
  "distributionJobId": "1f234567-8abc-9d0e-1234-fa56b7c890de",
  "integrationJobId": "432109b8-afe7-6dc5-4321-0ba98f7654e3",
  "accountId": "123456789012",
  "osVersion": "Amazon Linux 2",
  "enhancedImageMetadataEnabled": true,
  "buildType": "USER_INITIATED",
  "tags": {
    "internalId": "901e234f-a567-89bc-0123-d4e567f89a01",
    "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image/1.0.0/3"
  }
}
```

다음 예제는 Linux 이미지에 대한 파이프라인 빌드 실패에 대해 Image Builder가 게시하는 일반적인 메시지에 대한 JSON 페이로드를 보여줍니다.

```
{
  "versionlessArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-
image",
  "semver": 1237940039285380274899124231,
  "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-image/1.0.0/7",
  "name": "My Example Image",
  "version": "1.0.0",
  "type": "AMI",
  "buildVersion": 7,
  "state": {
    "status": "FAILED",
    "reason": "Image Failure reason."
  },
  "platform": "Linux",
  "imageRecipe": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-
image/1.0.0",
```

```
    "name": "My Example Image",
    "version": "1.0.0",
    "description": "Testing Image recipe",
    "components": [
      {
        "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-
example-image-component/1.0.0/1"
      }
    ],
    "platform": "Linux",
    "parentImage": "ami-0cd12345db678d90f",
    "dateCreated": "Jun 21, 2022 11:36:14 PM",
    "tags": {
      "internalId": "1a234567-8901-2345-bcd6-ef7890123456",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-
example-image/1.0.0"
    },
    "accountId": "123456789012"
  },
  "sourcePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-
example-image-pipeline",
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/
my-example-infra-config",
    "name": "SNS topic Infra config",
    "description": "An example that will retain instances of failed builds",
    "instanceTypes": [
      "t2.micro"
    ],
    "instanceProfileName": "EC2InstanceProfileForImageBuilder",
    "tags": {
      "internalId": "234abc56-d789-0123-a4e5-6b789d012c34",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-
configuration/my-example-infra-config"
    },
    "terminateInstanceOnFailure": true,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:example-pipeline-notification-
topic",
    "dateCreated": "Jul 5, 2022 7:31:53 PM",
    "accountId": "123456789012"
  },
  "imageTestsConfigurationDocument": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 720
```

```
  },
  "distributionConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-
example-distribution-config",
    "name": "New distribution config",
    "dateCreated": "Dec 3, 2021 9:24:22 PM",
    "distributions": [
      {
        "region": "us-west-2",
        "amiDistributionConfiguration": {},
        "fastLaunchConfigurations": [
          {
            "enabled": true,
            "snapshotConfiguration": {
              "targetResourceCount": 2
            },
            "maxParallelLaunches": 2,
            "launchTemplate": {
              "launchTemplateId": "lt-01234567890"
            },
            "accountId": "123456789012"
          }
        ]
      }
    ],
    "tags": {
      "internalId": "1fecd23a-4f56-7f89-01e2-345678abbe90",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-
configuration/my-example-distribution-config"
    },
    "accountId": "123456789012"
  },
  "dateCreated": "Jul 5, 2022 7:40:15 PM",
  "outputResources": {
    "amis": []
  },
  "accountId": "123456789012",
  "enhancedImageMetadataEnabled": true,
  "buildType": "SCHEDULED",
  "tags": {
    "internalId": "456c78b9-0e12-3f45-afb6-7e89b0f1a23b",
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-
image/1.0.0/7"
  }
```

```
}
```

Image Builder 이미지를 위한 규정 준수 제품

보안 표준이 계속 발전함에 따라 규정 준수를 유지하고 조직을 사이버 위협으로부터 보호하는 것이 어려울 수 있습니다. 사용자 지정 이미지가 규정을 준수하도록 하고 게시자가 새 버전을 릴리스할 때 자동 업데이트를 계속 진행하기 위해 Image Builder는 AWS Marketplace 규정 준수 제품 및 Image Builder 구성 요소와 통합됩니다.

Image Builder는 다음 규정 준수 제품과 통합됩니다.

- Center for Internet Security(CIS) 벤치마크 하드닝

  CIS 하드닝 이미지 및 관련 CIS 하드닝 구성 요소를 사용하여 최신 CIS 벤치마크 레벨 1 지침을 준수하는 사용자 정의 이미지를 구축할 수 있습니다. CIS 강화 이미지는에서 사용할 수 있습니다 AWS Marketplace. CIS 강화 이미지 및 강화 구성 요소를 설정하고 사용하는 방법에 대한 자세한 내용은 CIS 웹 사이트 지원 포털의 빠른 시작 가이드를 참조하세요.

  > **Note**
  >
  > CIS 하드닝 이미지를 구독하면 구성에 CIS 벤치마크 레벨 1 지침을 적용하는 스크립트를 실행하는 관련 빌드 구성 요소에도 액세스할 수 있습니다. 자세한 내용은 CIS 하드닝 구성 요소 단원을 참조하십시오.

- 보안 기술 구현 가이드(STIG)

  STIG 규정 준수를 위해를 사용하면 Image Builder 레시피에서 Amazon 관리형 AWS Task Orchestrator and Executor (AWSTOE) STIG 구성 요소를 사용할 수 있습니다. STIG 구성 요소는 빌드 인스턴스의 구성 오류를 스캔하고 수정 스크립트를 실행하여 발견된 문제를 수정합니다. Image Builder로 빌드한 이미지에 대해서는 STIG 규정 준수를 보장할 수 없습니다. 조직의 규정 준수 팀과 협력하여 최종 이미지가 규정을 준수하는지 확인해야 합니다. Image Builder 레시피에 사용할 수 있는 AWSTOE STIG 구성 요소의 전체 목록은 섹션을 참조하세요Image Builder용 Amazon 관리형 STIG 하드닝 구성 요소.

Image Builder에서 이벤트 및 로그 모니터링

EC2 Image Builder 파이프라인의 신뢰성, 가용성 및 성능을 유지하려면 이벤트와 로그를 모니터링하는 것이 중요합니다. 이벤트와 로그를 통해 API 직접 호출 실패 시 전체적인 상황을 파악하고 세부 정보를 자세히 파악할 수 있습니다. Image Builder는 이벤트가 구성한 기준과 일치할 경우 알림을 보내고 자동 응답을 시작할 수 있는 서비스와 통합됩니다.

다음 항목에서는 Image Builder와 통합되는 서비스를 통해 사용할 수 있는 모니터링 기술에 대해 설명합니다.

이벤트와 로그 모니터링

- CloudTrail을 사용하여 Image Builder API 직접 호출 로깅
- Amazon CloudWatch Logs를 사용하여 Image Builder 로그 모니터링

CloudTrail을 사용하여 Image Builder API 직접 호출 로깅

EC2 Image Builder는 Image Builder에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Image Builder에 대한 모든 API 직접 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Image Builder 콘솔로부터의 호출과 Image Builder API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 Image Builder에 대한 이벤트를 포함하여 CloudTrail 이벤트를 Amazon S3 버킷에 지속적으로 전달할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Image Builder에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Image Builder 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. Image Builder에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. 에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다 AWS 계정. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Image Builder에 대한 이벤트를 AWS 계정포함하여에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전

에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

CloudTrail은 모든 Image Builder 작업을 로깅하고 이 작업은 EC2 Image Builder API 참조에 문서화됩니다. 예를 들어 `CreateImagePipeline`, `UpdateInfrastructureConfiguration` 및 `StartImagePipelineExecution` 작업을 직접적으로 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부입니다.

자세한 내용은 CloudTrail userIdentity 요소를 참조하십시오.

Amazon CloudWatch Logs를 사용하여 Image Builder 로그 모니터링

CloudWatch Logs 지원은 기본적으로 활성화되어 있습니다. 로그는 빌드 프로세스 중에 인스턴스에 유지되고 CloudWatch Logs로 스트리밍됩니다. 인스턴스 로그는 이미지를 생성하기 전에 인스턴스에서 제거됩니다.

빌드 로그는 Image Builder CloudWatch Logs 그룹 및 스트림을 따라 스트리밍됩니다.

LogGroup:

```
/aws/imagebuilder/ImageName
```

LogStream(x.x.x/x):

```
ImageVersion/ImageBuildVersion
```

인스턴스 프로파일과 관련된 다음 권한을 제거하여 CloudWatch Logs 스트리밍을 거부할 수 있습니다.

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "logs:CreateLogStream",
            "logs:CreateLogGroup",
            "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/imagebuilder/*"
    }
]
```

고급 문제 해결을 위해 AWS Systems Manager 명령 실행을 사용하여 사전 정의된 명령과 스크립트를 실행할 수 있습니다. 자세한 내용은 Image Builder 문제 해결 단원을 참조하십시오.

Image Builder의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 클라우드 AWS 서비스 에서 AWS 실행되는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. EC2 Image Builder에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 AWS 서비스 규정 준수 프로그램의 범위에 속하는를 참조합니다.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Image Builder 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 Image Builder를 구성하는 방법을 보여줍니다. 또한 Image Builder 리소스를 모니터링하고 보호하는 데 도움이 AWS 서비스 되는 다른를 사용하는 방법을 알아봅니다.

주제

- Image Builder의 데이터 보호 및 AWS 공동 책임 모델
- Image Builder의 자격 증명 및 액세스 관리 통합
- Image Builder의 규정 준수 검증 리소스
- Image Builder의 데이터 중복성 및 복원력
- Image Builder의 인프라 보안
- Image Builder 이미지에 대한 패치 관리
- Image Builder의 보안 모범 사례

Image Builder의 데이터 보호 및 AWS 공동 책임 모델

AWS 공동 책임 모델 EC2 Image Builder의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스

팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 Image Builder 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

Image Builder의 암호화 및 키 관리

Image Builder는 서비스 소유의 KMS 키를 사용하여 전송 중 데이터와 저장된 데이터를 기본적으로 암호화합니다. 단, 다음과 같은 경우는 예외입니다.

- 사용자 지정 구성 요소-Image Builder는 기본 KMS 키 또는 서비스 소유 KMS 키를 사용하여 사용자 지정 구성 요소를 암호화합니다.

- 이미지 워크플로-Image Builder는 워크플로 생성 중에 키를 지정하는 경우 고객 관리형 키를 사용하여 이미지 워크플로를 암호화할 수 있습니다. Image Builder는 키로 암호화 및 복호화를 처리하여 이미지에 대해 구성한 워크플로를 실행합니다.

를 통해 자체 키를 관리할 수 있습니다 AWS KMS. 하지만 Image Builder가 소유한 Image Builder KMS 키를 관리할 권한은 없습니다. 를 사용하여 KMS 키를 관리하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 시작하기를 AWS Key Management Service참조하세요.

암호화 컨텍스트

암호화된 데이터에 대한 무결성 및 신뢰성 검사를 추가로 제공하기 위해 데이터를 암호화할 때 암호화 컨텍스트를 포함할수 있습니다. 리소스가 암호화 컨텍스트로 암호화되면 AWS KMS 암호화 방식으로 컨텍스트를 사이퍼텍스트에 바인딩합니다. 요청자가 컨텍스트에 대해 대소문자를 구분하여 정확히 일치하는 항목을 제공하는 경우에만 리소스를 해독할 수 있습니다.

이 섹션의 정책 예제는 Image Builder 워크플로 리소스의 Amazon 리소스 이름(ARN)과 유사한 암호화 컨텍스트를 사용합니다.

고객 관리형 키로 이미지 워크플로 암호화

보호 계층을 추가하려면 Image Builder 워크플로 리소스를 자체 고객 관리형 키로 암호화할 수 있습니다. 고객 관리형 키를 사용하여 생성한 Image Builder 워크플로를 암호화하는 경우 Image Builder가 워크플로 리소스를 암호화 및 복호화할 때 Image Builder가 키를 사용하도록 키 정책에서 액세스 권한을 부여해야 합니다. 언제든지 액세스 권한을 취소할 수 있습니다. 그러나 키에 대한 액세스 권한을 취소할 경우 Image Builder는 이미 암호화된 워크플로에 액세스할 수 없습니다.

Image Builder에 고객 관리형 키를 사용할 수 있는 액세스 권한을 부여하는 프로세스는 다음과 같이 두 단계로 되어 있습니다.

1단계: Image Builder 워크플로에 키 정책 권한 추가

워크플로를 만들거나 사용할 때 Image Builder에서 워크플로 리소스를 암호화하고 해독할 수 있도록 하려면 KMS 키 정책에 권한을 지정해야 합니다.

이 예제의 키 정책은 Image Builder 파이프라인이 생성 프로세스 중에 워크플로 리소스를 암호화하고 워크플로 리소스를 해독하여 사용할 수 있는 액세스 권한을 부여합니다. 또한 이 정책은 주요 관리자에게 액세스 권한을 부여합니다. 암호화 컨텍스트 및 리소스 사양은 와일드카드를 사용하여 워크플로 리소스가 있는 모든 리전을 포함합니다.

이미지 워크플로를 사용하기 위한 전제 조건으로 Image Builder에 워크플로 작업을 실행할 권한을 부여하는 IAM 워크플로 실행 역할을 생성했습니다. 다음 키 정책 예제에 표시된 첫 번째 명령문의 위탁자는 IAM 워크플로 실행 역할을 지정해야 합니다.

고객 관리형 키에 대한 자세한 내용은AWS Key Management Service 개발자 안내서의 고객 관리형 키 액세스 관리를 참조하세요.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Allow access to build images with encrypted workflow",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/YourImageBuilderExecutionRole"
   },
   "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
   ],
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "kms:EncryptionContext:aws:imagebuilder:arn":
 "arn:aws:imagebuilder:*:111122223333:workflow/*"
    }
   }
  },
  {
   "Sid": "Allow access for key administrators",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:root"
   },
   "Action": [
    "kms:*"
   ],
   "Resource": "arn:aws:kms:*:111122223333:key/"
  }
 ]
}
```

2단계: 워크플로 실행 역할에 대한 키 액세스 권한 부여

Image Builder가 워크플로를 실행하기 위해 맡는 IAM 역할에는 고객 관리형 키를 사용할 권한이 필요합니다. 키에 액세스하지 않으면 Image Builder에서 해당 키로 워크플로 리소스를 암호화 또는 복호화할 수 없습니다.

워크플로 실행 역할의 정책을 편집하여 다음 정책 설명을 추가합니다.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Allow access to the workflow key",
   "Effect": "Allow",
   "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
   ],
   "Resource": "arn:aws:kms:us-west-2:111122223333:key/key_ID",
   "Condition": {
    "StringLike": {
     "kms:EncryptionContext:aws:imagebuilder:arn":
 "arn:aws:imagebuilder:*:111122223333:workflow/*"
    }
   }
  }
 ]
}
```

AWS CloudTrail 이미지 워크플로에 대한 이벤트

다음 예제에서는 고객 관리형 키로 저장된 이미지 워크플로를 암호화하고 해독하기 위한 일반적인 AWS CloudTrail 항목을 보여줍니다.

예: GenerateDataKey

이 예제는 Image Builder가 Image Builder API 작업에서 API CreateWorkflow 작업을 호출할 AWS KMS GenerateDataKey 때 CloudTrail 이벤트가 어떻게 보일 수 있는지 보여줍니다. Image Builder는 워크플로 리소스를 생성하기 전에 새 워크플로를 암호화해야 합니다.

```
{
 "eventVersion": "1.08",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "PRINCIPALID1234567890:workflow-role-name",
```

```
   "arn": "arn:aws:sts::111122223333:assumed-role/Admin/workflow-role-name",
   "accountId": "111122223333",
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "sessionContext": {
    "sessionIssuer": {
     "type": "Role",
     "principalId": "PRINCIPALID1234567890",
     "arn": "arn:aws:iam::111122223333:role/Admin",
     "accountId": "111122223333",
     "userName": "Admin"
    },
    "webIdFederationData": {},
    "attributes": {
     "creationDate": "2023-11-21T20:29:31Z",
     "mfaAuthenticated": "false"
    }
   },
   "invokedBy": "imagebuilder.amazonaws.com"
  },
  "eventTime": "2023-11-21T20:31:03Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "imagebuilder.amazonaws.com",
  "userAgent": "imagebuilder.amazonaws.com",
  "requestParameters": {
   "encryptionContext": {
    "aws:imagebuilder:arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
sample-encrypted-workflow/1.0.0/*",
    "aws-crypto-public-key": "key value"
   },
   "keyId": "arn:aws:kms:us-west-2:111122223333:alias/ExampleKMSKey",
   "numberOfBytes": 32
  },
  "responseElements": null,
  "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
  "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
  "readOnly": true,
  "resources": [
   {
    "accountId": "111122223333",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLEzzzzz"
   }
```

```
 ],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "eventCategory": "Management"
}
```

예: Decrypt

이 예제는 Image Builder가 Image Builder API 작업에서 API GetWorkflow 작업을 호출할 AWS KMS Decrypt 때 CloudTrail 이벤트가 어떻게 보일 수 있는지 보여줍니다. Image Builder 파이프라인은 워크플로 리소스를 사용하기 전에 먼저 복호화해야 합니다.

```
{
 "eventVersion": "1.08",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "PRINCIPALID1234567890:workflow-role-name",
  "arn": "arn:aws:sts::111122223333:assumed-role/Admin/workflow-role-name",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "PRINCIPALID1234567890",
    "arn": "arn:aws:iam::111122223333:role/Admin",
    "accountId": "111122223333",
    "userName": "Admin"
   },
   "webIdFederationData": {},
   "attributes": {
    "creationDate": "2023-11-21T20:29:31Z",
    "mfaAuthenticated": "false"
   }
  },
  "invokedBy": "imagebuilder.amazonaws.com"
 },
 "eventTime": "2023-11-21T20:34:25Z",
 "eventSource": "kms.amazonaws.com",
 "eventName": "Decrypt",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "imagebuilder.amazonaws.com",
 "userAgent": "imagebuilder.amazonaws.com",
```

```
 "requestParameters": {
  "keyId": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLEzzzzz",
  "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
  "encryptionContext": {
   "aws:imagebuilder:arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
sample-encrypted-workflow/1.0.0/*",
   "aws-crypto-public-key": "ABC123def4567890abc12345678/90dE/F123abcDEF+4567890abc123D
+ef1=="
  }
 },
 "responseElements": null,
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
 "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
 "readOnly": true,
 "resources": [
  {
   "accountId": "111122223333",
   "type": "AWS::KMS::Key",
   "ARN": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLEzzzzz"
  }
 ],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "eventCategory": "Management"
}
```


Image Builder의 데이터 스토리지

Image Builder는 서비스에 어떤 로그도 저장하지 않습니다. 모든 로그는 이미지를 빌드하는 데 사용되는 Amazon EC2 인스턴스 또는 Systems Manager 자동화 로그에 저장됩니다.

Image Builder의 인터네트워크 트래픽 프라이버시

Image Builder와 온프레미스 위치 간, AWS 리전 내 AZs 간, HTTPS를 통해 AWS 리전 간 연결이 보호됩니다. 계정 간에는 직접 연결이 없습니다.

Image Builder의 자격 증명 및 액세스 관리 통합

주제

- 대상
- ID를 통한 인증
- Image Builder가 IAM 정책 및 역할을 사용하는 방법
- Image Builder에서 S3 버킷 다운로드 액세스의 데이터 경계 관리
- Image Builder 자격 증명 기반 정책
- Image Builder 리소스 기반 정책
- EC2 Image Builder에 대한 AWS 관리형 정책 사용
- Image Builder의 IAM 서비스 연결 역할 사용
- Image Builder의 IAM 문제 해결

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 Image Builder에서 수행하는 작업에 따라 다릅니다.

서비스 사용자- Image Builder 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명 및 권한을 관리자가 제공합니다. 더 많은 Image Builder 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. Image Builder의 기능에 액세스할 수 없는 경우 Image Builder의 IAM 문제 해결(을)를 참조합니다.

서비스 관리자 - 회사에서 Image Builder 리소스를 책임지고 있는 경우 Image Builder에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 사용자가 액세스해야 하는 Image Builder 기능과 리소스를 결정하는 것은 귀하의 임무입니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해합니다. 회사가 Image Builder에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Image Builder가 IAM 정책 및 역할을 사용하는 방법을 참조합니다.

IAM 관리자 - IAM 관리자라면 Image Builder에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 Image Builder 자격 증명 기반 정책 예제를 보려면 Image Builder 자격 증명 기반 정책(을)를 참조합니다.

ID를 통한 인증

의 사용자 및 프로세스에 인증을 제공하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 자격 증명을 AWS 계정참조하세요.

Image Builder가 IAM 정책 및 역할을 사용하는 방법

IAM을 사용하여 Image Builder에 대한 액세스를 관리하기 전에 Image Builder와 함께 사용할 수 있는 IAM 기능을 알아봅니다.

Image Builder 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

Image Builder를 위한 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. ID 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로 위탁자를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용자 설명서의 IAM JSON 정책 요소 참조(IAM JSON policy elements reference)를 참조합니다.

Image Builder의 자격 증명 기반 정책 예

Image Builder 자격 증명 기반 정책의 예를 보려면 Image Builder 자격 증명 기반 정책(을)를 참조합니다.

Image Builder 내 리소스 기반 정책

리소스 기반 정책 지원: 예

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 위탁자를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는이 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 위탁자로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 다른 경우 신뢰할 수 AWS

계정있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 위탁자에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

Image Builder의 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 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 포함하세요.

Image Builder 작업 목록을 보려면 서비스 권한 부여 참조의 EC2 Image Builder에서 정의한 작업을 참조합니다.

Image Builder의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
imagebuilder
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
 "imagebuilder:action1",
 "imagebuilder:action2"
 ]
```

Image Builder 자격 증명 기반 정책의 예를 보려면 Image Builder 자격 증명 기반 정책(을)를 참조합니다.

Image Builder용 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` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Image Builder 리소스 유형 및 해당 ARN 목록을 보려면 서비스 권한 부여 참조의 EC2 Image Builder가 정의한 리소스를 참조합니다. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 EC2 Image Builder가 정의한 작업을 참조합니다.

Image Builder 자격 증명 기반 정책의 예를 보려면 Image Builder 자격 증명 기반 정책(을)를 참조합니다.

Image Builder의 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` 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 `OR` 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Image Builder 조건 키 목록을 보려면 서비스 인증 참조의 EC2 Image Builder의 조건 키를 참조합니다. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 EC2 Image Builder가 정의한 작업을 참조합니다.

Image Builder 자격 증명 기반 정책의 예를 보려면 Image Builder 자격 증명 기반 정책(을)를 참조합니다.

Image Builder의 ACL

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Image Builder의 ABAC

ABAC 지원(정책의 태그): 부분적

속성 기반 액세스 제어(ABAC)는 속성에 근거하여 권한을 정의하는 권한 부여 전략입니다. 에서는 AWS이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS 리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 위탁자의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 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.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Image Builder에서 임시 보안 인증 사용

임시 자격 증명 지원: 예

일부 AWS 서비스 는 임시 자격 증명을 사용하여 로그인할 때 작동하지 않습니다. 임시 자격 증명으로 AWS 서비스 작업하는를 포함한 추가 정보는 AWS 서비스 IAM 사용 설명서의 IAM으로 작업하는 섹션을 참조하세요.

사용자 이름 및 암호를 제외한 방법을 AWS Management Console 사용하여에 로그인하는 경우 임시 자격 증명을 사용합니다. 예를 들어 회사의 SSO(Single Sign-On) 링크를 AWS 사용하여에 액세스하면 해당 프로세스가 임시 자격 증명을 자동으로 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 자격 증명을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 사용자에서 IAM 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다. 그런 다음 이러한 임시 자격 증명을 사용하여 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 access AWS. AWS recommends에 액세스할 수 있습니다. 자세한 정보는 IAM의 임시 보안 인증을 참조합니다.

Image Builder의 서비스 간 위탁자 권한

전달 액세스 세션(FAS) 지원: 예

IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하기 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Image Builder의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 Create a role to delegate permissions to an AWS 서비스를 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Image Builder 기능이 중단될 수 있습니다. Image Builder에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Image Builder의 서비스 연결 역할

서비스 링크 역할 지원: 예

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Image Builder 서비스 연결 역할에 대한 자세한 내용은 Image Builder의 IAM 서비스 연결 역할 사용(을)를 참조합니다.

Image Builder 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스 및 작업이 허용되거나 거부되는 조건도 지정할 수 있습니다. Image Builder는 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대한 자세한 내용은 IAM 사용자 설명서의 Amazon EC2 Image Builder의 작업, 리소스 및 조건 키를 참조합니다.

작업

Image Builder의 정책 작업은 작업 전에 `imagebuilder:` 접두사를 사용합니다. 정책 문에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. Image Builder는 이 서비스로 수행할 수 있는 태스크를 설명하는 고유한 작업 세트를 정의합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
 "imagebuilder:action1",
 "imagebuilder:action2"
]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `List`라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "imagebuilder:List*"
```

Image Builder 작업 목록을 보려면 IAM 사용자 설명서의 AWS 서비스를 위한 작업, 리소스 및 조건 키를 참조합니다.

정책을 사용하여 액세스 관리

정책을 생성하고 IAM 자격 증명 또는 AWS 리소스 AWS 에 연결하여에서 액세스를 관리하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 정책 및 권한을 참조하세요.

인스턴스 프로파일에 연결하는 IAM 역할에는 이미지에 포함된 빌드 및 테스트 구성 요소를 실행할 수 있는 권한이 있어야 합니다. 다음 IAM 역할 정책은 인스턴스 프로파일과 관련된 IAM 역할에 연결되어야 합니다.

- EC2InstanceProfileForImageBuilder
- EC2InstanceProfileForImageBuilderECRContainerBuilds
- AmazonSSMManagedInstanceCore

리소스

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

ARN은 리소스를 식별하고 이름이 고유한지 확인하는 데 도움이 되는 여러 노드로 구성됩니다. 이름의 마지막 노드에는 리소스 유형, 이름 및 ID의 형식 지정에 몇 가지 변형이 포함됩니다. Image Builder는 리소스를 생성할 때 다음 형식을 사용합니다.

`arn:aws:imagebuilder:`*`region`*`:`*`owner`*`:`*`resource-type/resource-name/version/build-version`*

> **Note**
>
> 빌드 버전이 리소스 ARN에 항상 포함되는 것은 아닙니다. 그러나 GetComponent와 같은 일부 API 작업은 검색할 리소스를 고유하게 식별하려면 빌드 버전이 필요합니다.

Image Builder가 기본 이미지 또는 구성 요소와 같이 레시피에서 사용하는 리소스의 경우 소유자 노드는 다음 중 하나일 수 있습니다.

- 리소스 소유자의 계정 번호
- Amazon 관리형 리소스의 경우: `aws`
- AWS Marketplace 리소스의 경우: `aws-marketplace`

다음 예제에서는 Linux에 Amazon CloudWatch 에이전트를 설치하기 위한 관리형 구성 요소의 ARN을 보여줍니다.

```
arn:aws:imagebuilder:us-east-1:aws:component/amazon-cloudwatch-agent-linux/1.0.1/1
```

이 예제는 AWS Marketplace다음에서 가상의 관리형 구성 요소에 대한 ARN을 보여줍니다.

```
arn:aws:imagebuilder:us-east-1:aws-marketplace:component/example-linux-software-component/1.0.1
```

소유권 필터 사용을 포함하여 구성 요소 목록을 가져오는 방법에 대한 자세한 내용은 섹션을 참조하세요Image Builder 구성 요소 나열.

예제 ARN

다음은 IAM 정책에서 지정할 수 있는 리소스 ARNs의 몇 가지 예입니다.

- 인스턴스 ARN

```
"Resource": "arn:aws:imagebuilder:us-east-1:111122223333:instance/i-1234567890abcdef0"
```

- 지정된 계정의 모든 인스턴스를 지정하는 와일드카드(*) 예제

```
"Resource": "arn:aws:imagebuilder:us-east-1:111122223333:instance/*"
```

- 관리형 이미지 워크플로의 모든 버전을 지정하는 와일드카드(*) 예제

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:workflow/build/build-image/*"
```

- 관리형 이미지 ARN

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-arm64/2024.12.17/1"
```

- 관리형 이미지의 모든 버전을 지정하는 와일드카드(*) 예제

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-arm64/x.x.x"
```

다양한 EC2 Image Builder API 작업에는 여러 리소스가 관여합니다. 단일 문에서 여러 리소스를 지정하려면 ARN을 쉼표로 구분합니다.

```
"Resource": [
   "resource1",
   "resource2"
]
```

조건 키

Image Builder는 서비스별 조건 키를 제공하고 일부 글로벌 조건 키를 사용하여 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요. 다음과 같은 서비스별 조건 키가 제공됩니다.

imagebuilder:CreatedResourceTagKeys

문자열 연산자를 사용합니다.

요청에 태그 키가 있는지 여부를 기준으로 액세스를 필터링하는 이 키를 사용합니다. 이를 통해 Image Builder가 생성하는 리소스를 관리할 수 있습니다.

가용성 — 이 키는 `CreateInfrastrucutreConfiguration` 및 `UpdateInfrastructureConfiguration` API에서만 사용할 수 있습니다.

imagebuilder:CreatedResourceTag/<key>

문자열 연산자를 사용합니다.

Image Builder가 생성한 리소스에 연결된 태그 키-값 페어를 기준으로 액세스를 필터링하는 데 이 키를 사용합니다. 이렇게 하면 정의된 태그를 통해 Image Builder 리소스를 관리할 수 있습니다.

가용성 — 이 키는 `CreateInfrastrucutreConfiguration` 및 `UpdateInfrastructureConfiguration` API에서만 사용할 수 있습니다.

imagebuilder:LifecyclePolicyResourceType

문자열 연산자를 사용합니다.

이 키를 사용하여 요청에 지정된 수명 주기 리소스 유형을 기준으로 액세스를 필터링합니다.

이 키의 값은 AMI_IMAGE 또는 일 수 있습니다CONTAINER_IMAGE.

가용성 — 이 키는 CreateLifecyclePolicy 및 UpdateLifecyclePolicy API에서만 사용할 수 있습니다.

imagebuilder:Ec2MetadataHttpTokens

문자열 연산자를 사용합니다.

요청에 지정된 EC2 인스턴스 메타데이터 HTTP 토큰 요구 사항을 기준으로 액세스를 필터링하는 이 키를 사용합니다.

이 키의 값은 optional 또는 required일 수 있습니다.

가용성 — 이 키는 CreateInfrastrucutreConfiguration 및 UpdateInfrastructureConfiguration API에서만 사용할 수 있습니다.

imagebuilder:StatusTopicArn

문자열 연산자를 사용합니다.

터미널 상태 알림을 게시하는 요청에서 SNS Topic ARN을 기준으로 액세스를 필터링하는 이 키를 사용합니다.

가용성 — 이 키는 CreateInfrastrucutreConfiguration 및 UpdateInfrastructureConfiguration API에서만 사용할 수 있습니다.

예시

Image Builder 자격 증명 기반 정책의 예를 보려면 Image Builder 자격 증명 기반 정책(을)를 참조합니다.

Image Builder 리소스 기반 정책

리소스 기반 정책은 지정된 위탁자가 Image Builder 리소스에 대해 수행할 수 있는 작업 및 관련 조건을 지정합니다. Image Builder는 구성 요소, 이미지 및 이미지 레시피에 대한 리소스 기반 권한 정책을 지원합니다. 리소스 기반 정책을 사용하여 리소스별로 다른 계정에 사용 권한을 부여할 수 있습니다.

리소스 기반 정책을 사용하여 AWS 서비스가 구성 요소, 이미지 및 이미지 레시피에 액세스하도록 허용할 수도 있습니다.

구성 요소, 이미지 또는 이미지 레시피에 리소스 기반 정책을 연결하는 방법에 대한 자세한 내용은 와 Image Builder 리소스 공유 AWS RAM(을)를 참조합니다.

> **Note**
>
> Image Builder를 사용하여 리소스 정책을 업데이트하면 업데이트가 RAM 콘솔에 표시됩니다.

Image Builder 태그 기반 권한 부여

태그를 Image Builder 리소스에 연결하거나 Image Builder 요청을 통해 태그를 에 전달할 수 있습니다. 태그에 근거하여 액세스를 제어하려면 `imagebuilder:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다. Image Builder 리소스 태깅에 대한 자세한 내용은 에서 리소스에 태그 지정 AWS CLI(을)를 참조합니다.

Image Builder IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내의 엔터티입니다.

Image Builder에서 임시 보안 인증 사용

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken과 같은 AWS STS API 작업을 호출하여 임시 보안 자격 증명을 얻습니다.

서비스 연결 역할

서비스 연결 역할을 사용하면 AWS 서비스 가 다른 서비스의 리소스에 액세스하여 사용자를 대신하여 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. 관리자 엑세스를 가진 사용자는 서비스 연결 역할에 대한 권한을 볼 수는 있지만 편집할 수는 없습니다.

Image Builder는 서비스 연결 역할을 지원합니다. Image Builder 서비스 연결 역할을 생성하거나 관리하는 방법에 대한 자세한 내용은 Image Builder의 IAM 서비스 연결 역할 사용(을)를 참조합니다.

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 서비스 역할을 수임할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역

할은 IAM 계정에 나타나고, 해당 계정이 소유합니다. 즉, 관리자 엑세스를 가진 사용자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

Image Builder에서 S3 버킷 다운로드 액세스의 데이터 경계 관리

EC2 Image Builder는 계정에서 Image Builder 워크로드를 실행하는 데 필요한 다운로드 가능한 리소스가 포함된 AWS 서비스 소유 S3 버킷의 두 가지 클래스를 유지 관리합니다. 데이터 경계를 사용하여 환경에서 Amazon S3에 대한 액세스를 제어하는 경우 이러한 버킷에 대한 액세스를 명시적으로 허용해야 할 수 있습니다. Amazon S3에 대한 액세스를 제어하는 방법에 따라 버킷 ARN 또는 버킷 URL을 사용하여 이러한 버킷을 허용할 수 있습니다.

구성 요소 관리 부트스트래핑 스크립트(필수)

이 S3 버킷에는 이미지를 생성하는 데 사용되는 EC2 인스턴스에서 AWSTOE 애플리케이션을 설정하는 부트스트래핑 스크립트가 포함되어 있습니다. Image Builder는 새 이미지에 대한 빌드 및 테스트를 지원하기 위해 스크립트를 다운로드할 수 있는 액세스 권한이 필요합니다.

- S3 버킷 ARN: `arn:<AWS partition>:s3:::ec2imagebuilder-managed-resources-<AWS Region>-prod`
- S3 버킷 URL: `https://ec2imagebuilder-managed-resources-<AWS Region>.s3.<AWS Region>.<AWS partition-specific domain name>`

관리형 구성 요소

이 S3 버킷에는 Amazon 관리형 구성 요소에 대한 패키지 페이로드가 포함되어 있습니다. Image Builder를 사용하려면 레시피에 구성된 관리형 구성 요소를 다운로드할 수 있는 액세스 권한이 필요합니다.

- S3 버킷 ARN: `arn:<AWS partition>:s3:::ec2imagebuilder-toe-<AWS Region>-prod`
- S3 버킷 URL: `https://ec2imagebuilder-toe-<AWS Region>.s3.<AWS Region>.<AWS partition-specific domain name>`

Image Builder 자격 증명 기반 정책

주제

- 자격 증명 기반 정책 모범 사례
- Image Builder 콘솔 사용

자격 증명 기반 정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Image Builder 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

Image Builder 콘솔 사용

EC2 Image Builder 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은 AWS 계정에서 Image Builder 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용합니다. 최소 필수 권한보다 더 제한적인 보안 인증 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(IAM 사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

IAM 엔터티가 Image Builder 콘솔을 사용할 수 있도록 하려면 다음 AWS 관리형 정책 중 하나를 연결해야 합니다.

- AWSImageBuilderReadOnlyAccess 정책
- AWSImageBuilderFullAccess 정책

Image Builder 관리형 정책에 대한 자세한 내용은 EC2 Image Builder에 대한 AWS 관리형 정책 사용(을)를 참조합니다.

> **Important**
>
> Image Builder 서비스 연결 역할을 생성하려면 AWSImageBuilderFullAccess 정책이 필요합니다. 이 정책을 IAM 엔티티에 연결할 때는 다음과 같은 사용자 지정 정책도 연결하고 리소스 이름에 `imagebuilder`(이)가 없는 사용하려는 리소스를 포함해야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "sns topic arn"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetInstanceProfile"
            ],
            "Resource": "instance profile role arn"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "instance profile role arn",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
```

```
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "bucket arn"
        }
    ]
}
```

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

Image Builder 리소스 기반 정책

구성 요소를 생성하는 방법에 대한 자세한 내용은 구성 요소를 사용하여 Image Builder 이미지 사용자 지정(을)를 참조합니다.

Image Builder 구성 요소 액세스를 특정 IP 주소로 제한

다음 예에서는 구성 요소에서 모든 Image Builder 작업을 수행할 수 있는 권한을 모든 사용자에게 부여합니다. 하지만 조건에 지정된 IP 주소 범위에서만 요청을 허용해야 합니다.

이 문의 조건은 허용되는 IPv4(인터넷 프로토콜 버전 4) IP 주소의 54.240.143.* 범위를 식별하며 단, 한 가지 예외는 54.240.143.188입니다.

`Condition` 블록은 `IpAddress` 및 `NotIpAddress` 조건과 AWS전체 `aws:SourceIp` 조건 키인 조건 키를 사용합니다. 이러한 조건 키에 대한 자세한 내용은 정책의 조건 지정을 참조합니다. `aws:sourceIp` IPv4 값은 표준 CIDR 표기법을 사용합니다. 자세한 내용은 IAM 사용자 설명서의 IP 주소 조건 연산자를 참조합니다.

```
{
  "Version": "2012-10-17",
  "Id": "IBPolicyId1",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "imagebuilder.GetComponent:*",
```

```
      "Resource": "arn:aws:imagebuilder:::examplecomponent/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"},
         "NotIpAddress": {"aws:SourceIp": "54.240.143.188/32"}
      }
    }
  ]
}
```

EC2 Image Builder에 대한 AWS 관리형 정책 사용

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 줍니다. AWS 는 새 AWS 서비스 가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 설명서의 AWS 관리형 정책을 참조하세요.

AWSImageBuilderFullAccess 정책

AWSImageBuilderFullAccess 정책은 연결된 역할에 Image Builder 리소스에 대한 전체 액세스 권한을 부여하여 역할이 Image Builder 리소스를 나열, 설명, 생성, 업데이트 및 삭제할 수 있도록 허용합니다. 또한이 정책은 리소스를 확인하거나의 계정에 대한 현재 리소스를 표시하는 데 AWS 서비스 필요한 관련에 대상 권한을 부여합니다 AWS Management Console.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- Image Builder - 관리 권한이 부여되어 해당 역할이 Image Builder 리소스를 나열, 설명, 생성, 업데이트 및 삭제할 수 있습니다.
- Amazon EC2 - 리소스 존재를 확인하거나 계정에 속한 리소스 목록을 가져오는 데 필요한 Amazon EC2 Describe 작업에 대한 액세스 권한이 부여됩니다.

- IAM - 이름에 'imagebuilder'가 포함된 인스턴스 프로필을 가져와 사용하고, `iam:GetRole` API 작업을 통해 Image Builder 서비스 연결 역할이 있는지 확인하고, Image Builder 서비스 연결 역할을 생성할 수 있는 액세스 권한이 부여됩니다.
- License Manager - 리소스에 대한 라이선스 구성 또는 라이선스를 나열할 수 있는 액세스 권한이 부여됩니다.
- Amazon S3 - 계정에 속한 버킷과 이름에 'imagebuilder'가 포함된 Image Builder 버킷을 나열할 수 있는 액세스 권한이 부여됩니다.
- Amazon SNS - Amazon SNS에 'imagebuilder'를 포함하는 주제에 대한 주제 소유권을 확인할 수 있는 쓰기 권한이 부여됩니다.

이 정책의 권한을 보려면AWS 관리형 정책 참조의 AWSImageBuilderFullAccess를 참조하세요.

AWSImageBuilderReadOnlyAccess 정책

이 AWSImageBuilderReadOnlyAccess 정책은 모든 Image Builder 리소스에 대한 읽기 전용 액세스 권한을 제공합니다. `iam:GetRole` API 작업을 통해 Image Builder 서비스 연결 역할이 존재하는지 확인할 수 있는 권한이 부여됩니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- Image Builder - Image Builder 리소스에 대한 읽기 전용 액세스 권한이 부여됩니다.
- IAM - API 작업을 통해 Image Builder 서비스 연결 역할이 있는지 확인할 수 있는 액세스 권한이 부여됩니다`iam:GetRole`.

이 정책의 권한을 보려면AWS 관리형 정책 참조의 AWSImageBuilderReadOnlyAccess를 참조하세요.

AWSServiceRoleForImageBuilder 정책

이 AWSServiceRoleForImageBuilder 정책은 Image Builder가 사용자를 대신하여 AWS 서비스 를 호출하도록 허용합니다.

권한 세부 정보

이 정책은 Image Builder 서비스 연결 역할이 Systems Manager를 통해 생성될 때 해당 역할에 연결됩니다. Image Builder 서비스 연결 역할에 대한 자세한 내용은 Image Builder의 IAM 서비스 연결 역할 사용 섹션을 참조하세요.

다음 정책에는 이러한 권한이 포함됩니다.

- CloudWatch Logs - 이름이 `/aws/imagebuilder/`(으)로 시작하는 모든 로그 그룹에 CloudWatch 로그를 생성하고 업로드할 수 있는 액세스 권한이 부여됩니다.
- Amazon EC2 - Image Builder가 계정에서 EC2 인스턴스를 생성하고 시작하는 이미지(AMIs)를 생성, 스냅샷 생성 및 등록할 수 있는 액세스 권한이 부여됩니다. Image Builder는 생성되거나 사용되는 이미지, 인스턴스 및 볼륨에 `CreatedBy: EC2 Image Builder` 또는 로 태그가 지정된 경우 필요에 따라 관련 스냅샷, 볼륨, 네트워크 인터페이스, 서브넷, 보안 그룹, 라이선스 구성 및 키 페어를 사용합니다`CreatedBy: EC2 Fast Launch`.

  Image Builder는 Amazon EC2 이미지, 인스턴스 속성, 인스턴스 상태, 계정에서 사용할 수 있는 인스턴스 유형, 시작 템플릿, 서브넷, 호스트, Amazon EC2 리소스의 태그에 대한 정보를 가져올 수 있습니다.

  Image Builder는 이미지 설정을 업데이트하여 이미지에 `CreatedBy: EC2 Image Builder` 태그가 지정된 계정에서 Windows 인스턴스의 빠른 실행을 활성화하거나 비활성화할 수 있습니다.

  또한 Image Builder는 계정에서 실행 중인 인스턴스를 시작, 중지 및 종료하고, Amazon EBS 스냅샷을 공유하고, 이미지를 생성 및 업데이트하고, 템플릿을 시작하고, 기존 이미지를 등록 취소하고, 태그를 추가하고, Ec2ImageBuilderCrossAccountDistributionAccess 정책을 통해 권한을 부여한 계정 간에 이미지를 복제할 수 있습니다. 앞에서 설명한 대로 이러한 모든 작업에는 Image Builder 태그 지정이 필요합니다.
- Amazon ECR - Image Builder는 컨테이너 이미지 취약성 스캔에 필요한 경우 리포지토리를 생성하고 생성한 리소스에 태그를 지정하여 작업 범위를 제한할 수 있는 액세스 권한을 부여합니다. 또한 Image Builder가 취약성 스냅샷을 생성한 후 스캔을 위해 생성한 컨테이너 이미지를 삭제할 수 있는 액세스 권한도 부여됩니다.
- EventBridge - Image Builder에서 EventBridge 규칙을 생성하고 관리할 수 있는 액세스 권한이 부여됩니다.
- IAM - Image Builder가 사용자 계정의 모든 역할을 Amazon EC2와 VM Import/Export에 넘길 수 있는 액세스 권한이 부여됩니다.
- Amazon Inspector - Image Builder는 Amazon Inspector가 빌드 인스턴스 스캔을 완료하는 시점을 결정하고 이를 허용하도록 구성된 이미지에 대한 결과를 수집할 수 있는 액세스 권한이 부여됩니다.
- AWS KMS - Amazon EBS에 Amazon EBS 볼륨을 암호화, 복호화 또는 재암호화할 수 있는 액세스 권한이 부여됩니다. 이는 Image Builder가 이미지를 빌드할 때 암호화된 볼륨이 작동하도록 하는 데 매우 중요합니다.

- 라이선스 관리자 - Image Builder에 `license-manager:UpdateLicenseSpecificationsForResource`를 통해 라이선스 관리자 사양을 업데이트할 수 있는 액세스 권한이 부여됩니다.
- Amazon SNS-계정 내 모든 Amazon SNS 주제에 대한 쓰기 권한이 부여됩니다.
- Systems Manager-Image Builder에서 Systems Manager 명령 및 호출, 인벤토리 항목을 나열하고, 인스턴스 정보와 자동화 실행 상태를 설명하고, 인스턴스 배치 지원에 대한 호스트를 설명하고, 명령 간접 호출 세부 정보를 가져올 수 있는 액세스 권한이 부여됩니다. 또한 Image Builder는 자동화 신호를 보내고 계정의 모든 리소스에 대한 자동화 실행을 중지할 수 있습니다.

  Image Builder는 `AWS-RunPowerShellScript`, `AWS-RunShellScript` 또는 `AWSEC2-RunSysprep` 스크립트 파일에 `"CreatedBy": "EC2 Image Builder"` 태그가 지정된 모든 인스턴스에 실행 명령 간접 호출을 실행할 수 있습니다. Image Builder를 사용하면 사용자 계정에서 이름이 `ImageBuilder`로 시작하는 자동화 문서에 대해 Systems Manager 자동화 실행을 시작할 수 있습니다.

  Image Builder는 또한 연결 문서가 `AWS-GatherSoftwareInventory`인 경우에 한해 사용자 계정의 모든 인스턴스에 대해 상태 관리자 연결을 만들거나 삭제할 수 있으며 사용자 계정에서 Systems Manager 서비스 연결 역할을 생성할 수 있습니다.
- AWS STS - 역할에 대한 신뢰 정책이 허용하는 모든 계정에 대해 귀하의 계정에서 EC2ImageBuilderDistributionCrossAccountRole로 명명된 역할을 맡을 수 있도록 Image Builder에 대한 액세스 권한이 부여됩니다. 교차 계정 Image Builder에 사용됩니다.

이 정책의 권한을 보려면AWS 관리형 정책 참조의 AWSServiceRoleForImageBuilder를 참조하세요.

Ec2ImageBuilderCrossAccountDistributionAccess 정책

Ec2ImageBuilderCrossAccountDistributionAccess 정책은 Image Builder가 대상 리전의 여러 계정에 이미지를 배포할 수 있는 권한을 부여합니다. 또한 Image Builder는 계정의 모든 Amazon EC2 이미지를 설명하고 복사하고 태그를 적용할 수 있습니다. 또한 이 정책은 `ec2:ModifyImageAttribute` API 작업을 통해 AMI 권한을 수정할 수 있는 권한을 부여합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- Amazon EC2 - Amazon EC2에 이미지 속성을 설명, 복사 및 수정하고 계정 내 모든 Amazon EC2 이미지에 대한 태그를 생성할 수 있는 액세스 권한이 부여됩니다.

이 정책의 권한을 보려면AWS 관리형 정책 참조의 Ec2ImageBuilderCrossAccountDistributionAccess를 참조하세요.

EC2ImageBuilderLifecycleExecutionPolicy 정책

이 EC2ImageBuilderLifecycleExecutionPolicy 정책은 Image Builder에 Image Builder 이미지 리소스 및 기본 리소스(AMI, 스냅샷)의 사용 중단, 비활성화 또는 삭제와 같은 작업을 수행할 수 있는 권한을 부여하여 이미지 수명 주기 관리 작업에 대한 자동화된 규칙을 지원합니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- Amazon EC2-Amazon EC2가 `CreatedBy: EC2 Image Builder`로 태그가 지정된 계정의 Amazon Machine Image(AMI)에 대해 다음 작업을 수행할 수 있는 액세스 권한이 부여됩니다.
  - AMI 활성화 및 비활성화.
  - 이미지 사용 중지 활성화 및 비활성화.
  - AMI 설명 및 등록 취소.
  - AMI 이미지 속성 설명 및 수정.
  - AMI와 연결된 볼륨 스냅샷 삭제.
  - 리소스에 대한 태그 검사.
  - 지원 중단을 위해 AMI에서 태그를 추가 또는 삭제.
- Amazon ECR-Amazon ECR이 `LifecycleExecutionAccess: EC2 Image Builder` 태그를 사용하여 ECR 리포지토리에서 다음과 같은 일괄 작업을 수행할 수 있는 액세스 권한이 부여됩니다. 일괄 작업은 자동화된 컨테이너 이미지 수명 주기 규칙을 지원합니다.
  - `ecr:BatchGetImage`
  - `ecr:BatchDeleteImage`

  `LifecycleExecutionAccess: EC2 Image Builder` 태그가 지정된 ECR 리포지토리에 대해서는 리포지토리 수준에서 액세스 권한이 부여됩니다.
- AWS 리소스 그룹 - Image Builder가 태그를 기반으로 리소스를 가져올 수 있는 액세스 권한이 부여됩니다.
- EC2 Image Builder-Image Builder가 Image Builder 이미지 리소스를 삭제할 수 있는 액세스 권한이 부여됩니다.

이 정책의 권한을 보려면AWS 관리형 정책 참조의 EC2ImageBuilderLifecycleExecutionPolicy를 참조하세요.

EC2InstanceProfileForImageBuilder 정책

이 EC2InstanceProfileForImageBuilder 정책은 EC2 인스턴스가 Image Builder와 함께 작동하는 데 필요한 최소 권한을 부여합니다. Systems Manager 에이전트를 사용하는 데 필요한 권한은 여기에 포함되지 않습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- CloudWatch Logs - 이름이 `/aws/imagebuilder/`(으)로 시작하는 모든 로그 그룹에 CloudWatch 로그를 생성하고 업로드할 수 있는 액세스 권한이 부여됩니다.
- Amazon EC2 - 볼륨 및 스냅샷을 설명하고, Image Builder가 생성한 볼륨 또는 스냅샷 리소스의 스냅샷을 생성하고, Image Builder 리소스에 대한 태그를 생성할 수 있는 액세스 권한이 부여됩니다.
- Image Builder - Image Builder 또는 AWS Marketplace 구성 요소를 가져올 수 있는 액세스 권한이 부여됩니다.
- AWS KMS - Image Builder 구성 요소가를 통해 암호화된 경우 복호화할 수 있는 액세스 권한이 부여됩니다 AWS KMS.
- Amazon S3 - 이름이 로 시작하는 Amazon S3 버킷에 저장된 객체 `ec2imagebuilder-`또는 ISO 파일 확장자가 있는 리소스를 가져올 수 있는 액세스 권한이 부여됩니다.

이 정책의 권한을 보려면AWS 관리형 정책 참조의 EC2InstanceProfileForImageBuilder를 참조하세요.

EC2InstanceProfileForImageBuilderECRContainerBuilds 정책

EC2InstanceProfileForImageBuilderECRContainerBuilds 정책은 Image Builder로 작업하여 Docker 이미지를 구축한 다음 Amazon ECR 컨테이너 리포지토리에 이미지를 등록 및 저장할 때 EC2 인스턴스에 필요한 최소 권한을 부여합니다. Systems Manager 에이전트를 사용하는 데 필요한 권한은 여기에 포함되지 않습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- CloudWatch Logs - 이름이 `/aws/imagebuilder/`(으)로 시작하는 모든 로그 그룹에 CloudWatch 로그를 생성하고 업로드할 수 있는 액세스 권한이 부여됩니다.

- Amazon ECR - Amazon ECR이 컨테이너 이미지를 가져오고, 등록하고, 저장하고, 인증 토큰을 받을 수 있는 액세스 권한이 부여됩니다.
- Image Builder - Image Builder 구성 요소 또는 컨테이너 레시피를 가져올 수 있는 액세스 권한이 부여됩니다.
- AWS KMS -를 통해 암호화된 경우 Image Builder 구성 요소 또는 컨테이너 레시피를 해독할 수 있는 액세스 권한이 부여됩니다 AWS KMS.
- Amazon S3 - 이름이 `ec2imagebuilder-`(으)로 시작하는 Amazon S3 버킷에 저장된 객체를 가져올 수 있는 액세스 권한이 부여됩니다.

이 정책의 권한을 보려면AWS 관리형 정책 참조의 EC2InstanceProfileForImageBuilderECRContainerBuilds를 참조하세요.

AWS 관리형 정책에 대한 Image Builder 업데이트

이 섹션에서는이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 Image Builder의 AWS 관리형 정책 업데이트에 대한 정보를 제공합니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 Image Builder 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AWSServiceRoleForImageBuilder - 기존 정책에 대한 업데이트	Image Builder는 Microsoft 클라이언트 OS ISO 파일을 기본 이미지로 가져올 수 있도록 서비스 역할을 다음과 같이 변경했습니다. • Image Builder가 스냅샷을 생성하고 확인된 ISO 디스크 파일에서 기준 운영 체제를 가져온 AMI를 생성 및 등록할 수 있도록 ec2:RegisterImage가 추가되었습니다.	2024년 12월 30일
EC2InstanceProfileForImageBuilder -기존 정책 업데이트	Image Builder는 디스크 이미지 파일에서 이미지 생성을 지원하기 위해 인스턴스 프로파	2024년 12월 30일

변경 사항	설명	날짜
	일 정책을 다음과 같이 변경했습니다. • 세부 정보를 검색할 모든 리소스에 대한 DescribeVolumes 및 DescribeSnapshots ec2 작업이 추가되었습니다. 또한 Image Builder에서 생성한 리소스에 대해 볼륨 및 스냅샷 리소스에 대한 CreateSnapshot 및 CreateTags 작업을 추가했습니다. "ISO" 파일 확장명이 있는 리소스에 대해 s3:GetObject를 추가했습니다.	
EC2InstanceProfileForImageBuilder - 정책 업데이트	Image Builder는 Image Builder가 AWS Marketplace 구성 요소를 가져올 수 있도록 `EC2InstanceProfileForImageBuilder` 정책을 업데이트했습니다.	2024년 12월 2일
EC2ImageBuilderLifecycleExecutionPolicy - 새 정책	Image Builder는 이미지 수명 주기 관리에 대한 권한이 포함된 새 `EC2ImageBuilderLifecycleExecutionPolicy` 정책을 추가했습니다.	2023년 11월 17일

변경 사항	설명	날짜
AWSServiceRoleForImageBuilder-기존 정책 업데이트	Image Builder는 인스턴스 배치 지원 제공을 위해 서비스 역할을 다음과 같이 변경했습니다. • ec2:DescribeHosts를 추가하면 Image Builder가 hostId를 폴링하여 인스턴스를 시작하는 데 유효한 상태인지 확인할 수 있습니다. • Image Builder에서 명령 간접 호출의 세부 정보를 가져오는 데 사용하는 메서드를 개선하기 위해 API 작업인 ssm:GetCommandInvocation을 추가했습니다.	2023년 10월 19일

변경 사항	설명	날짜
AWSServiceRoleForImageBuilder -기존 정책 업데이트	Image Builder는 인스턴스 배치 지원 제공을 위해 서비스 역할을 다음과 같이 변경했습니다. • 추가된 ec2:DescribeHosts를 사용하면 Image Builder가 hostId를 폴링하여 인스턴스를 시작하기에 적합한 상태인지 확인할 수 있습니다. • Image Builder에서 명령 간접 호출의 세부 정보를 가져오는 데 사용하는 메서드를 개선하기 위해 API 작업인 ssm:GetCommandInvocation을 추가했습니다.	2023년 9월 28일

변경 사항	설명	날짜
AWSServiceRoleForImageBuilder-기존 정책 업데이트	Image Builder는 Image Builder 워크플로에서 AMI 및 ECR 컨테이너 이미지 빌드 모두에 대한 취약성 결과를 수집할 수 있도록 서비스 역할을 다음과 같이 변경했습니다. 새 권한은 CVE 탐지 및 보고 기능을 지원합니다. • Image Builder에서 Amazon Inspector가 테스트 인스턴스 스캔을 완료하는 시점을 결정하고 이를 허용하도록 구성된 이미지에 대한 결과를 수집할 수 있도록 Inspector2:ListCoverage 및 Inspector2:ListFindings가 추가되었습니다. • Image Builder가 리포지토리에 `CreatedBy: EC2 Image Builder`(생성 시 태그 지정) 태그를 지정해야 하는 요구 사항과 함께 ECR:CreateRepository가 추가되었습니다. 또한 동일한 CreatedBy 태그 제약 조건과 `image-builder-*` (으)로 시작하는 리포지토리 이름을 요구하는 추가 제약 조건을 사용하여 ecr:TagResource(생성 시 태그 필요)를 추가했습니다. 이런 이름 제약 조건은 권한 에스컬	2023년 3월 30일

변경 사항	설명	날짜
	레이션을 방지하고 Image Builder가 생성하지 않은 리포지토리의 변경을 방지합니다. • `CreatedBy: EC2 Image Builder` 태그가 지정된 ECR 리포지토리에 대해 ECR: BatchDeleteImage를 추가했습니다. 이 권한을 사용하려면 리포지토리 이름이 `image-builder-*` (으)로 시작해야 합니다. • 이름에 `ImageBuilder-*` 포함된 Amazon EventBridge 관리 규칙을 생성하고 관리할 수 있도록 Image Builder에 대한 이벤트 권한을 추가했습니다.	
AWSServiceRoleForImageBuilder-기존 정책 업데이트	Image Builder는 서비스 역할을 다음과 같이 변경했습니다. • 고객이 라이선스 구성과 연결된 기본 이미지 AMI를 사용할 수 있도록 EC2:RunInstance 호출을 위한 리소스로 License Manager 라이선스를 추가했습니다.	2022년 3월 22일

변경 사항	설명	날짜
AWSServiceRoleForImageBuilder-기존 정책 업데이트	Image Builder는 서비스 역할을 다음과 같이 변경했습니다. • Windows 인스턴스의 빠른 시작을 활성화하거나 비활성화할 수 있도록 EC2 EnableFastLaunch API 작업에 대한 권한을 추가했습니다. • EC2:CreateTags 작업 및 리소스 태그 조건에 대한 범위를 더 좁혔습니다.	2022년 2월 21일
AWSServiceRoleForImageBuilder-기존 정책 업데이트	Image Builder는 서비스 역할을 다음과 같이 변경했습니다. • VMIE 서비스를 호출하여 VM을 가져오고 기본 AMI를 생성할 수 있는 권한이 추가되었습니다. • EC2:CreateTags 작업 및 리소스 태그 조건에 대한 범위를 좁혔습니다.	2021년 11월 20일
AWSServiceRoleForImageBuilder-기존 정책 업데이트	Image Builder는 둘 이상의 인벤토리 연결로 인해 이미지 빌드가 중단되는 문제를 해결하기 위해 새 사용 권한을 추가했습니다.	2021년 8월 11일

변경 사항	설명	날짜
AWSImageBuilderFullAccess-기존 정책 업데이트	Image Builder는 전체 액세스 역할을 다음과 같이 변경했습니다. • `ec2:DescribeInstanceTypeOffereings` (을)를 허용하는 권한 추가. • Image Builder 콘솔이 계정에서 사용 가능한 인스턴스 유형을 정확하게 반영하도록 `ec2:DescribeInstanceTypeOffereings` (을)를 호출할 수 있는 권한이 추가되었습니다.	2021년 4월 13일
Image Builder에서 변경 내용 추적 시작	Image Builder가 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 4월 2일

Image Builder의 IAM 서비스 연결 역할 사용

EC2 Image Builder는 AWS Identity and Access Management (IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Image Builder에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Image Builder에서 사전 정의하며 서비스가 AWS 서비스 사용자를 대신하여 다른를 호출하는 데 필요한 모든 권한을 포함합니다.

서비스 연결 역할을 사용하면 필요한 권한을 수동으로 추가할 필요가 없으므로 Image Builder를 더 효율적으로 설정할 수 있습니다. Image Builder는 서비스 연결 역할의 권한을 정의하며, 다르게 정의되지 않는 한 Image Builder만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함됩니다. 권한 정책은 다른 어떤 IAM 엔터티에도 연결할 수 없습니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스을(를) 참조하고 서비스 연결 역할 열에 예가 있는 서비스를 찾아보세요. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Image Builder에 대한 서비스 연결 역할 권한

Image Builder는 AWSServiceRoleForImageBuilder 서비스 연결 역할을 사용하여 EC2 Image Builder가 사용자를 대신하여 AWS 리소스에 액세스할 수 있도록 허용합니다. 서비스 연결 역할은 imagebuilder.amazonaws.com 서비스를 신뢰하여 역할을 맡습니다.

이 서비스 연결 역할은 수동으로 생성할 필요가 없습니다. AWS 관리 콘솔, AWS CLI또는 AWS API에서 첫 번째 Image Builder 이미지를 생성하면 Image Builder가 서비스 연결 역할을 생성합니다.

다음 작업을 수행하면 새 이미지가 생성됩니다.

- Image Builder 콘솔에서 파이프라인 마법사를 실행하여 사용자 지정 이미지를 생성합니다.
- 다음 API 작업 중 하나 또는 해당 AWS CLI 명령을 사용합니다.
  - CreateImage API 작업(create-image의 AWS CLI).
  - ImportVmImage API 작업(import-vm-image의 AWS CLI).
  - StartImagePipelineExecution API 작업(start-image-pipeline-execution의 AWS CLI).

> **Important**
>
> 서비스 연결 역할이 계정에서 삭제된 경우 동일한 프로세스를 사용하여 다시 생성할 수 있습니다. 첫 번째 EC2 Image Builder 리소스를 생성하면 Image Builder가 서비스 연결 역할을 다시 생성합니다.

AWSServiceRoleForImageBuilder에 대한 권한을 보려면 AWSServiceRoleForImageBuilder 정책 페이지를 참조하세요. 서비스 연결 역할에 대한 권한 구성에 대해 자세히 알아보려면 IAM 사용자 설명서의 서비스 연결 역할 권한을 참조하세요.

계정에서 Image Builder 서비스 연결 역할 제거

IAM 콘솔, AWS CLI또는 AWS API를 사용하여 계정에서 Image Builder의 서비스 연결 역할을 수동으로 제거할 수 있습니다. 하지만 이 작업을 수행하기 전에 해당 리소스를 참조하는 Image Builder 리소스가 활성화되어 있지 않은지 확인해야 합니다.

> **Note**
>
> 리소스를 삭제하려 할 때 Image Builder 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

AWSServiceRoleForImageBuilder 역할이 사용하는 Image Builder 리소스 정리

1. 시작하기 전에 실행 중인 파이프라인 빌드가 없는지 확인하세요. 실행 중인 빌드를 취소하려면 AWS CLI의 `cancel-image-creation` 명령을 사용합니다.

```
aws imagebuilder cancel-image-creation --image-build-version-
arn arn:aws:imagebuilder:us-east-1:123456789012:image-pipeline/sample-pipeline
```

2. 수동 빌드 프로세스를 사용하도록 모든 파이프라인 일정을 변경하거나 다시 사용하지 않으려면 삭제하세요. 사용자 삭제에 대한 자세한 내용은 오래되거나 사용되지 않는 Image Builder 리소스 삭제 섹션을 참조하세요.

IAM을 사용하여 서비스 연결 역할을 삭제

IAM 콘솔, AWS CLI또는 AWS API를 사용하여 계정에서 `AWSServiceRoleForImageBuilder` 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용자 설명서의 서비스 연결 역할 삭제 섹션을 참조하세요.

Image Builder 서비스 연결 역할이 지원되는 리전

Image Builder는 서비스를 사용할 수 있는 모든 AWS 리전에서 서비스 연결 역할 사용을 지원합니다. 지원되는 AWS 리전 목록은 섹션을 참조하세요AWS 리전 및 엔드포인트.

Image Builder의 IAM 문제 해결

주제

- Image Builder에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 Image Builder 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

Image Builder에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 `mateojackson` IAM 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `imagebuilder:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 imagebuilder:GetWidget on resource: my-example-widget
```

이 경우, `imagebuilder:GetWidget` 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 `mateojackson` 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Image Builder에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예시 오류는 `marymajor`(이)라는 IAM 사용자가 콘솔을 사용하여 Image Builder에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 Image Builder 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조합니다.

- Image Builder에서 이러한 기능을 지원하는지 여부를 알아보려면 Image Builder가 IAM 정책 및 역할을 사용하는 방법(을)를 참조합니다.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.

- 리소스에 대한 액세스 권한을 타사에 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

Image Builder의 규정 준수 검증 리소스

EC2 Image Builder는 규정 AWS 준수 프로그램의 범위에 속하지 않습니다.

특정 규정 준수 프로그램의 범위 AWS 서비스 내 목록은 규정 준수 AWS 프로그램 제공 범위 내 서비스 규정 준수 참조하세요. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Artifact에서 AWS 보고서 다운로드.

Image Builder 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS 은 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 설명서 – 이 배포 설명서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- 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 -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

Image Builder 이미지에의 규정 준수 제품 AWS Marketplace 또는 AWS Task Orchestrator and Executor (AWSTOE)의 구성 요소를 통합하여 이미지가 규정을 준수하는지 확인할 수 있습니다. 자세한 내용은 Image Builder 이미지를 위한 규정 준수 제품 단원을 참조하십시오.

Image Builder의 데이터 중복성 및 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이

높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

EC2 Image Builder 서비스를 사용하면 한 리전에 빌드된 이미지를 다른 리전과 함께 배포하여 AMI에 대한 다중 리전 복원력을 제공할 수 있습니다. 이미지 파이프라인, 레시피 또는 구성 요소를 '백업'하는 메커니즘은 없습니다. 레시피 및 구성 요소 문서를 Image Builder 서비스 외부(예: Amazon S3 버킷) 에 저장할 수 있습니다.

EC2 Image Builder는 고가용성(HA)을 위해 구성될 수 없습니다. 이미지를 여러 리전에 배포하여 이미지의 가용성을 높일 수 있습니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

Image Builder의 인프라 보안

AWS 글로벌 네트워크는 보안 기능을 제공하고 EC2 Image Builder와 같은 서비스에 대한 네트워크 액세스를 제어합니다. AWS (이)가 해당 서비스에 제공하는 인프라 보안에 대한 자세한 내용은 AWS 보안에 대한 소개 백서에 있는 인프라 보안 섹션을 참조합니다.

Image Builder API 작업에 대한 요청을 AWS 글로벌 네트워크를 통해 보내려면 클라이언트 소프트웨어가 다음 보안 지침을 준수해야 합니다.

- Image Builder API 작업에 대한 요청을 보내려면 클라이언트 소프트웨어에서 지원되는 버전의 전송 계층 보안(TLS)을 사용해야 합니다.

  > **Note**
  >
  > AWS 는 TLS 버전 1.0 및 1.1에 대한 지원을 단계적으로 중단합니다. 계속 연결할 수 있도록 클라이언트 소프트웨어가 TLS 버전 1.2 이상을 사용하도록 업데이트하는 것이 좋습니다. 자세한 내용은 AWS 보안 블로그 게시물을 참조하세요.

- 클라이언트 소프트웨어는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)와 같은 전달 완전 보안(PFS)이 포함된 암호화 스위트를 지원해야 합니다. Java 7 이상과 같은 최신 시스템은 대부분 이러한 모드를 지원합니다.
- 액세스 키 ID 및 AWS Identity and Access Management (IAM) 보안 주체와 연결된 보안 액세스 키를 사용하여 API 요청에 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)을 사용하여 요청에 따라 임시 보안 인증을 생성할 수 있습니다.

또한 Image Builder가 이미지를 빌드하고 테스트하는 데 사용하는 EC2 인스턴스는 AWS Systems Manager에 액세스할 수 있어야 합니다.

Image Builder 이미지에 대한 패치 관리

AWS 는 다음 운영 체제에 적용되는 최신 업데이트 및 보안 패치가 있는 업데이트된 관리형 AMIs를 매월 제공합니다. 이러한 AMI를 사용자 지정의 기본 이미지로 사용할 수 있습니다. 자세한 내용은 지원되는 운영 체제 단원을 참조하십시오.

- Amazon Linux 2, AL2023, Red Hat Enterprise Linux (RHEL), CentOS, Ubuntu, SUSE Linux Enterprise Server를 포함한 Linux 배포판
- Windows Server 2016 이상
- macOS 10.14.x 이상

사용자 지정 이미지를 생성한 후 공동 책임 모델에 따라 Amazon EC2 시스템 패치를 지정해야 합니다. 애플리케이션 워크로드의 EC2 인스턴스를 쉽게 교체할 수 있는 경우 기본 AMI를 업데이트하고 이 이미지를 기반으로 모든 컴퓨팅 노드를 재배포하는 것이 가장 효율적일 수 있습니다.

> **Note**
>
> macOS 패치의 경우 기본 이미지에 최신 관리형 AMI를 사용하는 레시피의 새 버전을 생성한 다음 레시피 및 기타 이미지 빌드 리소스에서 업데이트된 사용자 지정 이미지를 빌드하는 것이 좋습니다. Mac 인스턴스를 쉽게 교체할 수 없는 경우 자세한 내용은 Amazon EC2 사용 설명서의 Mac 인스턴스에서 운영 체제 및 소프트웨어 업데이트 페이지를 참조하세요.

Image Builder AMI를 최신 상태로 유지할 수 있는 두 가지 방법은 다음과 같습니다.

- AWS에서 제공되는 패치 구성 요소 — EC2 Image Builder는 보류 중인 모든 운영 체제 업데이트를 설치하는 다음 빌드 구성 요소를 제공합니다.
  - `update-linux`
  - `update-windows`

  이러한 구성 요소는 `UpdateOS` 작업 모듈을 사용합니다. 자세한 내용은 UpdateOS 단원을 참조하십시오. AWS에 제공된 구성 요소 목록에서 구성 요소를 선택하여 이미지 빌드 파이프라인에 구성 요소를 추가할 수 있습니다.

- 패치 작업이 포함된 사용자 지정 빌드 구성 요소 - 지원되는 AMI의 운영 체제에 패치를 선택적으로 설치하거나 업데이트하려면 Image Builder 구성 요소를 작성하여 필요한 패치를 설치할 수 있습니다. 사용자 지정 구성 요소는 셸 스크립트(Bash 또는 PowerShell)를 사용하여 패치를 설치하거나 UpdateOS 작업 모듈을 사용하여 설치 또는 제외할 패치를 지정할 수 있습니다. 자세한 내용은 AWSTOE 구성 요소 관리자가 지원하는 작업 모듈 단원을 참조하십시오.

  UpdateOS 작업 모듈을 사용하는 구성 요소(Linux 및 Windows만 해당. UpdateOS 작업 모듈은 macOS에서 지원되지 않습니다.)

```
schemaVersion: 1.0
phases:
  - name: build
 steps:
   - name: UpdateOS
  action: UpdateOS
```

  Bash를 사용하여 yum 업데이트를 설치하는 구성 요소

```
schemaVersion: 1.0
phases:
  - name: build
 steps:
   - name: InstallYumUpdates
  action: ExecuteBash
  inputs:
    commands:
   - sudo yum update -y
```

Image Builder의 보안 모범 사례

EC2 Image Builder는 자체 보안 정책을 개발하고 구현할 때 고려해야 할 여러 보안 기능을 제공합니다. 다음 모범 사례는 일반적인 지침이며 완벽한 보안 솔루션을 나타내지는 않습니다. 이러한 모범 사례는 환경에 적절하지 않거나 충분하지 않을 수 있으므로 참고용으로만 사용합니다.

- Image Builder 레시피에는 지나치게 허용적인 보안 그룹을 사용하지 않습니다.
- 신뢰할 수 없는 계정에 있는 이미지는 공유하지 않습니다.
- 비공개 또는 민감한 데이터가 포함된 이미지는 공개하지 않습니다.
- 이미지 빌드 중에 사용 가능한 모든 Windows 또는 Linux 보안 패치를 적용합니다.

- 관리형 AMI 업데이트를 macOS 레시피에 주기적으로 적용하고 새 이미지를 생성하여 최신 보안 패치가 있는 인스턴스를 시작합니다.

이미지를 테스트하여 보안 상태 및 해당 보안 규정 준수 수준을 검증하는 것이 좋습니다. Amazon Inspector와 같은 솔루션은 이미지의 보안 및 규정 준수 상태를 검증하는 데 도움이 될 수 있습니다.

Image Builder 파이프라인용 IMDSv2

Image Builder 파이프라인이 실행되면 Image Builder에서 이미지를 빌드하고 테스트하는 데 사용하는 EC2 인스턴스를 시작하기 위한 HTTP 요청을 보냅니다. 파이프라인이 시작 요청에 사용하는 IMDS 버전을 구성하려면 Image Builder 인프라 구성 인스턴스 메타데이터 설정에서 `httpTokens` 파라미터를 설정합니다.

> **Note**
>
> Image Builder가 파이프라인 빌드에서 시작하는 모든 EC2 인스턴스가 IMDSv2를 사용하도록 구성하여 인스턴스 메타데이터 검색 요청에 서명된 토큰 헤더가 필요하도록 하는 것이 좋습니다.

Image Builder 인프라 구성에 대한 자세한 내용은 Image Builder 인프라 구성 관리(을)를 참조합니다. Linux 이미지용 EC2 인스턴스 메타데이터 옵션에 대한 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스 메타데이터 옵션 구성하기를 참조합니다. Windows 이미지의 경우 Amazon EC2 사용 설명서의 인스턴스 메타데이터 옵션 구성하기를 참조합니다.

빌드 후 정리 필요

Image Builder가 사용자 지정 이미지의 모든 빌드 단계를 완료한 후 Image Builder는 테스트 및 이미지 생성을 위해 빌드 인스턴스를 준비합니다. 스냅샷을 생성하기 위해 빌드 인스턴스를 종료하기 전에 Image Builder는 이미지의 보안을 보장하기 위해 다음과 같은 정리 작업을 수행합니다.

Linux

Image Builder 파이프라인은 최종 이미지가 보안 모범 사례를 따르는지 확인하고 스냅샷으로 전달해서는 안 되는 빌드 아티팩트 또는 설정을 제거하는 데 도움이 되는 정리 스크립트를 실행합니다. 하지만 스크립트의 일부 섹션을 건너뛰거나 사용자 데이터를 완전히 오버라이드할 수 있습니다. 따라서 Image Builder 파이프라인에서 생성된 이미지가 특정 규제 기준을 반드시 준수하는 것은 아닙니다.

파이프라인이 빌드 및 테스트 단계를 완료하면 Image Builder는 출력 이미지를 생성하기 직전에 다음 정리 스크립트를 자동으로 실행합니다.

> **Important**
>
> 레시피의 사용자 데이터를 오버라이드하면 스크립트가 실행되지 않습니다. 이 경우 `perform_cleanup`(이)라는 빈 파일을 생성하는 사용자 데이터에 명령을 포함해야 합니다. Image Builder는 이 파일을 탐지하고 새 이미지를 생성하기 전에 정리 스크립트를 실행합니다.

```
#!/bin/bash
if [[ ! -f {{workingDirectory}}/perform_cleanup ]]; then
    echo "Skipping cleanup"
    exit 0
else
    sudo rm -f {{workingDirectory}}/perform_cleanup
fi

function cleanup() {
    FILES=("$@")
    for FILE in "${FILES[@]}"; do
        if [[ -f "$FILE" ]]; then
            echo "Deleting $FILE";
            sudo shred -zuf $FILE;
        fi;
        if [[ -f $FILE ]]; then
            echo "Failed to delete '$FILE'. Failing."
            exit 1
        fi;
    done
};


# Clean up for cloud-init files
CLOUD_INIT_FILES=(
    "/etc/sudoers.d/90-cloud-init-users"
    "/etc/locale.conf"
    "/var/log/cloud-init.log"
    "/var/log/cloud-init-output.log"
)
if [[ -f {{workingDirectory}}/skip_cleanup_cloudinit_files ]]; then
```

```
    echo "Skipping cleanup of cloud init files"
else
    echo "Cleaning up cloud init files"
    cleanup "${CLOUD_INIT_FILES[@]}"
    if [[ $( sudo find /var/lib/cloud -type f | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting files within /var/lib/cloud/*"
        sudo find /var/lib/cloud -type f -exec shred -zuf {} \;
    fi;

    if [[ $( sudo ls /var/lib/cloud | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/lib/cloud/*"
        sudo rm -rf /var/lib/cloud/* || true
    fi;
fi;


# Clean up for temporary instance files
INSTANCE_FILES=(
    "/etc/.updated"
    "/etc/aliases.db"
    "/etc/hostname"
    "/var/lib/misc/postfix.aliasesdb-stamp"
    "/var/lib/postfix/master.lock"
    "/var/spool/postfix/pid/master.pid"
    "/var/.updated"
    "/var/cache/yum/x86_64/2/.gpgkeyschecked.yum"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_files ]]; then
    echo "Skipping cleanup of instance files"
else
    echo "Cleaning up instance files"
    cleanup "${INSTANCE_FILES[@]}"
fi;


# Clean up for ssh files
SSH_FILES=(
    "/etc/ssh/ssh_host_rsa_key"
    "/etc/ssh/ssh_host_rsa_key.pub"
    "/etc/ssh/ssh_host_ecdsa_key"
    "/etc/ssh/ssh_host_ecdsa_key.pub"
    "/etc/ssh/ssh_host_ed25519_key"
    "/etc/ssh/ssh_host_ed25519_key.pub"
    "/root/.ssh/authorized_keys"
```

```
)
if [[ -f {{workingDirectory}}/skip_cleanup_ssh_files ]]; then
    echo "Skipping cleanup of ssh files"
else
    echo "Cleaning up ssh files"
    cleanup "${SSH_FILES[@]}"
    USERS=$(ls /home/)
    for user in $USERS; do
        echo Deleting /home/"$user"/.ssh/authorized_keys;
        sudo find /home/"$user"/.ssh/authorized_keys -type f -exec shred -zuf {} \;
    done
    for user in $USERS; do
        if [[ -f /home/"$user"/.ssh/authorized_keys ]]; then
            echo Failed to delete /home/"$user"/.ssh/authorized_keys;
            exit 1
        fi;
    done;
fi;


# Clean up for instance log files
INSTANCE_LOG_FILES=(
    "/var/log/audit/audit.log"
    "/var/log/boot.log"
    "/var/log/dmesg"
    "/var/log/cron"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_log_files ]]; then
    echo "Skipping cleanup of instance log files"
else
    echo "Cleaning up instance log files"
    cleanup "${INSTANCE_LOG_FILES[@]}"
fi;

# Clean up for TOE files
if [[ -f {{workingDirectory}}/skip_cleanup_toe_files ]]; then
    echo "Skipping cleanup of TOE files"
else
    echo "Cleaning TOE files"
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]];
 then
        echo "Deleting files within {{workingDirectory}}/TOE_*"
        sudo find {{workingDirectory}}/TOE_* -type f -exec shred -zuf {} \;
    fi
```

```
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]];
 then
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]];
 then
        echo "Deleting {{workingDirectory}}/TOE_*"
        sudo rm -rf {{workingDirectory}}/TOE_*
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]];
 then
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
    fi
fi

# Clean up for ssm log files
if [[ -f {{workingDirectory}}/skip_cleanup_ssm_log_files ]]; then
    echo "Skipping cleanup of ssm log files"
else
    echo "Cleaning up ssm log files"
    if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
        echo "Deleting files within /var/log/amazon/ssm/*"
        sudo find /var/log/amazon/ssm -type f -exec shred -zuf {} \;
    fi
    if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete /var/log/amazon/ssm"
        exit 1
    fi
    if [[ -d "/var/log/amazon/ssm" ]]; then
        echo "Deleting /var/log/amazon/ssm/*"
        sudo rm -rf /var/log/amazon/ssm
    fi
    if [[ -d "/var/log/amazon/ssm" ]]; then
        echo "Failed to delete /var/log/amazon/ssm"
        exit 1
    fi
fi


if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
    echo "Deleting /var/log/sa/sa*"
    sudo shred -zuf /var/log/sa/sa*
```

```
fi
if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
    echo "Failed to delete /var/log/sa/sa*"
    exit 1
fi

if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
        echo "Deleting /var/lib/dhclient/dhclient*.lease"
        sudo shred -zuf /var/lib/dhclient/dhclient*.lease
fi
if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
        echo "Failed to delete /var/lib/dhclient/dhclient*.lease"
        exit 1
fi

if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
        echo "Deleting files within /var/tmp/*"
        sudo find /var/tmp -type f -exec shred -zuf {} \;
fi
if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete /var/tmp"
        exit 1
fi
if [[ $( sudo ls /var/tmp | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/tmp/*"
        sudo rm -rf /var/tmp/*
fi

# Shredding is not guaranteed to work well on rolling logs

if [[ -f "/var/lib/rsyslog/imjournal.state" ]]; then
        echo "Deleting /var/lib/rsyslog/imjournal.state"
        sudo shred -zuf /var/lib/rsyslog/imjournal.state
        sudo rm -f /var/lib/rsyslog/imjournal.state
fi

if [[ $( sudo ls /var/log/journal/ | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/log/journal/*"
        sudo find /var/log/journal/ -type f -exec shred -zuf {} \;
        sudo rm -rf /var/log/journal/*
fi
```

```
sudo touch /etc/machine-id
```

Windows

Image Builder 파이프라인은 Windows 이미지를 사용자 지정한 후 Microsoft Sysprep 유틸리티를 실행합니다. 이러한 작업은 AWS 이미지 강화 및 정리 모범 사례를 따릅니다.

macOS

Image Builder 파이프라인은 최종 이미지가 보안 모범 사례를 따르는지 확인하고 스냅샷으로 전달해서는 안 되는 빌드 아티팩트 또는 설정을 제거하는 데 도움이 되는 정리 스크립트를 실행합니다. 하지만 스크립트의 일부 섹션을 건너뛰거나 사용자 데이터를 완전히 오버라이드할 수 있습니다. 따라서 Image Builder 파이프라인에서 생성된 이미지가 특정 규제 기준을 반드시 준수하는 것은 아닙니다.

파이프라인이 빌드 및 테스트 단계를 완료하면 Image Builder는 출력 이미지를 생성하기 직전에 다음 정리 스크립트를 자동으로 실행합니다.

> **Important**
>
> 레시피의 사용자 데이터를 오버라이드하면 스크립트가 실행되지 않습니다. 이 경우 `perform_cleanup`(이)라는 빈 파일을 생성하는 사용자 데이터에 명령을 포함해야 합니다. Image Builder는 이 파일을 탐지하고 새 이미지를 생성하기 전에 정리 스크립트를 실행합니다.

```
#!/bin/bash
if [[ ! -f {{workingDirectory}}/perform_cleanup ]]; then
  echo "Skipping cleanup"
  exit 0
else
  sudo rm -f {{workingDirectory}}/perform_cleanup
fi

function cleanup() {
  FILES=("$@")
  for FILE in "${FILES[@]}"; do
      if [[ -f "$FILE" ]]; then
          echo "Deleting $FILE";
          sudo rm -f $FILE;
      fi;
      if [[ -f $FILE ]]; then
```

```
          echo "Failed to delete '$FILE'. Failing."
          exit 1
      fi;
  done
};

# Clean up for cloud-init files
CLOUD_INIT_FILES=(
  "/etc/sudoers.d/90-cloud-init-users"
  "/etc/locale.conf"
  "/var/log/cloud-init.log"
  "/var/log/cloud-init-output.log"
)
if [[ -f {{workingDirectory}}/skip_cleanup_cloudinit_files ]]; then
  echo "Skipping cleanup of cloud init files"
else
  echo "Cleaning up cloud init files"
  cleanup "${CLOUD_INIT_FILES[@]}"
  if [[ $( sudo find /var/lib/cloud -type f | sudo wc -l ) -gt 0 ]]; then
      echo "Deleting files within /var/lib/cloud/*"
      sudo find /var/lib/cloud -type f -exec rm -f {} \;
  fi;

  if [[ $( sudo ls /var/lib/cloud | sudo wc -l ) -gt 0 ]]; then
      echo "Deleting /var/lib/cloud/*"
      sudo rm -rf /var/lib/cloud/* || true
  fi;
fi;


# Clean up for temporary instance files
INSTANCE_FILES=(
  "/etc/.updated"
  "/etc/aliases.db"
  "/etc/hostname"
  "/var/lib/misc/postfix.aliasesdb-stamp"
  "/var/lib/postfix/master.lock"
  "/var/spool/postfix/pid/master.pid"
  "/var/.updated"
  "/var/cache/yum/x86_64/2/.gpgkeyschecked.yum"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_files ]]; then
  echo "Skipping cleanup of instance files"
else
```

```
  echo "Cleaning up instance files"
  cleanup "${INSTANCE_FILES[@]}"
fi;


# Clean up for ssh files
SSH_FILES=(
  "/etc/ssh/ssh_host_rsa_key"
  "/etc/ssh/ssh_host_rsa_key.pub"
  "/etc/ssh/ssh_host_ecdsa_key"
  "/etc/ssh/ssh_host_ecdsa_key.pub"
  "/etc/ssh/ssh_host_ed25519_key"
  "/etc/ssh/ssh_host_ed25519_key.pub"
  "/root/.ssh/authorized_keys"
)
if [[ -f {{workingDirectory}}/skip_cleanup_ssh_files ]]; then
  echo "Skipping cleanup of ssh files"
else
  echo "Cleaning up ssh files"
  cleanup "${SSH_FILES[@]}"
  USERS=$(ls /home/)
  for user in $USERS; do
      echo Deleting /home/"$user"/.ssh/authorized_keys;
      sudo find /home/"$user"/.ssh/authorized_keys -type f -exec rm -f {} \;
  done
  for user in $USERS; do
      if [[ -f /home/"$user"/.ssh/authorized_keys ]]; then
          echo Failed to delete /home/"$user"/.ssh/authorized_keys;
          exit 1
      fi;
  done;
fi;


# Clean up for instance log files
INSTANCE_LOG_FILES=(
  "/var/log/audit/audit.log"
  "/var/log/boot.log"
  "/var/log/dmesg"
  "/var/log/cron"
  "/var/log/amazon/ec2/ec2-macos-init.log"
  "/var/log/amazon/ec2/ena-ethernet.log"
  "/var/log/amazon/ec2/system-monitoring.log"
)
```

```
if [[ -f {{workingDirectory}}/skip_cleanup_instance_log_files ]]; then
  echo "Skipping cleanup of instance log files"
else
  echo "Cleaning up instance log files"
  cleanup "${INSTANCE_LOG_FILES[@]}"
fi;

# Clean up for TOE files
if [[ -f {{workingDirectory}}/skip_cleanup_toe_files ]]; then
  echo "Skipping cleanup of TOE files"
else
  echo "Cleaning TOE files"
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]]; then
      echo "Deleting files within {{workingDirectory}}/TOE_*"
      sudo find {{workingDirectory}}/TOE_* -type f -exec rm -f {} \;
  fi
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]]; then
      echo "Failed to delete {{workingDirectory}}/TOE_*"
      exit 1
  fi
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]]; then
      echo "Deleting {{workingDirectory}}/TOE_*"
      sudo rm -rf {{workingDirectory}}/TOE_*
  fi
  if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]]; then
      echo "Failed to delete {{workingDirectory}}/TOE_*"
      exit 1
  fi
fi

# Clean up for ssm log files
if [[ -f {{workingDirectory}}/skip_cleanup_ssm_log_files ]]; then
  echo "Skipping cleanup of ssm log files"
else
  echo "Cleaning up ssm log files"
  if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
      echo "Deleting files within /var/log/amazon/ssm/*"
      sudo find /var/log/amazon/ssm -type f -exec rm -f {} \;
  fi
  if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
      echo "Failed to delete /var/log/amazon/ssm"
      exit 1
  fi
  if [[ -d "/var/log/amazon/ssm" ]]; then
```

```
      echo "Deleting /var/log/amazon/ssm/*"
      sudo rm -rf /var/log/amazon/ssm
  fi
  if [[ -d "/var/log/amazon/ssm" ]]; then
      echo "Failed to delete /var/log/amazon/ssm"
      exit 1
  fi
fi


if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
  echo "Deleting /var/log/sa/sa*"
  sudo rm -f /var/log/sa/sa*
fi
if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
  echo "Failed to delete /var/log/sa/sa*"
  exit 1
fi

if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
      echo "Deleting /var/lib/dhclient/dhclient*.lease"
      sudo rm -f /var/lib/dhclient/dhclient*.lease
fi
if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
      echo "Failed to delete /var/lib/dhclient/dhclient*.lease"
      exit 1
fi

if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
      echo "Deleting files within /var/tmp/*"
      sudo find /var/tmp -type f -exec rm -f {} \;
fi
if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
      echo "Failed to delete /var/tmp"
      exit 1
fi
if [[ $( sudo ls /var/tmp | sudo wc -l ) -gt 0 ]]; then
      echo "Deleting /var/tmp/*"
      sudo rm -rf /var/tmp/*
fi

# Shredding is not guaranteed to work well on rolling logs
```

```
if [[ -f "/var/lib/rsyslog/imjournal.state" ]]; then
      echo "Deleting /var/lib/rsyslog/imjournal.state"
      sudo rm -f /var/lib/rsyslog/imjournal.state
      sudo rm -f /var/lib/rsyslog/imjournal.state
fi

if [[ $( sudo ls /var/log/journal/ | sudo wc -l ) -gt 0 ]]; then
      echo "Deleting /var/log/journal/*"
      sudo find /var/log/journal/ -type f -exec rm -f {} \;
      sudo rm -rf /var/log/journal/*
fi

sudo touch /etc/machine-id
```

Linux 정리 스크립트를 오버라이드합니다.

Image Builder는 기본적으로 안전한 이미지를 생성하고 보안 모범 사례를 따릅니다. 하지만 일부 고급 사용 사례에서는 내장된 정리 스크립트의 한 개 이상의 섹션을 건너뛰어야 할 수도 있습니다. 일부 정리 단계를 건너뛰어야 하는 경우 출력 AMI를 테스트하여 이미지의 보안을 확인하는 것이 좋습니다.

> **Important**
>
> 정리 스크립트의 섹션을 건너뛰면 소유자 계정 세부 정보 또는 SSH 키와 같은 민감한 정보가 최종 이미지 및 해당 이미지에서 실행되는 인스턴스에 포함될 수 있습니다. 다른 가용 영역, 리전 또는 계정에서 시작하는 데 문제가 발생할 수도 있습니다.

다음 표에는 정리 스크립트의 섹션, 해당 섹션에서 삭제된 파일 및 Image Builder에서 건너뛰어야 하는 섹션에 플래그를 지정하는 데 사용할 수 있는 파일 이름이 요약되어 있습니다. 정리 스크립트의 특정 섹션을 건너뛰려면 CreateFile 구성 요소 작업 모듈이나 사용자 데이터의 명령(오버라이드하는 경우)을 사용하여 건너뛰기 섹션 파일 이름 열에 지정된 이름으로 빈 파일을 생성할 수 있습니다.

> **Note**
>
> 정리 스크립트의 한 부분을 건너뛰기 위해 생성하는 파일에는 파일 확장자가 포함되어서는 안 됩니다. 예를 들어, 스크립트의 `CLOUD_INIT_FILES` 섹션을 건너뛰고 싶지만 `skip_cleanup_cloudinit_files.txt`(이)라는 파일을 생성하면 Image Builder는 건너뛰기 파일을 인식하지 못합니다.

입력

섹션 정리하기	파일 제거됨	섹션 파일 이름 건너뛰기
CLOUD_INIT_FILES	/etc/sudoers.d/90-cloud-init-users /etc/locale.conf /var/log/cloud-init.log /var/log/cloud-init-output.log	skip_cleanup_cloud init_files
INSTANCE_FILES	/etc/.updated /etc/aliases.db /etc/hostname /var/lib/misc/postfix.aliasesdb-stamp /var/lib/postfix/master.lock /var/spool/postfix/pid/master.pid /var/.updated /var/cache/yum/x86_64/2/.gpgkeyschecked.yum	skip_cleanup_instance_files
SSH_FILES	/etc/ssh/ssh_host_rsa_key /etc/ssh/ssh_host_rsa_key.pub	skip_cleanup_ssh_files

섹션 정리하기	파일 제거됨	섹션 파일 이름 건너뛰기
	/etc/ssh/ssh_host_ecdsa_key /etc/ssh/ssh_host_ecdsa_key.pub /etc/ssh/ssh_host_ed25519_key /etc/ssh/ssh_host_ed25519_key.pub /root/.ssh/authorized_keys /home/<all users>/.ssh/authorized_keys;	
INSTANCE_LOG_FILES	/var/log/audit/audit.log /var/log/boot.log /var/log/dmesg /var/log/cron	skip_cleanup_instance_log_files
TOE_FILES	{{workingDirectory}}/TOE_*	skip_cleanup_toe_files
SSM_LOG_FILES	/var/log/amazon/ssm/*	skip_cleanup_ssm_log_files

Image Builder 문제 해결

EC2 Image Builder는 모니터링 및 문제 해결을 AWS 서비스 위해와 통합되어 이미지 빌드 문제를 해결하는 데 도움이 됩니다. Image Builder는 이미지 빌드 프로세스의 각 단계에 대한 진행 상황을 추적하고 표시합니다. 또한 Image Builder는 사용자가 제공한 Amazon S3 위치로 로그를 내보낼 수 있습니다.

고급 문제 해결을 위해 AWS Systems Manager 명령 실행을 사용하여 사전 정의된 명령과 스크립트를 실행할 수 있습니다.

내용

- 파이프라인 빌드 문제 해결
- 문제 해결 시나리오

파이프라인 빌드 문제 해결

Image Builder 파이프라인 빌드가 실패하는 경우 Image Builder는 실패를 설명하는 오류 메시지를 반환합니다. 또한 Image Builder는 다음 예제 출력의 경우와 같이 실패 메시지에서 `workflow execution ID`(을)를 반환합니다.

```
Workflow Execution ID: wf-12345abc-6789-0123-abc4-567890123abc failed with reason: …
```

Image Builder는 표준 이미지 생성 프로세스의 런타임 단계에 대해 정의된 일련의 단계를 통해 이미지 빌드 작업을 정렬하고 지시합니다. 각 프로세스의 빌드 및 테스트 단계에는 관련 워크플로우가 있습니다. Image Builder는 워크플로우를 실행하여 새 이미지를 빌드하거나 테스트할 때 런타임 세부 정보를 추적하는 워크플로우 메타데이터 리소스를 생성합니다.

컨테이너 이미지에는 배포 중에 실행되는 추가 워크플로우가 있습니다.

워크플로우의 런타임 인스턴스 실패에 대한 세부 정보를 조사하십시오.

워크플로우의 런타임 실패 문제를 해결하려면 `workflow execution ID`(을)를 사용하여 GetWorkflowExecution 및 ListWorkflowStepExecutions API 작업을 호출할 수 있습니다.

워크플로우 런타임 로그를 검토하세요.

- Amazon CloudWatch Logs

  Image Builder는 상세한 워크플로우 실행 로그를 다음 Image Builder CloudWatch Logs 그룹 및 스트림에 게시합니다.

LogGroup:

```
/aws/imagebuilder/ImageName
```

LogStream(x.x.x/x):

```
ImageVersion/ImageBuildVersion
```

CloudWatch Logs를 통해 필터 패턴으로 로그 데이터를 검색할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 필터 패턴을 사용하여 로그 데이터 검색을 참조하세요.

- AWS CloudTrail

  또한, 계정에서 활성화된 경우 모든 빌드 활동은 CloudTrail에 기록됩니다. 소스 `imagebuilder.amazonaws.com`별로 CloudTrail 이벤트를 필터링할 수 있습니다. 또는 실행 로그에서 반환되는 Amazon EC2 인스턴스 ID를 검색하여 파이프라인 실행에 대한 자세한 내용을 확인할 수 있습니다.

- Amazon Simple Storage Service(S3)

  인프라 구성에서 S3 버킷 이름과 키 접두사를 지정한 경우 워크플로우 단계 런타임 로그 경로는 다음 패턴을 따릅니다.

  ```
  S3://S3BucketName/KeyPrefix/ImageName/ImageVersion/ImageBuildVersion/
  WorkflowExecutionId/StepName
  ```

  S3 버킷으로 보내는 로그는 이미지 빌드 프로세스 중 EC2 인스턴스 활동에 대한 단계 및 오류 메시지를 보여줍니다. 로그에는 구성 요소 관리자의 로그 출력, 실행된 구성 요소의 정의, 인스턴스에서 수행한 모든 단계의 세부 출력(JSON)이 포함됩니다. 문제가 발생하면 먼저 `application.log`(을)를 시작으로 이러한 파일을 검토하여 인스턴스에서 문제의 원인을 진단해야 합니다.

기본적으로 Image Builder는 파이프라인에 장애가 발생하면 실행 중인 Amazon EC2 빌드 또는 테스트 인스턴스를 종료합니다. 파이프라인이 사용하는 인프라 구성 리소스의 인스턴스 설정을 변경하여 문제 해결을 위해 빌드 또는 테스트 인스턴스를 유지할 수 있습니다.

콘솔에서 인스턴스 설정을 변경하려면 인프라 구성 리소스의 설정 문제 해결 섹션에 있는 장애 시 인스턴스 종료 확인란의 선택을 취소해야 합니다.

AWS CLI의 update-infrastructure-configuration 명령을 사용하여 인스턴스 설정을 변경할 수도 있습니다. 명령이 `--cli-input-json` 파라미터와 함께 참조하는 JSON 파일에서 `terminateInstanceOnFailure` 값을 `false`(으)로 설정합니다. 자세한 내용은 인프라 구성 업데이트을 참조하세요.

문제 해결 시나리오

이 섹션에는 다음과 같은 세부 문제 해결 시나리오가 나열되어 있습니다.

- 액세스 거부 — 상태 코드 403
- 빌드 인스턴스에서 Systems Manager 에이전트 가용성을 확인하는 중 빌드 시간이 초과되었습니다.
- 시작 시 Windows 보조 디스크가 오프라인 상태입니다.
- CIS로 강화된 기본 이미지를 사용해 빌드 실패
- AssertInventoryCollection 실패(Systems Manager 자동화)

시나리오의 세부 정보를 보려면 시나리오 제목을 선택하여 확장하십시오. 동시에 여러 타이틀을 확장할 수 있습니다.

액세스 거부 — 상태 코드 403

설명

'AccessDenied: 액세스 거부 상태 코드: 403'과 함께 파이프라인 빌드가 실패합니다.

원인

가능한 원인은 다음과 같습니다.

- 인스턴스 프로파일에는 API 또는 구성 요소 리소스에 액세스하는 데 필요한 권한이 없습니다.
- 인스턴스 프로파일 역할에 Amazon S3에 로깅하는 데 필요한 권한이 없습니다. 가장 일반적으로 이는 인스턴스 프로파일 역할에 S3 버킷에 대한 PutObject 권한이 없을 때 발생합니다.

Solution

원인에 따라 이 문제는 다음과 같이 해결될 수 있습니다.

- 인스턴스 프로파일에 관리형 정책이 없습니다 — 누락된 정책을 인스턴스 프로파일 역할에 추가합니다. 그런 다음 파이프라인을 다시 실행하세요.

- 인스턴스 프로파일에 S3 버킷에 대한 쓰기 권한이 없습니다 — S3 버킷에 쓸 수 있는 PutObject 권한을 부여하는 정책을 인스턴스 프로파일 역할에 추가합니다. 그런 다음 파이프라인을 다시 실행하세요.

빌드 인스턴스에서 Systems Manager 에이전트 가용성을 확인하는 중 빌드 시간이 초과되었습니다.

설명

파이프라인 빌드가 '상태 = 'TimeOut' 및 '실패 메시지 = '단계가 대상 인스턴스에서 Systems Manager Agent 가용성을 확인하는 동안 단계 제한 시간이 초과되었습니다''와 함께 실패합니다.

원인

가능한 원인은 다음과 같습니다.

- 빌드 작업을 수행하고 구성 요소를 실행하기 위해 시작된 인스턴스가 Systems Manager 엔드포인트에 액세스할 수 없었습니다.
- 인스턴스 프로파일에 필요한 권한이 없습니다.

Solution

가능한 원인에 따라 이 문제는 다음과 같이 해결할 수 있습니다.

- 액세스 문제, 프라이빗 서브넷 — 프라이빗 서브넷을 구축하는 경우 Systems Manager, Image Builder에 대해 PrivateLink 엔드포인트를 설정하고, 로깅을 원할 경우 Amazon S3/CloudWatch에 대해 PrivateLink 엔드포인트를 설정했는지 확인하십시오. PrivateLink 엔드포인트 설정에 대한 자세한 내용은 를 통한 AWS 서비스 액세스를 AWS PrivateLink 참조하세요.
- 권한 누락 — Image Builder의 IAM 서비스 연결 역할에 다음과 같은 관리형 정책을 추가합니다.
  - EC2InstanceProfileForImageBuilder
  - EC2InstanceProfileForImageBuilderECRContainerBuilds
  - AmazonSSMManagedInstanceCore

  Image Builder 서비스 연결 역할에 대한 자세한 내용은 Image Builder의 IAM 서비스 연결 역할 사용 섹션을 참조하세요.

시작 시 Windows 보조 디스크가 오프라인 상태입니다.

설명

Image Builder Windows AMI를 빌드하는 데 사용된 인스턴스 유형이 AMI에서 시작하는 데 사용되는 인스턴스 유형과 일치하지 않는 경우, 시작 시 루트가 아닌 볼륨이 오프라인 상태인 문제가 발생할 수 있습니다. 이는 주로 빌드 인스턴스가 시작 인스턴스보다 새로운 아키텍처를 사용할 때 발생합니다.

다음 예제는 Image Builder AMI를 EC2 Nitro 인스턴스 유형에 구축하여 EC2 Xen 인스턴스에서 시작할 때 발생하는 상황을 보여줍니다.

빌드 인스턴스 유형: m5.large(Nitro)

시작 인스턴스 유형: t2.medium(Xen)

```
PS C:\Users\Administrator> get-disk
Number  Friendly Name  Serial Number         Health Status  Operational Status  Total
 Size  Partition Style
------  -------------  -------------         -------------  ------------------
 ----------  ---------------
0       AWS PVDISK     vol0abc12d34e567f8a9  Healthy        Online                  30
 GB  MBR
1       AWS PVDISK     vol1bcd23e45f678a9b0  Healthy        Offline                  8
 GB  MBR
```

원인

Windows 기본 설정으로 인해 새로 검색된 디스크는 자동으로 온라인 상태로 전환되거나 포맷되지 않습니다. EC2에서 인스턴스 유형이 변경되면 Windows는 이를 새 디스크가 검색된 것으로 간주합니다. 이는 기본 드라이버 변경 때문입니다.

Solution

Windows AMI를 빌드할 때는 시작하려는 것과 동일한 인스턴스 유형 시스템을 사용하는 것이 좋습니다. 서로 다른 시스템에 구축된 인스턴스 유형을 인프라 구성에 포함시키지 마십시오. 지정한 인스턴스 유형 중 Nitro 시스템을 사용하는 경우 모두 Nitro 시스템을 사용해야 합니다.

Nitro 시스템에 구축된 인스턴스에 대한 자세한 내용은 Amazon EC2 사용 설명서의 Nitro 시스템에 구축된 인스턴스 섹션을 참조하세요.

CIS로 강화된 기본 이미지를 사용해 빌드 실패

설명

CIS 강화 기본 이미지를 사용하고 있는데 빌드가 실패합니다.

원인

`/tmp` 디렉토리가 `noexec`(으)로 분류되면 Image Builder가 실패할 수 있습니다.

Solution

이미지 레시피 `workingDirectory` 필드에서 작업 디렉토리로 사용할 다른 위치를 선택합니다. 자세한 내용은 Image Recider 데이터 유형 설명을 참조하세요.

AssertInventoryCollection 실패(Systems Manager 자동화)

설명

Systems Manager 자동화는 `AssertInventoryCollection` 자동화 단계에서 실패를 표시합니다.

원인

사용자 또는 조직에서 EC2 인스턴스의 인벤토리 정보를 수집하는 Systems Manager State Manager 연결을 생성했을 수 있습니다. Image Builder 파이프라인에 향상된 이미지 메타데이터 수집이 활성화된 경우 (기본값) Image Builder는 빌드 인스턴스에 대한 새 인벤토리 연결을 생성하려고 시도합니다. 하지만 Systems Manager는 관리형 인스턴스에 대한 다중 인벤토리 연결을 허용하지 않으며, 이미 존재하는 경우 새 연결을 금지합니다. 이로 인해 작업이 실패하고 파이프라인 빌드가 실패합니다.

Solution

이 문제를 해결하려면 다음 방법 중 하나를 사용하여 향상된 Image Builder를 끄세요.

- 콘솔에서 이미지 파이프라인을 업데이트하여 향상된 메타데이터 수집 활성화 확인란의 선택을 취소하세요. 변경 내용 저장 및 파이프라 빌드 실행.

  EC2 Image Builder 콘솔을 사용한 AMI 이미지 파이프라인 업데이트에 대한 자세한 내용은 콘솔에서 AMI 이미지 파이프라인 업데이트(을)를 참조하세요. EC2 Image Builder 콘솔을 사용한 컨테이너 이미지 파이프라인 업데이트에 대한 자세한 내용은 콘솔에서 컨테이너 이미지 파이프라인 업데이트(을)를 참조하세요.
- AWS CLI의 update-image-pipeline 명령을 사용하여 이미지 파이프라인을 업데이트할 수도 있습니다. 이렇게 하려면 `false`(으)로 설정된 `EnhancedImageMetadataEnabled` 속성을 JSON 파일에 포함시키십시오. 다음 예제에서는 `false`(으)로 설정된 속성을 보여줍니다.

```
{
    "name": "MyWindows2019Pipeline",
    "description": "Builds Windows 2019 Images",
    "enhancedImageMetadataEnabled": false,
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-recipe/2020.12.03",
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-infrastructure-configuration",
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-distribution-configuration",
    "imageTestsConfiguration": {
        "imageTestsEnabled": true,
        "timeoutMinutes": 60
    },
    "schedule": {
        "scheduleExpression": "cron(0 0 * * SUN *)",
        "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
    },
    "status": "ENABLED"
}
```

새 파이프라인에서 이런 일이 발생하지 않도록 하려면 EC2 Image Builder 콘솔을 사용하여 새 파이프라인을 생성할 때 향상된 메타데이터 수집 활성화 확인란의 선택을 취소하거나 AWS CLI(을)를 사용하여 파이프라인을 생성할 때 JSON 파일의 `EnhancedImageMetadataEnabled` 속성 값을 `false`(으)로 설정하십시오.

Image Builder 사용 설명서 변경 사항의 문서 기록

다음 표에서는 설명서에서 중요한 변경 사항을 날짜별로 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- API 버전: 2025-04-30

변경 사항	설명	날짜
STIG 2025년 Q1 업데이트	Windows STIG 버전을 업데이트하고 2025년 1분기 릴리스에 STIGS 및 SCAP 구성 요소 버전을 적용했습니다.	2025년 5월 5일
기능 릴리스: SSM 파라미터 지원	Image Builder는 이제 레시피 및 이미지 배포 중에 AWS Systems Manager(SSM) 파라미터 스토어 파라미터 사용을 지원합니다.	2025년 4월 30일
STIG 4분기 업데이트	Windows STIG 버전을 업데이트하고 2024년 4분기 릴리스에 STIGS를 적용했습니다.	2025년 2월 4일
IAM 정책 업데이트: 서비스 역할 정책	가져온 공식 Microsoft 클라이언트 OS ISO 파일에서 이미지 생성을 지원하도록 서비스 연결 역할 정책을 업데이트했습니다. 자세한 내용은 AWSServiceRoleForImageBuilder 정책을 참조하세요.	2024년 12월 30일
IAM 정책 업데이트: 인스턴스 프로파일 정책	디스크 이미지 파일에서 이미지 생성을 지원하도록 인스턴스 프로파일 역할 정책을 업데이트했습니다. 자세한 내용은	2024년 12월 30일

	EC2InstanceProfileForImageBuilder 정책을 참조하세요.	
기능 릴리스: ISO에서 AMI로	Image Builder는 이제 Microsoft Windows 11 이상 클라이언트 운영 체제용 공식 ISO 디스크 파일에서 이미지를 생성할 수 있습니다.	2024년 12월 30일
STIG 4분기 업데이트	Linux STIG 버전을 업데이트하고 2024년 4분기 릴리스에 STIGS를 적용했습니다. Linux 구성 요소의 두 가지 새로운 입력 파라미터에 대한 정보가 추가되었습니다.	2024년 12월 10일
IAM 정책 업데이트: 인스턴스 프로파일 정책	AWS Marketplace 구성 요소를 가져올 수 있는 액세스 권한을 부여하도록 인스턴스 프로파일 정책을 업데이트했습니다. 자세한 내용은 EC2InstanceProfileForImageBuilder 정책을 참조하세요.	2024년 12월 2일
기능 릴리스: AWS Marketplace 구성 요소	AWS Marketplace 소프트웨어 구성 요소에 대한 지원이 추가되었습니다.	2024년 12월 2일
기능 릴리스: macOS 지원	macOS 이미지에 대한 지원이 추가되었습니다.	2024년 10월 22일
에서 논리 연산자에 대한 문서 지원 AWS Task Orchestrator and Executor	논리 연산자를 사용하여 구성 요소 문서에서 조건부 표현식을 추가하거나 수정합니다.	2024년 8월 16일

의 조건부 구문에 대한 문서 지원 AWS Task Orchestrator and Executor	"if" 문과 같은 조건부 구문을 사용하여 구성 요소 문서의 구성 요소 작업에 대한 제어 흐름을 지시합니다.	2024년 8월 16일
에서 비교 연산자에 대한 문서 지원 AWS Task Orchestrator and Executor	비교 연산자를 사용하여 구성 요소 문서의 값을 비교합니다.	2024년 8월 16일
Assert 작업 모듈 추가	`ExecuteDocument` 작업 모듈에 대한 문서가 `General execution` 에 추가되었습니다.	2024년 8월 14일
운영 체제 지원	RHEL 9 및 Ubuntu 24.04 LTS에 대한 지원이 추가되었습니다.	2024년 8월 2일
문서 업데이트: 콘텐츠 재구성	프레젠테이션 및 탐색을 개선하기 위해 문서를 재구성했습니다.	2024년 6월 21일
STIG 2분기 업데이트	Linux STIG 버전을 업데이트하고 2024년 2분기 릴리스에 STIG를 적용했습니다. RHEL 9, CentOS Stream 9 및 Ubuntu 22.04에 대한 지원이 추가되었습니다. Windows 버전에는 변경 사항이 없습니다.	2024년 5월 10일
STIG 1분기 업데이트	Linux STIG 버전을 업데이트하고 2024년 1분기 릴리스에 STIG를 적용했습니다. Windows 버전에는 변경 사항이 없습니다.	2024년 2월 23일

STIG 1분기 업데이트	Linux STIG 버전을 업데이트하고 2024년 1분기 릴리스에 STIG를 적용했습니다. Windows 버전에는 변경 사항이 없습니다.	2024년 2월 23일
기능 릴리스: 이미지 워크플로 관리	이미지 워크플로를 사용하면 이미지 생성 프로세스에서의 유연성, 가시성, 제어력이 향상됩니다. 워크플로의 빌드 및 테스트 단계를 사용자 지정하거나 Image Builder 기본 워크플로를 사용할 수 있습니다.	2023년 12월 12일
STIG 4분기 업데이트	Linux STIG 버전을 업데이트하고 2023년 4분기 릴리스에 STIG를 적용했습니다. Windows 버전에는 변경 사항이 없습니다. 또한 새 구성 요소, 소프트웨어 및 벤치마크 번호에 대해 Linux 및 Windows SCAP를 업데이트했습니다.	2023년 12월 7일
기능 릴리스: 이미지 수명 주기 관리	이미지 수명 주기 관리 정책 및 규칙을 사용하여 오래된 이미지와 관련 리소스가 태그 지정 및 제거 프로세스를 거치도록 리소스 관리 전략을 정의할 수 있습니다.	2023년 11월 17일
IAM 정책 업데이트: 서비스 역할	인스턴스 배치 지원을 위한 서비스 연결 역할 정책이 업데이트되었습니다. 자세한 내용은 AWSServiceRoleForImageBuilder 정책을 참조하세요.	2023년 10월 19일

STIG 3분기 업데이트	STIG 버전을 업데이트하고 2023년 3분기 릴리스에 STIG를 적용했습니다. 극히 일부 예외를 제외하고 타사 패키지가 자동으로 설치되지 않음을 명확히 하기 위해 메시지가 추가로 업데이트되었습니다. 건너뛰는 STIG는 모두 기록됩니다.	2023년 10월 5일
새 STIG 버전	STIG 버전을 업데이트하고 2023년 2분기 릴리스에 STIG를 적용했습니다.	2023년 5월 3일
새 STIG 버전	STIG 버전을 업데이트하고 2023년 1분기 릴리스에 STIG를 적용했습니다. AL2023에 대한 지원이 추가되었습니다.	2023년 4월 14일
에 대해 지원되는 리전 업데이트 AWSTOE	AWS 리전아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 유럽(취리히), 유럽(스페인), 중동(UAE)에 대한 AWSTOE 지원이 추가되었습니다.	2023년 4월 13일
AWSTOE 애플리케이션 다운로드 업데이트	Windows에서 AWSTOE 설치 다운로드에 대한 서명을 업데이트했습니다. 또한 TLS 업데이트된 S3 버킷에서 애플리케이션을 다운로드하려면 이제 TLS 버전 1.2 이상이 필요합니다.	2023년 3월 31일

기능 릴리스: 향상된 빌드 워크플로	이미지 빌드 버전 세부 정보의 새 워크플로 탭에 이미지 빌드의 런타임 세부 정보가 추가되었습니다. 빌드 문제 해결을 위한 정보가 개선되었습니다.	2023년 3월 30일
기능 릴리스: CVE 탐지 및 보고	Amazon Inspector 스캔을 활성화한 계정의 경우 Image Builder는 Amazon ECR에 저장된 컨테이너 이미지를 포함하여 새 이미지를 빌드하는 동안 테스트 단계 중에 Amazon Inspector에서 일반적인 취약성 및 노출(CVE) 결과를 캡처할 수 있습니다. Image Builder는 세부 분석을 지원하기 위해 결과의 스냅샷을 생성합니다. 또한 Image Builder는 계정, 파이프라인 또는 이미지별로 필터링할 수 있는 결과 수를 보고하며 세부 정보를 자세히 분석할 수 있습니다.	2023년 3월 30일
버전 이력 추가	Windows 및 Linux 섹션에 버전 이력을 추가했습니다.	2023년 2월 17일
새 STIG 버전	STIG 버전을 업데이트하고 2022년 4분기 릴리스에 STIG를 적용했습니다.	2023년 2월 1일

기능 릴리스: AWS Marketplace 통합 및 CIS 강화	CIS 강화 이미지와 인터넷 보안 센터의 새로운 CIS 강화 구성 요소를 포함하여 구독된 이미지를 새로운 사용자 지정 이미지의 기준으로 쉽게 찾고 사용할 수 있도록 AWS Marketplace 통합을 추가했습니다.	2023년 1월 13일
CIS 하드닝 구성 요소	CIS가 소유하고 유지 관리하는 CIS 하드닝 구성 요소가 추가되었습니다.	2023년 1월 13일
새 STIG 버전	Ubuntu 지원을 도입하고, STIG 버전을 업데이트하고, 2022년 2분기 릴리스에 STIGS를 적용했습니다.	2022년 7월 20일
문서 업데이트: YAML 구성 요소 문서 만들기 페이지 탐색	Create YAML 구성 요소 문서 콘텐츠를 자체 페이지로 이동하고 이를 참조하도록 다른 페이지를 업데이트했습니다.	2022년 6월 7일
새 STIG 버전	STIG 버전을 업데이트하고 2022년 1분기 릴리스에 STIG를 적용했습니다.	2022년 4월 25일
ExecuteDocument 작업 모듈이 추가됨	`ExecuteDocument` 작업 모듈에 대한 문서가 `General execution` 에 추가되었습니다.	2022년 3월 28일
기능 릴리스: Windows AMI 실행 속도 향상 지원	Windows AMI에 대한 더 빠른 시작을 사용할 수 있는 배포 구성 설정을 추가했습니다.	2022년 2월 21일

유지 관리 릴리스: AWSTOE 이진 지문 업데이트	AWSTOE 서명자 인증서에 대한 이진 지문을 업데이트했습니다.	2022년 2월 18일
기능 릴리스:에 대한 입력 구성 AWSTOE	JSON 구성 파일을 AWSTOE run 명령의 입력으로 사용하기 위한 지원이 추가되었습니다.	2022년 2월 3일
새 STIG 버전	STIG 버전을 업데이트하고 2021년 4분기 릴리스에 STIG를 적용했습니다. 새로운 SCAP 규정 준수 검사기(SCC) 구성 요소에 대한 섹션도 추가되었습니다.	2022년 12월 22일
기능 릴리스: VM Import/Export(VMIE) 통합	모든 채널(콘솔, API/CLI 등)을 통한 VM 가져오기 및 API/CLI를 통한 VM 내보내기에 대한 지원이 추가되었습니다. VM 내보내기는 현재 Image Builder 콘솔에서 사용할 수 없습니다.	2021년 12월 20일
기능 릴리스: AWS Organizations 및 OUs 대한 AMI 공유	AWS Organizations 및 OU와 출력 AMIs 공유에 대한 지원을 추가하도록 배포 구성을 업데이트했습니다. OUs	2021년 11월 24일
문서 업데이트: 구성 요소 단계 및 단계 업데이트	Image Builder의 구성 요소 단계에 대한 콘텐츠와 이러한 콘텐츠가 AWSTOE 구성 요소 단계와 상호 작용하는 방식을 확장했습니다.	2021년 9월 22일
문서 업데이트: CloudTrail 통합 콘텐츠 추가	모니터링 요약 및 CloudTrail 통합 콘텐츠가 추가되었습니다.	2021년 9월 17일

새 STIG 버전	STIG 버전을 업데이트하고 2021년 3분기 릴리스에 STIG를 적용했습니다.	2021년 9월 10일
기능 릴리스: Amazon EventBridge 통합	Image Builder를 AWS 서비스관련 이벤트와 연결하고 EventBridge에서 정의된 규칙에 따라 이벤트를 시작할 수 있는 EventBridge 지원이 추가되었습니다.	2021년 8월 18일
문서 업데이트: AWSTOE 페이지 재정렬	명확성을 위해 AWSTOE 페이지를 재배열했습니다.	2021년 8월 11일
기능 릴리스: 파라미터화된 구성 요소 및 추가 인스턴스 구성	레시피의 구성 요소를 사용자 지정하기 위한 파라미터 지정 지원이 추가되었습니다. 시작 시 실행할 명령을 지정하는 기능과 Systems Manager 에이전트의 설치 및 제거에 대한 제어를 강화하는 등 이미지 구축 및 테스트에 사용되는 EC2 인스턴스의 구성이 확장되었습니다.	2021년 7월 7일
새 STIG 버전	STIG 버전을 업데이트하고 2021년 2분기 릴리스에 STIG를 적용했습니다.	2021년 6월 30일
개선: 태깅 개선	리소스 태깅과 관련된 메시지를 개선했습니다.	2021년 6월 25일
기능 릴리스: 템플릿 통합 시작	배포 설정에 AMI 배포용 Amazon EC2 시작 템플릿 사용에 대한 지원이 추가되었습니다.	2021년 4월 7일

기능 릴리스: 컨테이너 빌드 개선	블록 디바이스 매핑을 구성하고 컨테이너 빌드의 기본 이미지로 사용할 AMI를 지정하는 데 대한 지원이 추가되었습니다.	2021년 4월 7일
새 STIG 버전	STIG 버전을 업데이트하고 STIG를 적용했습니다.	2021년 3월 5일
cron 표현식 업데이트	Image Builder cron 처리가 업데이트되어 cron 표현식의 세분성을 분 단위로 높이고 표준 cron 스케줄링 엔진을 사용합니다. 예제가 새 형식으로 업데이트되었습니다.	2021년 2월 8일
기능 릴리스: 컨테이너 지원	Image Builder를 사용하여 Docker 컨테이너 이미지를 만들 수 있는 지원이 추가되었으며 Amazon Elastic Container Registry(Amazon ECR)에서 결과 이미지를 등록 및 저장할 수 있습니다. 콘텐츠는 새로운 기능을 반영하고 향후 성장에 맞춰 재배치되었습니다.	2020년 12월 17일
재구성된 cron 문서	이 페이지는 이제 cron이 Image Builder 파이프라인 빌드와 함께 작동하는 방식에 대한 자세한 정보를 강조하고 UTC 시간에 대한 세부 정보를 포함합니다. 특정 필드에 허용되지 않는 와일드카드가 제거되었습니다. 이제 콘솔과 CLI에 대한 표현식 샘플이 예제에 포함됩니다.	2020년 11월 13일

콘솔 버전 2.0: 업데이트된 파이프라인 편집	시작하기 및 파이프라인 생성 튜토리얼과 새로운 콘솔 기능 및 흐름을 통합하기 위한 이미지 파이프라인 관리 페이지의 내용이 변경되었습니다.	2020년 11월 13일
새 STIG 버전	STIG 버전을 업데이트하고 STIG를 적용했습니다. 참고 - 목록 형식이 기본적으로 적용되는 STIG를 표시하도록 변경되었습니다.	2020년 10월 15일
에서 구문 반복 지원 AWSTOE	루프 구문을 만들어 AWSTOE 애플리케이션에서 반복되는 명령 시퀀스를 정의합니다.	2020년 7월 29일
AWSTOE 구성 요소의 로컬 개발 지원	AWSTOE 애플리케이션을 사용하여 로컬에서 이미지 구성 요소를 개발하고 테스트합니다.	2020년 7월 28일
암호화된 AMI	EC2 Image Builder는 암호화된 AMI 배포에 대한 지원을 추가합니다.	2020년 7월 1일
AutoScaling 사용 중단	인스턴스 시작을 위한 AutoScaling 사용이 중단되었습니다.	2020년 6월 15일

AWS PrivateLink를 통한 연결 지원	인터페이스 VPC 엔드포인트를 생성하여 VPC와 EC2 Image Builder 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 Image Builder APIs에 비공개로 액세스할 수 있는 기술인 AWS PrivateLink로 구동됩니다. VPC의 인스턴스는 Image Builder API와 통신하는 데 퍼블릭 IP 주소를 필요로 하지 않습니다. VPC와 Image Builder 간 트래픽은 Amazon 네트워크를 벗어나지 않습니다.	2020년 6월 10일
새 STIG 버전	STIG 버전을 업데이트하고 STIG를 적용했습니다.	2020년 1월 23일
문제 해결	일반 문제 해결 시나리오가 추가되었습니다.	2020년 1월 22일
STIG 구성 요소	STIG 구성 요소를 사용하여 AWSTOE STIG 호환 이미지를 생성할 수 있습니다.	2020년 1월 22일

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