



사용자 가이드

# Amazon ECR



API 버전 2015-09-21

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

# Amazon ECR: 사용자 가이드

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

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

# Table of Contents

Amazon ECR이란 무엇입니까 .....	1
Amazon ECR의 구성 요소 .....	1
Amazon ECR의 기능 .....	2
Amazon ECR 시작 방법 .....	3
Amazon ECR 가격 .....	3
수명 주기 전반에 걸친 이미지 이동 .....	4
필수 조건 .....	4
설치 AWS CLI .....	4
Docker 설치 .....	4
1단계: 도커 이미지 생성 .....	5
2단계: 기본 레지스트리에 대해 인증 .....	8
3단계: 리포지토리 생성 .....	8
4단계: Amazon ECR에 이미지 푸시 .....	9
5단계: Amazon ECR에서 이미지 가져오기 .....	10
6단계: 이미지 삭제 .....	11
7단계: 리포지토리 삭제 .....	11
성능 최적화 .....	12
프라이빗 레지스트리 .....	14
레지스트리 개념 .....	14
레지스트리 인증 .....	14
Amazon ECR 자격 증명 헬퍼 사용 .....	15
권한 부여 토큰 사용 .....	15
HTTP API 인증 사용 .....	16
레지스트리 설정 .....	17
레지스트리 권한 .....	17
레지스트리 정책 예제 .....	18
계정 간 복제를 위한 권한 부여 .....	21
풀스루 캐시에 대한 권한 부여 .....	22
프라이빗 리포지토리 .....	24
리포지토리 개념 .....	24
이미지를 저장할 리포지토리 생성 .....	25
다음 단계 .....	26
리포지토리 세부 정보 보기 .....	26
리포지토리 삭제 .....	27

리포지토리 정책 .....	28
리포지토리 정책과 IAM 정책 비교 .....	28
리포지토리 정책 예제 .....	30
리포지토리 정책 설명 설정 .....	35
리포지토리 태그 지정 .....	36
태그 기본 사항 .....	36
리소스에 결제용 태그 지정 .....	37
태그 추가 .....	37
태그 삭제 .....	39
프라이빗 이미지 .....	40
이미지 푸시 .....	40
필수 IAM 권한 .....	41
Docker 이미지 푸시하기 .....	42
다중 아키텍처 이미지 푸시 .....	44
Helm 차트 푸시 .....	45
이미지 서명 .....	48
고려 사항 .....	48
필수 조건 .....	48
Notary 클라이언트에 대한 인증 구성 .....	48
이미지 서명 .....	49
다음 단계 .....	50
서명 삭제 .....	50
이미지 세부 정보 보기 .....	51
이미지 가져오기 .....	51
아마존 리눅스 컨테이너 이미지 가져오기 .....	53
이미지 삭제 .....	54
이미지에 태그 다시 지정 .....	56
이미지 태그의 덮어쓰기 방지 .....	58
이미지 태그 변경 가능성 설정 (AWS Management Console) .....	58
이미지 태그 가변성 설정 (AWS CLI) .....	59
컨테이너 이미지 매니페스트 형식 .....	60
Amazon ECR 이미지 매니페스트 전환 .....	60
Amazon ECS에서 Amazon ECR 이미지 사용 .....	61
필수 IAM 권한 .....	62
태스크 정의에서 Amazon ECR 이미지 지정 .....	63
Amazon EKS에서 Amazon ECR 이미지 사용 .....	64

필수 IAM 권한 .....	64
Amazon EKS 클러스터에 헬름 차트 설치 .....	65
이미지에 취약성이 있는지 스캔합니다. ....	67
리포지토리 필터 .....	68
와일드카드 필터링 .....	68
고급 스캔 .....	69
고급 스캔 고려 사항 .....	69
필수 IAM 권한 .....	70
향상된 스캔 구성 .....	71
고급 스캔 기간 변경 .....	73
EventBridge 이벤트 .....	74
결과 검색 .....	79
기본 스캔 .....	80
향상된 기본 스캔을 위한 지역 지원 .....	81
기본 스캔 및 향상된 기본 스캔을 위한 운영 체제 지원 .....	82
향상된 기본 스캔 구성 .....	83
기본 스캔 구성 .....	84
수동으로 이미지 스캔 .....	84
결과 검색 .....	85
이미지 스캔 문제 해결 .....	87
스캔 상태 SCAN_ELIGIBILITY_EXPIRED 이해 .....	88
업스트림 레지스트리 동기화 .....	89
리포지토리 생성 템플릿 .....	89
폴스루 캐시 규칙 사용 시 고려 사항 .....	90
필수 IAM 권한 .....	91
레지스트리 권한 사용 .....	92
다음 단계 .....	94
폴스루 캐시 규칙 생성 .....	94
필수 조건 .....	94
사용: AWS Management Console .....	95
사용 AWS CLI .....	100
다음 단계 .....	103
리포지토리 생성 템플릿 .....	103
작동 방식 .....	104
필수 IAM 권한 .....	106
리포지토리 생성 템플릿 만들기 .....	107

리포지토리 생성 템플릿 삭제 .....	109
풀스루 캐시 규칙 검증 .....	109
풀스루 캐시 규칙으로 이미지 가져오기 .....	110
업스트림 리포지토리 보안 인증 정보 저장 .....	112
풀스루 캐시 문제 해결 .....	119
이미지 복제 .....	121
프라이빗 이미지 복제 관련 고려 사항 .....	121
복제 예제 .....	122
예제: 단일 대상 리전에 대한 교차 리전 복제 구성 .....	123
예제: 리포지토리 필터를 사용하여 교차 리전 복제 구성 .....	123
예제: 단일 대상 리전에 대한 교차 리전 복제 구성 .....	124
예제: 교차 계정 복제 구성 .....	124
예제: 구성에서 여러 규칙 지정 .....	125
복제 구성 .....	125
이미지 정리 자동화 .....	128
수명 주기 정책의 작동 방식 .....	128
수명 주기 정책 평가 규칙 .....	129
수명 주기 정책 미리 보기 생성 .....	130
수명 주기 정책 생성 .....	132
전제 조건 .....	132
수명 주기 정책의 예제 .....	133
수명 주기 정책 템플릿 .....	134
이미지 수명으로 필터링 .....	134
이미지 수로 필터링 .....	135
복수의 규칙으로 필터링 .....	135
단일 규칙에서 복수의 태그로 필터링 .....	138
모든 이미지 필터링 .....	140
수명 주기 정책 속성 .....	143
규칙 우선 순위 .....	143
설명 .....	143
태그 상태 .....	144
태그 패턴 목록 .....	144
태그 접두사 목록 .....	144
카운트 유형 .....	145
카운트 단위 .....	145
카운트 수 .....	146

작업 .....	146
보안 .....	147
ID 및 액세스 관리 .....	147
고객 .....	148
ID를 통한 인증 .....	149
정책을 사용한 액세스 관리 .....	151
Amazon Elastic Container Registry가 IAM과 작동하는 방식 .....	153
자격 증명 기반 정책 예시 .....	158
태그 기반 액세스 제어 사용 .....	163
AWS Amazon ECR에 대한 관리형 정책 .....	164
서비스 링크 역할 사용 .....	172
문제 해결 .....	178
데이터 보호 .....	180
저장된 데이터 암호화 .....	181
규정 준수 확인 .....	188
인프라 보안 .....	189
인터페이스 VPC 엔드포인트(AWS PrivateLink) .....	189
교차 서비스 혼동된 대리인 방지 .....	198
모니터링 .....	200
Service Quotas 시각화 및 경보 설정 .....	201
사용량 지표 .....	202
사용 보고서 .....	203
리포지토리 지표 .....	203
CloudWatch 지표 활성화 .....	204
사용 가능한 지표 및 차원 .....	204
다음과 같은 지표 보기 CloudWatch .....	205
이벤트 및 EventBridge .....	205
Amazon ECR의 샘플 이벤트 .....	205
을 사용하여 AWS CloudTrail작업 로깅 .....	209
아마존 ECR 정보는 CloudTrail .....	210
Amazon ECR 로그 파일 항목 이해 .....	211
SDK를 사용한 작업 AWS .....	222
코드 예시 .....	224
작업 .....	224
DescribeRepositories .....	224
ListImages .....	226

서비스 할당량 .....	229
AWS Management Console에서 Amazon ECR 서비스 할당량 관리 .....	234
API 사용량 지표를 모니터링하기 위한 CloudWatch 경보 생성 .....	235
문제 해결 .....	236
도커 문제 해결 .....	236
Docker 로그에는 예상 오류 메시지가 포함되지 않습니다. ....	236
Amazon ECR 리포지토리로부터 이미지를 가져올 때 오류: "Filesystem Verification Failed" 또는 "404: Image Not Found" .....	236
Amazon ECR에서 이미지를 가져올 때 오류: "Filesystem Layer Verification Failed" 발생 .....	237
리포지토리에 푸시할 때 HTTP 403 오류 또는 "no basic auth credentials" 오류 발생 .....	238
Amazon ECR 오류 메시지 문제 해결 .....	239
HTTP 429: 요청이 너무 많거나 ThrottleException .....	239
HTTP 403: "User [arn] is not authorized to perform [operation]" .....	239
HTTP 404: "Repository Does Not Exist" 오류 발생 .....	240
오류: TTY가 아닌 장치에서 대화형 로그인을 수행할 수 없습니다. ....	240
문서 기록 .....	241
.....	ccxlv



# Amazon Elastic Container Registry란 무엇입니까?

Amazon Elastic 컨테이너 레지스트리 (Amazon ECR) 는 AWS 안전하고 확장 가능하며 신뢰할 수 있는 관리형 컨테이너 이미지 레지스트리 서비스입니다. Amazon ECR은 IAM을 사용하여 리소스 기반 권한을 가진 프라이빗 리포지토리를 지원합니다. AWS 따라서 지정된 사용자 또는 Amazon EC2 인스턴스가 컨테이너 리포지토리 및 이미지에 액세스할 수 있습니다. 원하는 CLI를 사용하여 도커 이미지, Open Container Initiative(OCI) 이미지 및 OCI 호환 아티팩트를 푸시, 풀 및 관리할 수 있습니다.

## Note

Amazon ECR은 퍼블릭 컨테이너 이미지 리포지토리도 지원합니다. 자세한 내용은 Amazon ECR 퍼블릭 사용 설명서의 [Amazon ECR 퍼블릭은 무엇인가요](#)를 참조하세요.

AWS 컨테이너 서비스 팀은 에 대한 공개 로드맵을 유지 관리합니다. GitHub 여기에는 팀이 진행 중인 작업에 대한 정보가 포함되어 있으며 모든 AWS 고객이 직접 피드백을 제공할 수 있습니다. 자세한 내용은 [AWS 컨테이너 로드맵](#)을 참조하세요.

## Amazon ECR의 구성 요소

Amazon ECR은 다음 구성 요소를 포함합니다.

### 레지스트리

Amazon ECR 사설 레지스트리가 각 AWS 계정에 제공됩니다. 레지스트리에 하나 이상의 리포지토리를 생성하고 이 리포지토리에 Docker 이미지, OCI (오픈 컨테이너 이니셔티브) 이미지 및 OCI 호환 아티팩트를 저장할 수 있습니다. 자세한 정보는 [Amazon ECR 프라이빗 레지스트리](#)을 참조하세요.

### 사용자 권한 토큰

클라이언트는 Amazon ECR 프라이빗 레지스트리에 AWS 사용자로서 인증을 해야 이미지를 푸시하고 가져올 수 있습니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.

### 리포지토리

Amazon ECR 리포지토리에는 Docker 이미지, Open Container Initiative(OCI) 이미지 및 OCI 호환 아티팩트가 포함되어 있습니다. 자세한 정보는 [Amazon ECR 프라이빗 리포지토리](#)을 참조하세요.

## 리포지토리 정책

리포지토리 정책을 사용하면 리포지토리 및 리포지토리 내 콘텐츠에 대한 액세스를 제어할 수 있습니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리 정책](#)을 참조하세요.

## 이미지

리포지토리에 컨테이너 이미지를 푸시하고 가져올 수 있습니다. 개발 시스템에서 로컬로 이러한 이미지를 사용하거나, Amazon ECS 태스크 정의 및 Amazon EKS 포드 사양에서 이를 사용할 수 있습니다. 자세한 정보는 [Amazon ECS에서 Amazon ECR 이미지 사용](#) 및 [Amazon EKS에서 Amazon ECR 이미지 사용](#) 단원을 참조하세요.

## Amazon ECR의 기능

Amazon ECR은 다음의 기능을 제공합니다.

- 수명 주기 정책은 리포지토리에 있는 이미지의 수명 주기를 관리하는 데 도움이 됩니다. 사용되지 않는 이미지를 정리하는 규칙을 정의합니다. 규칙을 리포지토리에 적용하기 전에 테스트할 수 있습니다. 자세한 정보는 [Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.](#)을 참조하세요.
- 이미지 스캔은 컨테이너 이미지의 소프트웨어 취약성을 식별하는 데 도움이 됩니다. 각 리포지토리는 푸시 시 스캔하도록 구성할 수 있습니다. 이렇게 하면 리포지토리로 푸시된 각각의 새 이미지가 스캔됩니다. 그런 다음 이미지 스캔 결과를 검색할 수 있습니다. 자세한 정보는 [이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다.](#)을 참조하세요.
- 교차 리전 및 교차 계정 복제를 통해 이미지를 필요한 곳에 쉽게 배치할 수 있습니다. 이는 레지스트리 설정으로 구성되며 리전별 단위로 구성됩니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 설정](#)을 참조하세요.
- 풀스루 캐시 규칙은 프라이빗 Amazon ECR 레지스트리의 업스트림 레지스트리에서 리포지토리를 캐시하는 방법을 제공합니다. Amazon ECR은 풀스루 캐시 규칙을 사용하여 정기적으로 업스트림 레지스트리에 연결하여 Amazon ECR 프라이빗 레지스트리의 캐시된 이미지가 최신 상태인지 확인합니다. 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화](#)을 참조하세요.

## Amazon ECR 시작 방법

Amazon Elastic Container Service (Amazon ECS) 또는 Amazon Elastic Kubernetes Service (Amazon EKS) 를 사용하는 경우, Amazon ECR은 두 서비스의 확장이므로 이 두 서비스의 설정은 Amazon ECR의 설정과 비슷하다는 점에 유의하십시오.

Amazon AWS Command Line Interface ECR과 함께 사용할 경우 최신 Amazon ECR 기능을 AWS CLI 지원하는 버전을 사용하십시오. 에서 Amazon ECR 기능에 대한 지원이 표시되지 않는 경우 최신 버전으로 업그레이드하십시오 AWS CLI. AWS CLI 최신 버전 설치에 대한 자세한 내용은 AWS Command Line Interface 사용 [설명서의 최신 버전 설치 또는 업데이트](#)를 참조하십시오. AWS CLI AWS CLI

AWS CLI 및 Docker를 사용하여 컨테이너 이미지를 프라이빗 Amazon ECR 리포지토리로 푸시하는 방법을 알아보려면 [을 참조하십시오. Amazon ECR에서 수명 주기 전반에 걸쳐 이미지 이동](#)

## Amazon ECR 가격

Amazon ECR을 사용하면 리포지토리에 저장한 데이터의 양과 이미지 푸시 및 풀에서 전송한 데이터 양에 대해서만 비용을 지불하면 됩니다. 자세한 정보는 [Amazon ECR 요금](#)을 참조하세요.

# Amazon ECR에서 수명 주기 전반에 걸쳐 이미지 이동

Amazon ECR을 처음 사용하는 경우 Docker CLI와 함께 다음 단계를 사용하여 샘플 이미지를 생성하고, 기본 레지스트리에 인증하고, 사설 리포지토리를 생성하십시오. AWS CLI 그런 다음 이미지를 프라이빗 리포지토리로 푸시하고 프라이빗 리포지토리에서 이미지를 가져옵니다. 샘플 이미지 작성이 끝나면 샘플 이미지와 리포지토리를 삭제합니다.

를 AWS Management Console 대신 사용하려면 AWS CLI을 참조하십시오 [the section called “이미지를 저장할 리포지토리 생성”](#).

[다양한 AWS SDK, IDE 툴킷, Windows PowerShell 명령줄 도구를 포함하여 AWS 리소스를 관리하는데 사용할 수 있는 다른 도구에 대한 자세한 내용은 <http://aws.amazon.com/tools/> 을 참조하십시오.](#)

## 필수 조건

최신 버전의 Docker가 설치되어 AWS CLI 있고 사용할 준비가 되지 않은 경우 다음 단계에 따라 두 도구를 모두 설치하십시오.

## 설치 AWS CLI

Amazon AWS CLI ECR과 함께 사용하려면 최신 AWS CLI 버전을 설치하십시오. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 [AWS Command Line Interface 설치](#)를 참조하세요.

## Docker 설치

Docker는 최신 Linux 배포 버전(Ubuntu 등)을 비롯하여 MacOS 및 Windows 등 다양한 운영 체제에서 사용할 수 있습니다. 특정 운영 체제에 Docker를 설치하는 방법에 대한 자세한 내용은 [Docker 설치 안내서](#)를 참조하십시오.

Docker를 사용하기 위해 로컬 개발 시스템이 필요하지 않습니다. Amazon EC2를 이미 사용 중인 경우 Amazon Linux 2023 인스턴스를 시작하고 Docker를 설치하여 시작할 수 있습니다.

이미 Docker가 설치되어 있으면 [1단계: 도커 이미지 생성](#) 단계로 건너뛸니다.

Amazon Linux 2023 AMI를 사용하는 Amazon EC2 인스턴스에 Docker를 설치하려면

1. 최신 Amazon Linux 2023 AMI에서 인스턴스를 시작합니다. 자세한 내용은 Amazon EC2 사용 설명서의 [인스턴스 시작](#)을 참조하십시오.
2. 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 [Linux 인스턴스에 연결](#)을 참조하십시오.

- 인스턴스에 설치한 패키지 및 패키지 캐시를 업데이트합니다.

```
sudo yum update -y
```

- 최신 Docker Community Edition 패키지를 설치합니다.

```
sudo yum install docker
```

- Docker 서비스를 시작합니다.

```
sudo service docker start
```

- sudo를 사용하지 않고도 Docker 명령을 실행할 수 있도록 docker 그룹에 ec2-user를 추가합니다.

```
sudo usermod -a -G docker ec2-user
```

- 로그아웃하고 다시 로그인해서 새 docker 그룹 권한을 선택합니다. 이를 위해 현재 SSH 터미널 창을 닫고 새 창에서 인스턴스를 다시 연결할 수 있습니다. 새 SSH 세션은 해당되는 docker 그룹 권한을 갖게 됩니다.
- sudo 없이도 ec2-user가 Docker 명령을 실행할 수 있는지 확인합니다.

```
docker info
```

#### Note

경우에 따라서는 ec2-user가 Docker 데몬에 액세스할 수 있는 권한을 제공하기 위해 인스턴스를 재부팅해야 할 수도 있습니다. 다음 오류가 표시될 경우 인스턴스를 재부팅합니다.

```
Cannot connect to the Docker daemon. Is the docker daemon running on this host?
```

## 1단계: 도커 이미지 생성

이 단계에서는 간단한 웹 애플리케이션의 도커 이미지를 생성하여 이를 로컬 시스템이나 Amazon EC2 인스턴스에서 테스트합니다.

## 간단한 웹 애플리케이션의 Docker 이미지를 생성하려면

1. Dockerfile이라는 파일을 생성합니다. Dockerfile은 Docker 이미지에 사용할 기본 이미지 및 이를 설치하고 실행할 항목을 설명하는 매니페스트입니다. Dockerfile에 대한 자세한 내용은 [Dockerfile 참조](#)를 참조하세요.

### touch Dockerfile

2. 방금 만든 Dockerfile을 수정하고 다음 내용을 추가합니다.

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

# Install dependencies
RUN yum update -y && \
    yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
    echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
    echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
    chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

이 Dockerfile은 Amazon ECR 퍼블릭에서 호스팅되는 퍼블릭 Amazon Linux 2 이미지를 사용합니다. RUN 지침은 패키지 캐시를 업데이트하고, 웹 서버의 일부 소프트웨어 패키지를 설치하고, 'Hello World!'를 작성합니다. 콘텐츠를 웹 서버 문서 루트에 작성합니다. EXPOSE 지침은 컨테이너에 포트 80을 노출하고 CMD 지침은 웹 서버를 시작합니다.

3. Dockerfile에서 Docker 이미지를 빌드합니다.

### Note

아래의 명령에서 Docker의 일부 버전에서는 아래 보이는 상대 경로 대신에 Dockerfile의 전체 경로가 필요할 수 있습니다.

```
docker build -t hello-world .
```

4. 컨테이너 이미지를 나열합니다.

```
docker images --filter reference=hello-world
```

출력:

REPOSITORY SIZE	TAG	IMAGE ID	CREATED
hello-world 194MB	latest	e9ffedc8c286	4 minutes ago

5. 새로 빌드된 이미지를 실행합니다. `-p 80:80` 옵션은 컨테이너에 있는 노출된 포트 80을 호스트 시스템에 있는 포트 80에 매핑합니다. `docker run`에 대한 자세한 내용을 보려면 [Docker 실행 참조](#)를 참조하세요.

```
docker run -t -i -p 80:80 hello-world
```

#### Note

Apache 웹 서버로부터의 출력이 터미널 창에 표시됩니다. "Could not reliably determine the fully qualified domain name" 메시지는 무시해도 됩니다.

6. 브라우저를 열고 Docker를 실행하고 컨테이너를 호스팅하고 있는 서버를 가리킵니다.
- EC2 인스턴스를 사용하고 있는 경우 서버의 Public DNS 값이며, 이는 SSH로 인스턴스에 연결할 때 사용하는 주소와 동일합니다. 인스턴스의 보안 그룹에서 포트 80에 인바운드 트래픽을 허용해야 합니다.
  - Docker를 로컬에서 실행하고 있는 경우, 브라우저에서 <http://localhost/>를 가리킵니다.
  - Windows 또는 Mac 컴퓨터에서 사용하는 경우 `docker-machine ip` 명령을 사용하여 `docker-machine` Docker를 호스팅하는 VirtualBox VM의 IP 주소를 찾고, `### ## ## ## ###` `####` 대체합니다.

```
docker-machine ip machine-name
```

"Hello, World!" 문이 있는 웹 페이지가 표시됩니다.

7. Ctrl + c를 입력하여 Docker 컨테이너를 중지합니다.

## 2단계: 기본 레지스트리에 대해 인증

를 설치하고 구성한 후 Docker CLI를 기본 레지스트리에 인증합니다. AWS CLI이렇게 하면 docker 명령이 Amazon ECR을 사용하여 이미지를 푸시하고 가져올 수 있습니다. 는 인증 프로세스를 단순화하는 get-login-password 명령을 AWS CLI 제공합니다.

를 사용하여 Amazon ECR 레지스트리에 Docker를 get-login-password 인증하려면 명령을 실행합니다. aws ecr get-login-password 인증 토큰을 docker login 명령에 전달할 때 사용자 이름으로 AWS 값을 사용하고, 인증하려는 Amazon ECR 레지스트리 URI를 지정합니다. 여러 레지스트리에 대해 인증하는 경우 각 레지스트리에 대해 명령을 반복해야 합니다.

### Important

오류가 발생하면 최신 버전의 AWS CLI를 설치하거나 업그레이드합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 [AWS Command Line Interface 설치](#)를 참조하세요.

- [get-login-password](#)(AWS CLI)

```
aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

- [get-ECR \(\) LoginCommand](#)AWS Tools for Windows PowerShell

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

## 3단계: 리포지토리 생성

Amazon ECR에 푸시할 이미지가 준비되었으면 이를 보유할 리포지토리를 생성해야 합니다. 이 예에서는 hello-repository라는 리포지토리를 생성합니다. 나중에 이 리포지토리에 hello-world:latest 이미지를 푸시하게 됩니다. 리포지토리를 생성하려면 다음 명령을 실행합니다.



```
aws ecr create-repository \
  --repository-name hello-repository \
  --region region
```

## 4단계: Amazon ECR에 이미지 푸시

이제 이전 섹션에서 생성한 Amazon ECR 리포지토리에 이미지를 푸시할 수 있습니다. 다음 사전 요구 사항이 충족된 후 docker CLI를 사용하여 이미지를 푸시하십시오.

- 최소 버전 설치: docker 1.7.
- Amazon ECR 인증 토큰은 로 docker login 구성되었습니다.
- Amazon ECR 리포지토리가 있으며 사용자에게 리포지토리를 푸시할 수 있는 액세스 권한이 있습니다.

이러한 사전 조건이 만족되면, 계정의 기본 레지스트리에 있는 새롭게 생성된 리포지토리에 이미지를 푸시할 수 있습니다.

이미지에 태그를 지정하고 Amazon ECR에 푸시하려면

1. 태그를 지정하고 푸시할 이미지를 식별할 수 있도록 로컬에 저장한 이미지를 나열합니다.

```
docker images
```

출력:

REPOSITORY	TAG	IMAGE ID	CREATED
VIRTUAL SIZE			
hello-world	latest	e9ffedc8c286	4 minutes ago
241MB			

2. 리포지토리에 푸시할 이미지에 태그를 지정합니다.

```
docker tag hello-world:latest aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

3. 이미지를 푸시합니다.

```
docker push aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository
```

**출력:**

```
The push refers to a repository [aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository] (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
latest: digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
size: 6774
```

## 5단계: Amazon ECR에서 이미지 가져오기

이미지가 Amazon ECR 리포지토리로 푸시된 후 다른 위치에서 이미지를 가져올 수 있습니다. 다음 사전 요구 사항이 충족되면 docker CLI를 사용하여 이미지를 가져올 수 있습니다.

- 최소 버전 설치: docker 1.7.
- Amazon ECR 인증 토큰은 로 docker login 구성되었습니다.
- Amazon ECR 리포지토리가 있어야 하며 사용자에게 리포지토리로부터 가져올 수 있는 액세스 권한이 있어야 합니다.

이러한 사전 조건이 만족되면 이미지를 가져올 수 있습니다. Amazon ECR에서 예제 이미지를 가져오려면, 다음 명령을 실행합니다.

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/hello-repository:latest
```

**출력:**

```
latest: Pulling from hello-repository
0a85502c06c9: Pull complete
0998bf8fb9e9: Pull complete
a6785352b25c: Pull complete
e9ae3c220b23: Pull complete
Digest: sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636EXAMPLE
Status: Downloaded newer image for aws_account_id.dkr.region.amazonaws.com/hello-repository:latest
```

## 6단계: 이미지 삭제

리포지토리 중 하나에 이미지가 더 이상 필요하지 않은 경우 이미지를 삭제할 수 있습니다. 이미지를 삭제하려면 해당 이미지가 있는 저장소를 지정하고 이미지의 `imageTag` or `imageDigest` 값을 지정합니다. 다음 예제에서는 이미지 태그가 `latest` 있는 `hello-repository` 저장소의 이미지를 삭제합니다. 리포지토리에서 예제 이미지를 삭제하려면 다음 명령을 실행합니다.

```
aws ecr batch-delete-image \  
  --repository-name hello-repository \  
  --image-ids imageTag=latest \  
  --region region
```

## 7단계: 리포지토리 삭제

전체 이미지 리포지토리가 더 이상 필요하지 않은 경우 리포지토리를 삭제할 수 있습니다. 다음 예제에서는 `--force` 플래그를 사용하여 이미지가 포함된 저장소를 삭제합니다. 이미지가 들어 있는 리포지토리(및 리포지토리 안에 있는 모든 이미지)를 삭제하려면 다음 명령을 실행합니다.

```
aws ecr delete-repository \  
  --repository-name hello-repository \  
  --force \  
  --region region
```

# Amazon ECR의 성능 최적화

Amazon ECR을 사용할 때 성능을 최적화하기 위한 설정 및 전략에 대한 다음 권장 사항을 사용할 수 있습니다.

## Docker 1.10 이상을 사용하여 동시 계층 업로드 활용

도커 이미지는 계층으로 구성되어 있으며, 계층은 이미지의 중간 빌드 단계입니다. Dockerfile의 각 행은 새로운 계층을 생성합니다. Docker 1.10 이상을 사용하는 경우 Docker는 기본적으로 Amazon ECR에 동시 업로드를 수행하면서 가능한 많은 계층을 푸시하여 업로드 시간이 단축되도록 설정됩니다.

## 작은 크기의 기본 이미지 사용

Docker Hub를 통해 제공되는 기본 이미지에는 사용자의 애플리케이션에서는 필요하지 않은 많은 종속 프로그램이 포함되어 있을 수 있습니다. Docker 커뮤니티에서 다른 사람이 생성하고 유지 관리하는 작은 크기의 이미지를 사용하는 것을 고려하거나, 아니면 Docker의 최소 scratch 이미지를 사용하여 사용자 고유의 기본 이미지를 빌드하십시오. 자세한 내용은 Docker 설명서에서 [기본 이미지 생성](#)을 참조하십시오.

## Dockerfile에서 가장 최근에 변경된 종속 프로그램 찾기

Docker는 계층을 캐시하여 빌드 시간을 줄입니다. 마지막 빌드 이후 계층에 변경된 사항이 없으면 Docker는 계층을 다시 빌드하는 대신 캐시된 버전을 사용합니다. 그러나, 각 계층은 이전 빌드의 계층에 종속되어 있습니다. 계층이 변경되면 Docker는 해당 계층뿐만 아니라 해당 계층 이후의 계층도 다시 컴파일합니다.

Docker 파일을 다시 빌드하고 계층을 다시 업로드하는 데 필요한 시간을 최소화하려면, Dockerfile의 앞쪽에는 가장 적은 횟수로 변경되는 종속 프로그램을 넣습니다. 자주 변경되는 종속 프로그램(애플리케이션의 소스 코드 등)은 스택의 뒤쪽에 넣으십시오.

## 불필요한 파일 저장을 방지하는 체인 명령

계층에 생성된 중간 파일은 후속 계층에서 삭제되더라도 해당 계층의 일부로 남습니다. 다음 예제를 검토하십시오.

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz
RUN wget tar -xvf software.tar.gz
RUN mv software/binary /opt/bin/myapp
RUN rm software.tar.gz
```

이 예에서는, 첫 번째 및 두 번째 RUN 명령에 의해 생성된 계층에 원본 .tar.gz 파일 및 이 파일의 압축되지 않은 콘텐츠가 모두 들어 있습니다. 이는 네 번째 RUN 명령에 의해 .tar.gz 파일이 삭제되는 경우에도 그렇습니다. 불필요한 파일이 최종 도커 이미지에 포함되지 않도록 하기 위해 이러한 명령은 다음과 같이 단일 RUN 문에 함께 묶여 있을 수 있습니다.

```
WORKDIR /tmp
RUN wget http://example.com/software.tar.gz &&\
    wget tar -xvf software.tar.gz &&\
    mv software/binary /opt/bin/myapp &&\
    rm software.tar.gz
```

### 가장 가까운 리전의 엔드포인트 사용

애플리케이션을 실행하고 있는 위치에서 가장 가까운 리전의 엔드포인트를 사용함으로써 Amazon ECR에서 이미지를 가져올 때 발생하는 지연 시간을 줄일 수 있습니다. 애플리케이션을 Amazon EC2 인스턴스에서 실행하고 있는 경우, 다음 셸 코드를 사용하여 인스턴스의 가용 영역에서 리전을 가져올 수 있습니다.

```
REGION=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone
|\
    sed -n 's/\(\\d*\)[a-zA-Z]*$/\1/p')
```

--region 파라미터를 사용하여 지역을 AWS CLI 명령에 전달하거나 aws configure 명령을 사용하여 프로필의 기본 지역으로 설정할 수 있습니다. AWS SDK를 사용하여 전화를 걸 때 지역을 설정할 수도 있습니다. 자세한 내용은 해당 프로그래밍 언어의 SDK 설명서를 참조하십시오.

# Amazon ECR 프라이빗 레지스트리

Amazon ECR 프라이빗 레지스트리는 가용성 및 확장성이 뛰어난 아키텍처에서 이미지를 호스팅합니다. 프라이빗 레지스트리를 사용하여 도커 이미지 및 Open Container Initiative(OCI) 이미지 그리고 아티팩트로 구성된 프라이빗 이미지 리포지토리를 관리할 수 있습니다. 각 AWS 계정은 기본 프라이빗 Amazon ECR 레지스트리와 함께 제공됩니다. Amazon ECR 퍼블릭 레지스트리에 대한 자세한 내용은 Amazon Elastic Container Registry 퍼블릭 사용 설명서의 [퍼블릭 레지스트리](#)를 참조하세요.

## 프라이빗 레지스트리 개념

- 기본 프라이빗 레지스트리의 URL은 `https://aws_account_id.dkr.ecr.us-west-2.amazonaws.com`입니다.
- 기본적으로 사용자의 계정은 프라이빗 레지스트리에 있는 리포지토리에 대한 읽기 및 쓰기 액세스 권한을 갖습니다. 하지만 사용자는 Amazon ECR API를 호출하고 사설 리포지토리로 이미지를 푸시하거나 프라이빗 리포지토리에서 이미지를 가져올 수 있는 권한이 필요합니다. Amazon ECR은 다양한 수준에서 사용자 액세스를 제어하는 관리형 IAM 정책을 몇 가지 제공합니다. 자세한 내용은 [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#) 섹션을 참조하세요.
- 프라이빗 레지스트리에 대해 Docker 클라이언트를 인증해야 `docker push` 및 `docker pull` 명령을 사용하여 해당 레지스트리의 리포지토리에 이미지를 푸시하고 가져올 수 있습니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.
- 프라이빗 리포지토리는 사용자 액세스 정책 및 리포지토리 정책 모두를 사용하여 제어할 수 있습니다. 리포지토리 정책에 대한 자세한 내용은 [Amazon ECR의 프라이빗 리포지토리 정책](#) 단원을 참조하세요.
- 프라이빗 레지스트리의 리포지토리는 프라이빗 레지스트리에 대한 복제를 구성하여 고유의 프라이빗 레지스트리의 리전 간에 그리고 별도의 계정 간에 복제할 수 있습니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 이미지 복제](#)을 참조하세요.

## Amazon ECR에서의 프라이빗 레지스트리 인증

AWS Management Console AWS CLI, 또는 AWS SDK를 사용하여 프라이빗 리포지토리를 생성하고 관리할 수 있습니다. 이러한 방법을 사용하여 이미지에 대해 목록 조회 또는 삭제 같은 일부 작업을 수행할 수도 있습니다. 이러한 클라이언트는 표준 AWS 인증 방법을 사용합니다. Amazon ECR API를 사용하여 이미지를 푸시하고 가져올 수 있지만, Docker CLI 또는 언어별 Docker 라이브러리를 사용하기 쉽습니다.

Docker CLI는 기본 IAM 인증 방법을 지원하지 않습니다. Amazon ECR에서 Docker 푸시 및 풀 요청을 인증하고 승인할 수 있도록 추가 단계를 수행해야 합니다.

다음에 세부적으로 설명된 레지스트리 인증 방법을 사용할 수 있습니다.

## Amazon ECR 자격 증명 헬퍼 사용

Amazon ECR은 Amazon ECR에 대해 이미지를 푸시하고 가져올 때 Docker 자격 증명을 더 쉽게 저장하고 사용할 수 있도록 Docker 자격 증명 헬퍼를 제공합니다. 설치 및 구성 단계는 [Amazon ECR Docker 자격 증명 헬퍼](#)를 참조하세요.

### Note

Amazon ECR Docker 자격 증명 헬퍼는 현재 멀티 팩터 인증(MFA)을 지원하지 않습니다.

## 권한 부여 토큰 사용

권한 부여 토큰의 권한 범위는 권한 부여 토큰을 검색하는 데 사용된 IAM 보안 주체의 권한 범위와 일치합니다. 권한 부여 토큰은 IAM 보안 주체가 액세스하고 12시간 동안 유효한 Amazon ECR 레지스트리에 액세스하는 데 사용됩니다. 인증 토큰을 얻으려면 [GetAuthorizationToken](#) API 작업을 사용하여 사용자 AWS 이름과 인코딩된 비밀번호가 포함된 base64로 인코딩된 인증 토큰을 검색해야 합니다. 이 AWS CLI `get-login-password` 명령은 권한 부여 토큰을 검색 및 디코딩한 다음 명령에 연결하여 인증할 수 있으므로 이 작업을 간소화합니다. `docker login`

`get-login`을 사용하여 Amazon ECR 프라이빗 레지스트리에 대해 Docker를 인증하려면

- `aws ecr get-login-password` 명령을 사용하여 Amazon ECR 레지스트리에 Docker를 인증하려면 명령을 실행합니다. `aws ecr get-login-password` 인증 토큰을 `docker login` 명령에 전달할 때 사용자 이름으로 AWS 값을 사용하고, 인증하려는 Amazon ECR 레지스트리 URI를 지정합니다. 여러 레지스트리에 대해 인증하는 경우 각 레지스트리에 대해 명령을 반복해야 합니다.

### Important

오류가 발생하면 최신 버전의 AWS CLI를 설치하거나 업그레이드합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 [AWS Command Line Interface 설치](#)를 참조하세요.

- [get-login-password](#) (AWS CLI)

```
aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

- [get-ECR \(\) LoginCommand](#) AWS Tools for Windows PowerShell

```
(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

## HTTP API 인증 사용

Amazon ECR은 [Docker 레지스트리 HTTP API](#)를 지원합니다. 그러나 Amazon ECR은 프라이빗 레지스트리이기 때문에 모든 HTTP 요청에 권한 부여 토큰을 제공해야 합니다. `for -H curl` 옵션을 사용하여 HTTP 인증 헤더를 추가하고 명령에서 제공하는 인증 토큰을 전달할 수 있습니다. `get-authorization-token` AWS CLI

Amazon ECR HTTP API로 인증하는 방법

1. `aws ecr get-authorization-token`를 사용하여 인증 토큰을 검색하고 환경 변수로 설정합니다. AWS CLI

```
TOKEN=$(aws ecr get-authorization-token --output text --query 'authorizationData[].authorizationToken')
```

2. API를 인증하려면 `$TOKEN` 변수를 `curl`의 `-H` 옵션에 전달합니다. 예를 들어 다음 명령은 Amazon ECR 리포지토리의 이미지 태그를 나열합니다. 자세한 내용은 [도커 레지스트리 HTTP API](#) 참조 문서를 살펴보세요.

```
curl -i -H "Authorization: Basic $TOKEN" https://aws_account_id.dkr.ecr.region.amazonaws.com/v2/amazonlinux/tags/list
```

출력값은 다음과 같습니다.

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Date: Thu, 04 Jan 2018 16:06:59 GMT
Docker-Distribution-Api-Version: registry/2.0
Content-Length: 50
```



```
Connection: keep-alive
```

```
{"name":"amazonlinux","tags":["2017.09","latest"]}
```

## Amazon ECR의 프라이빗 레지스트리 설정

Amazon ECR은 프라이빗 레지스트리 설정을 사용하여 레지스트리 수준에서 기능을 구성합니다. 프라이빗 레지스트리 설정은 각 리전에 대해 별도로 구성됩니다. 프라이빗 레지스트리 설정을 사용하여 다음 기능을 구성할 수 있습니다.

- 레지스트리 권한 - 레지스트리 권한 정책은 복제에 대한 제어 및 풀스루 캐시 권한을 제공합니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 권한](#)을 참조하세요.
- 풀스루 캐시 규칙 - 풀스루 캐시 규칙은 Amazon ECR 프라이빗 레지스트리의 업스트림 레지스트리에서 이미지를 캐시할 때 사용됩니다. 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화](#)을 참조하세요.
- 복제 구성 - 복제 구성은 리포지토리가 AWS 리전 또는 계정 간에 복사되는지 여부를 제어하는 데 사용됩니다. 자세한 내용은 [Amazon ECR에서의 프라이빗 이미지 복제단원을 참조](#)하세요.
- 리포지토리 생성 템플릿 - Amazon ECR에서 사용자를 대신하여 새 리포지토리를 생성할 때 적용할 표준 설정을 정의하는 데 리포지토리 생성 템플릿이 사용됩니다. 풀스루 캐시 작업으로 생성된 리포지토리를 예로 들 수 있습니다. 자세한 정보는 [풀스루 캐시 작업 중에 생성된 리포지토리를 제어하기 위한 템플릿](#)을 참조하세요.
- 스캔 구성(Scanning configuration) - 기본적으로 레지스트리는 기본 스캔을 사용하도록 설정되어 있습니다. 고급 스캔을 사용하도록 설정할 수 있으며, 이 기능은 운영 체제 및 프로그래밍 언어 패키지 취약성을 모두 스캔하는 자동 연속 스캔 모드를 제공합니다. 자세한 정보는 [이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다](#)을 참조하세요.

## Amazon ECR의 프라이빗 레지스트리 권한

Amazon ECR은 레지스트리 정책을 사용하여 프라이빗 레지스트리 수준의 AWS 보안 주체에 권한을 부여합니다. 이러한 권한은 복제에 대한 액세스 범위를 지정하고 캐시 기능을 풀스루하는 데 사용됩니다.

Amazon ECR은 프라이빗 레지스트리 수준에서만 다음 권한을 적용합니다. 레지스트리 정책에 추가 작업이 추가되면 오류가 발생합니다.

- `ecr:ReplicateImage` – 소스 레지스트리라고 하는 다른 계정에 이미지를 레지스트리에 복제할 수 있는 권한을 부여합니다. 교차 계정 복제에만 사용됩니다.
- `ecr:BatchImportUpstreamImage` – 외부 이미지를 검색하고 이 이미지를 프라이빗 레지스트리로 가져올 수 있는 권한을 부여합니다.
- `ecr:CreateRepository` – 프라이빗 레지스트리에 리포지토리를 생성할 수 있는 권한을 부여합니다. 이 권한은 복제되거나 캐시된 이미지를 저장하는 리포지토리가 프라이빗 레지스트리에 이미 존재하지 않는 경우에 필요합니다.

### Note

`ecr:*` 작업을 프라이빗 레지스트리 권한 정책에 추가할 수는 있지만 와일드카드를 사용하는 대신 사용 중인 기능에 따라 필요한 특정 작업만 추가하는 것이 가장 좋습니다.

## 주제

- [Amazon ECR의 프라이빗 레지스트리 정책 예제](#)
- [Amazon ECR에서 계정 간 복제를 위한 레지스트리 권한 부여](#)
- [Amazon ECR의 풀스루 캐시에 대한 레지스트리 권한 부여](#)

## Amazon ECR의 프라이빗 레지스트리 정책 예제

다음 예제는 사용자가 갖는 Amazon ECR 레지스트리 관련 권한을 제어하는 데 사용할 수 있는 정책 설명 보여줍니다.

### Note

각 예제에서 `ecr:CreateRepository` 작업이 레지스트리 사용 권한 설명에서 제거되면 복제가 계속 발생할 수 있습니다. 그러나 복제가 성공하려면 계정 내에서 이름이 같은 리포지토리를 만들어야 합니다.

### 예제: 소스 계정의 루트 사용자가 모든 리포지토리를 복제하도록 허용

다음 레지스트리 권한 정책은 원본 계정의 루트 사용자가 모든 리포지토리를 복제할 수 있도록 허용합니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"ReplicationAccessCrossAccount",
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::source_account_id:root"
      },
      "Action":[
        "ecr:CreateRepository",
        "ecr:ReplicateImage"
      ],
      "Resource": [
        "arn:aws:ecr:us-west-2:your_account_id:repository/*"
      ]
    }
  ]
}
```

### 예: 여러 계정의 루트 사용자 허용

다음 레지스트리 권한 정책에는 두 가지 설명이 있습니다. 각 명령문은 소스 계정의 루트 사용자가 모든 리포지토리를 복제할 수 있도록 허용합니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"ReplicationAccessCrossAccount",
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::source_account_id:root"
      },
      "Action":[
        "ecr:CreateRepository",
        "ecr:ReplicateImage"
      ],
      "Resource": [
        "arn:aws:ecr:us-west-2:your_account_id:repository/*"
      ]
    },
    {
```

```

    "Sid": "ReplicationAccessCrossAccount",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::source_account_id:root"
    },
    "Action": [
      "ecr:CreateRepository",
      "ecr:ReplicateImage"
    ],
    "Resource": [
      "arn:aws:ecr:us-west-2:your_account_id:repository/*"
    ]
  }
]
}

```

예제: 소스 계정의 루트 사용자가 접두사가 **prod-**인 모든 리포지토리를 복제하도록 허용합니다.

다음 레지스트리 권한 정책은 소스 계정의 루트 사용자가 로 시작하는 모든 저장소를 복제할 수 있도록 허용합니다. prod-

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReplicationAccessCrossAccount",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::source_account_id:root"
      },
      "Action": [
        "ecr:CreateRepository",
        "ecr:ReplicateImage"
      ],
      "Resource": [
        "arn:aws:ecr:us-west-2:your_account_id:repository/prod-*"
      ]
    }
  ]
}

```

## Amazon ECR에서 계정 간 복제를 위한 레지스트리 권한 부여

교차 계정 정책 유형은 소스 레지스트리에서 사용 중인 레지스트리로 리포지토리를 복제할 수 있도록 AWS 보안 주체에 권한을 부여하는 데 사용됩니다. 기본적으로 자체 레지스트리 내에서 교차 리전 복제를 구성할 수 있는 권한이 있습니다. 레지스트리에 콘텐츠를 복제할 수 있는 다른 계정에 권한을 부여하는 경우에만 레지스트리 정책을 구성하면 됩니다.

레지스트리 정책은 `ecr:ReplicateImage` API 작업에 대해 권한을 부여해야 합니다. 이 API는 리전 또는 계정 간에 이미지를 복제할 수 있는 내부 Amazon ECR API입니다. `ecr:CreateRepository` 권한에 대한 권한을 부여할 수 있습니다. 이 권한을 사용하면 Amazon ECR이 레지스트리에 리포지토리가 없는 경우 리포지토리를 생성할 수 있습니다. 만약 `ecr:CreateRepository` 권한이 제공되지 않으면 소스 리포지토리와 이름이 같은 리포지토리를 레지스트리에 수동으로 생성해야 합니다. 두 작업이 모두 수행되지 않으면 복제가 실패합니다. 모든 실패 `CreateRepository` 또는 `ReplicateImage` API 작업이 에 표시됩니다. CloudTrail

복제에 대한 권한 정책을 구성하는 방법(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 레지스트리 정책을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리(Private registry), 레지스트리 권한(Registry permissions)을 선택합니다.
4. 레지스트리 권한(Registry permissions) 페이지에서 문 생성(Generate statement)을 선택합니다.
5. 정책 생성기를 사용하여 정책 설명을 정의하려면 다음 단계를 수행하세요.
  - a. 정책 유형(Policy type)으로 교차 계정 정책(Cross account policy)을 선택합니다.
  - b. 설명 ID(Statement ID)에 고유한 설명 ID를 입력합니다. 이 필드는 레지스트리 정책에서 Sid로 사용됩니다.
  - c. 계정(Accounts)에는 권한을 부여할 각 계정의 계정 ID를 입력합니다. 여러 계정 ID를 지정할 경우 쉼표로 구분합니다.
6. 정책 설명 미리 보기(Preview policy statement) 섹션을 확장하여 레지스트리 사용 권한 정책 설명을 검토합니다.
7. 정책 설명이 확인되면 정책에 추가(Add to policy)를 선택하여 정책을 레지스트리에 저장합니다.

복제에 대한 권한 정책을 구성하는 방법(AWS CLI)

1. `registry_policy.json`이라는 이름의 파일을 만들고 레지스트리 정책으로 채웁니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReplicationAccessCrossAccount",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::source_account_id:root"
      },
      "Action": [
        "ecr:CreateRepository",
        "ecr:ReplicateImage"
      ],
      "Resource": [
        "arn:aws:ecr:us-west-2:your_account_id:repository/*"
      ]
    }
  ]
}
```

2. 정책 파일을 사용하여 레지스트리 정책을 만듭니다.

```
aws ecr put-registry-policy \
  --policy-text file://registry_policy.json \
  --region us-west-2
```

3. 레지스트리에 대한 정책을 검색하여 확인합니다.

```
aws ecr get-registry-policy \
  --region us-west-2
```

## Amazon ECR의 풀스루 캐시에 대한 레지스트리 권한 부여

Amazon ECR 프라이빗 레지스트리 권한은 풀스루 캐시를 사용하도록 개별 IAM 엔터티의 권한 범위를 지정하는 데 활용할 수 있습니다. IAM 엔터티에 레지스트리 권한 정책에서 허용하는 권한보다 IAM 정책에서 부여한 권한이 많을 경우 IAM 정책이 우선합니다.

프라이빗 레지스트리에 대한 권한 정책을 생성하는 방법(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.

2. 탐색 모음에서 프라이빗 레지스트리 권한 문을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 레지스트리 권한을 선택합니다.
4. 레지스트리 권한(Registry permissions) 페이지에서 문 생성(Generate statement)을 선택합니다.
5. 생성하려는 각 풀스루 캐시 권한 정책 문에 대해 다음을 수행합니다.
  - a. 정책 유형(Policy type)으로 풀스루 캐시 정책(Pull through cache policy)을 선택합니다.
  - b. 문 ID(Statement id)에서 풀스루 캐시 문 정책의 이름을 입력합니다.
  - c. IAM 엔터티(IAM entities)에서 정책에 포함할 사용자, 그룹 또는 역할을 지정합니다.
  - d. 리포지토리 네임스페이스(Repository namespace)로 정책을 연결할 풀스루 캐시 규칙을 선택합니다.
  - e. 리포지토리 이름(Repository names)에서 규칙을 적용할 리포지토리 기본 이름을 지정합니다. 예를 들어, Amazon ECR Public에서 Amazon Linux 리포지토리를 지정하려는 경우 리포지토리 이름은 `amazonlinux`입니다.

# Amazon ECR 프라이빗 리포지토리

Amazon ECR 사설 리포지토리에는 Docker 이미지, 오픈 컨테이너 이니셔티브 (OCI) 이미지 및 OCI 호환 아티팩트가 포함되어 있습니다. Amazon ECR API 작업 또는 Amazon ECR 콘솔의 리포지토리 섹션을 사용하여 이미지 리포지토리를 생성, 모니터링 및 삭제하고 이미지 리포지토리에 액세스할 수 있는 사용자를 제어하는 권한을 설정할 수 있습니다. 또한 Amazon ECR은 Docker CLI와 통합되므로 개발 환경의 이미지를 리포지토리로 푸시하고 가져올 수 있습니다.

## 주제

- [프라이빗 리포지토리 개념](#)
- [이미지를 저장하기 위한 Amazon ECR 사설 리포지토리 생성](#)
- [Amazon ECR에서 프라이빗 리포지토리의 콘텐츠 및 세부 정보 보기](#)
- [Amazon ECR에서 프라이빗 리포지토리 삭제](#)
- [Amazon ECR의 프라이빗 리포지토리 정책](#)
- [Amazon ECR의 프라이빗 리포지토리에 태그 지정](#)

## 프라이빗 리포지토리 개념

- 기본적으로, 사용자의 계정은 자신의 기본 레지스트리에 있는 리포지토리에 대한 읽기 및 쓰기 액세스 권한을 갖습니다(`aws_account_id.dkr.ecr.region.amazonaws.com`). 그러나 사용자는 Amazon ECR API를 호출하고 리포지토리로 이미지를 푸시하거나 리포지토리에서 이미지를 가져올 수 있는 권한이 필요합니다. Amazon ECR은 다양한 수준에서 사용자 액세스를 제어하는 관리형 IAM 정책을 몇 가지 제공합니다. 자세한 정보는 [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#)을 참조하세요.
- 리포지토리는 사용자 액세스 정책 및 개별 리포지토리 정책 모두를 사용하여 제어할 수 있습니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리 정책](#)을 참조하세요.
- 리포지토리 이름은 네임스페이스를 지원할 수 있으며 이를 통해 유사한 리포지토리를 그룹화할 수 있습니다. 예를 들어 동일한 레지스트리를 사용하는 팀이 여러 개 있는 경우, 팀 A는 team-a 네임스페이스를 사용할 수 있는 반면 팀 B는 team-b 네임스페이스를 사용할 수 있습니다. 이렇게 함으로써 각 팀은 팀 네임스페이스가 앞에 붙은 web-app라는 고유의 이미지를 보유하게 됩니다. 이 구성을 사용하면 각 팀의 이러한 이미지를 간섭 없이 동시에 사용할 수 있습니다. 팀 A의 이미지는 team-a/web-app이고 팀 B의 이미지는 team-b/web-app입니다.



- 이미지를 자신의 레지스트리에 있는 리전 및 계정 간에 다른 리포지토리로 복제할 수 있습니다. 레지스트리 설정에서 복제 구성을 지정하여 이 작업을 수행할 수 있습니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 설정](#)을 참조하세요.

## 이미지를 저장하기 위한 Amazon ECR 사설 리포지토리 생성

Amazon ECR 사설 리포지토리를 생성한 다음 해당 리포지토리를 사용하여 컨테이너 이미지를 저장합니다. AWS Management Console을 사용하여 다음 단계에 따라 프라이빗 리포지토리를 생성합니다. 를 사용하여 리포지토리를 생성하는 단계는 AWS CLI을 참조하십시오 [3단계: 리포지토리 생성](#).

리포지토리를 생성하려면(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
  2. 탐색 모음에서 리포지토리를 생성할 리전을 선택합니다.
  3. 리포지토리 페이지에서 개인 리포지토리를 선택한 다음 리포지토리 생성을 선택합니다.
  4. 가시성 설정(Visibility settings)에서 프라이빗(Private)이 선택되었는지 확인합니다.
  5. 리포지토리 이름(Repository name)에 리포지토리의 고유한 이름을 입력합니다. 리포지토리 이름은 자체적으로 지정할 수 있습니다(예: nginx-web-app). 또는 리포지토리를 범주로 그룹화하기 위해 네임스페이스에 추가할 수 있습니다(예:project-a/nginx-web-app).
- ### Note
- 리포지토리 이름은 최대 256개의 문자를 포함할 수 있습니다. 이름은 문자로 시작해야 하고, 소문자와 숫자, 하이픈, 밑줄, 마침표 및 슬래시(/)만 포함할 수 있습니다. 이중 하이픈, 이중 밑줄 또는 이중 슬래시는 사용할 수 없습니다.
6. 태그 불변성(Tag immutability)에서 리포지토리의 태그 변경 가능 설정을 선택합니다. 변경 불가능 태그로 구성된 리포지토리는 이미지 태그를 덮어쓰는 것을 방지해 줍니다. 자세한 정보는 [Amazon ECR에서 이미지 태그의 덮어쓰기 방지](#)을 참조하세요.
  7. 푸시 시 스캔의 경우 기본 스캔을 위해 리포지토리 수준에서 스캔 설정을 지정할 수 있지만 프라이빗 레지스트리 수준에서 스캔 구성을 지정하는 것이 가장 좋습니다. 프라이빗 레지스트리에서 스캔 설정을 지정하면 고급 스캔 또는 기본 스캔을 사용하고 필터를 정의하여 스캔할 리포지토리를 지정할 수 있습니다. 자세한 정보는 [이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다](#)을 참조하세요.
  8. KMS 암호화의 경우 를 사용하여 저장소의 이미지 암호화를 활성화할지 여부를 선택합니다. AWS Key Management Service기본적으로 KMS 암호화가 활성화되면 Amazon ECR은 별칭이 있는

AWS 관리형 키 (KMS 키) 를 사용합니다. `aws/ecr` 이 키는 처음으로 KMS 암호화를 활성화하여 리포지토리를 만들 때 계정에 생성됩니다. 자세한 정보는 [저장된 데이터 암호화](#)을 참조하세요.

9. KMS 암호화가 활성화된 경우 고객 암호화 설정(고급)(Customer encryption settings (advanced))을 선택하여 고유의 KMS 키를 선택합니다. KMS 키가 클러스터와 동일한 리전에 있어야 합니다. [Create a AWS KMS key] 를 선택하여 AWS KMS 콘솔로 이동하여 자체 키를 생성합니다.
10. 리포지토리 생성을 선택합니다.

## 다음 단계

리포지토리로 이미지를 푸시하는 단계를 보려면 리포지토리를 선택하고 푸시 명령 보기를 선택합니다. 리포지토리로 이미지를 푸시하는 방법에 대한 자세한 내용은 [이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#) 섹션을 참조하세요.

## Amazon ECR에서 프라이빗 리포지토리의 콘텐츠 및 세부 정보 보기

사실 리포지토리를 생성한 후에는 다음에서 리포지토리에 대한 세부 정보를 볼 수 있습니다. AWS Management Console

- 리포지토리에 저장되는 이미지
- 각 이미지의 크기 및 SHA 다이제스트 등 리포지토리에 저장된 각 이미지에 대한 세부 정보
- 리포지토리 내용에 대해 지정된 스캔 빈도
- 리포지토리에 연결된 활성 폴스루 캐시 규칙이 있는지 여부
- 리포지토리에 대한 암호화 설정

### Note

Docker 버전 1.9로 시작하면, Docker 클라이언트가 V2 Docker 레지스트리에 이미지 계층을 푸시하기 전에 이를 압축합니다. `docker images` 명령의 출력은 압축되지 않은 이미지 크기를 표시합니다. 따라서 Docker는 AWS Management Console에 표시된 이미지보다 큰 이미지를 반환할 수 있음을 염두에 둡니다.

리포지토리 정보를 보려면(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.

2. 탐색 모음에서 보려는 리포지토리가 포함된 리전을 선택합니다.
3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 프라이빗(Private) 탭을 선택한 다음 리포지토리를 확인합니다.
5. 리포지토리 세부 정보 페이지에서 콘솔은 기본적으로 이미지(Images) 보기로 설정되어 있습니다. 리포지토리에 대한 다른 정보를 확인하기 위해서는 탐색 메뉴를 사용합니다.
  - 요약(Summary)을 선택하여 리포지토리에 대한 리포지토리 세부 정보 및 풀 카운트 데이터를 봅니다.
  - 이미지(Images)를 선택하여 리포지토리에 있는 이미지 태그에 대한 정보를 봅니다. 이미지에 대한 자세한 정보를 보려면 해당 이미지 태그를 선택합니다. 자세한 정보는 [Amazon ECR에서 이미지 세부 정보 보기](#)를 참조하세요.

태그가 지정되지 않은 이미지 중 삭제하고 싶은 이미지가 있는 경우, 삭제할 리포지토리의 왼쪽에 있는 상자를 선택하여 삭제>Delete)를 선택할 수 있습니다. 자세한 정보는 [Amazon ECR에서 이미지 삭제](#)를 참조하세요.

- 권한(Permissions)을 선택하여 리포지토리에 적용된 리포지토리 정책을 봅니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리 정책](#)을 참조하세요.
- 수명 주기 정책(Lifecycle Policy)을 선택하여 리포지토리에 적용된 수명 주기 정책을 봅니다. 여기서 수명 주기 내역도 볼 수 있습니다. 자세한 정보는 [Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.](#)을 참조하세요.
- 태그(Tags)를 선택하여 리포지토리에 적용된 메타데이터 태그를 봅니다.

## Amazon ECR에서 프라이빗 리포지토리 삭제

리포지토리 사용을 마치면 이를 삭제할 수 있습니다. 에서 리포지토리를 삭제하면 리포지토리에 포함된 모든 이미지도 삭제되므로 취소할 수 없습니다. AWS Management Console

### Important

삭제된 저장소의 이미지도 삭제됩니다. 이 작업은 실행 취소할 수 없습니다.

리포지토리를 삭제하려면(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 삭제하려는 리포지토리가 들어 있는 리전을 선택합니다.

3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 프라이빗(Private) 탭을 선택한 다음 삭제할 리포지토리를 선택하고 삭제>Delete)를 선택합니다.
5. **repository\_name** 삭제 창에서 삭제할 리포지토리가 선택되었는지 확인한 후 삭제>Delete)를 선택합니다.

## Amazon ECR의 프라이빗 리포지토리 정책

Amazon ECR은 리소스 기반 권한을 사용하여 리포지토리에 대한 액세스를 제어합니다. 리소스 기반 권한을 사용하면 리포지토리에 액세스할 수 있는 사용자 또는 역할과 해당 사용자가 리포지토리에서 수행할 수 있는 작업을 지정할 수 있습니다. 기본적으로 리포지토리를 만든 AWS 계정만 리포지토리에 액세스할 수 있습니다. 저장소에 대한 추가 액세스를 허용하는 저장소 정책을 적용할 수 있습니다.

### 주제

- [리포지토리 정책과 IAM 정책 비교](#)
- [Amazon ECR의 프라이빗 리포지토리 정책 예제](#)
- [Amazon ECR에서 프라이빗 리포지토리 정책 설명 설정](#)

## 리포지토리 정책과 IAM 정책 비교

Amazon ECR 리포지토리 정책은 개별 Amazon ECR 리포지토리에 대한 액세스를 제어하도록 범위가 지정되고 사용되는 IAM 정책의 하위 집합입니다. IAM 정책은 일반적으로 Amazon ECR 서비스 전체에 대한 권한을 적용하는 데 사용되지만, 특정 리소스에 대한 액세스를 제어하는 데도 사용할 수 있습니다.

Amazon ECR 리포지토리 정책과 IAM 정책 모두 특정 사용자 또는 역할이 리포지토리에서 수행할 수 있는 작업을 결정할 때 사용됩니다. 리포지토리 정책에서는 특정 사용자 또는 역할의 작업 수행이 허용되었지만 IAM 정책에서는 권한이 거부된 경우(또는 그 반대의 경우), 해당 작업은 거부됩니다. 리포지토리 정책 또는 IAM 정책 중 하나에서만 사용자 또는 역할의 작업 권한이 허용되면 되고, 양쪽 모두에서 작업이 허용되어야 할 필요는 없습니다.

### Important

Amazon ECR의 요구 사항에 따라 사용자가 레지스트리에 대해 인증하고 Amazon ECR 리포지토리에서 이미지를 푸시 또는 풀하기 전에 IAM 정책을 통해 `ecr:GetAuthorizationToken` API에 호출을 할 권한이 있어야 합니다. Amazon ECR은 다

양한 수준에서 사용자 액세스를 제어하는 관리형 IAM 정책을 몇 가지 제공합니다. 자세한 내용은 [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#)(를) 참조하십시오.

아래 예에서 보듯이 이러한 정책 유형 중 하나를 사용하여 리포지토리에 대한 액세스를 제어할 수 있습니다.

이 예에는 특정 사용자가 리포지토리와 그 리포지토리 내의 이미지를 설명할 수 있도록 하는 Amazon ECR 리포지토리 정책이 나와 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ECRRepositoryPolicy",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::account-id:user/username"},
      "Action": [
        "ecr:DescribeImages",
        "ecr:DescribeRepositories"
      ]
    }
  ]
}
```

이 예에서는 리소스 파라미터를 사용하여 정책 범위를 하나의 리포지토리로 지정(리포지토리의 전체 ARN으로 지정)하여 위와 동일한 목적을 달성하는 IAM 정책을 보여줍니다. Amazon 리소스 이름(ARN) 형식에 대한 자세한 내용은 [리소스](#) 단원을 참조하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDescribeRepoImage",
      "Effect": "Allow",
      "Action": [
        "ecr:DescribeImages",
        "ecr:DescribeRepositories"
      ],
      "Resource": ["arn:aws:ecr:region:account-id:repository/repository-name"]
    }
  ]
}
```

}

## Amazon ECR의 프라이빗 리포지토리 정책 예제

### ⚠ Important

이 페이지의 리포지토리 정책 예제는 Amazon ECR 프라이빗 리포지토리에 적용하기 위한 것입니다. Amazon ECR 리포지토리를 리소스로 지정하도록 수정하지 않으면 IAM 보안 주체와 직접 함께 사용할 경우 제대로 작동하지 않습니다. 리포지토리 정책 설정에 대한 자세한 내용은 [Amazon ECR에서 프라이빗 리포지토리 정책 설명 설정](#)를 참조하세요.

Amazon ECR 리포지토리 정책은 개별 Amazon ECR 리포지토리에 대한 액세스를 제어하도록 범위가 지정되고 사용되는 IAM 정책의 하위 집합입니다. IAM 정책은 일반적으로 Amazon ECR 서비스 전체에 대한 권한을 적용하는 데 사용되지만, 특정 리소스에 대한 액세스를 제어하는 데도 사용할 수 있습니다. 자세한 정보는 [리포지토리 정책과 IAM 정책 비교](#)를 참조하세요.

다음 리포지토리 정책 예제는 Amazon ECR 프라이빗 리포지토리에 대한 액세스를 제어하는 데 사용할 수 있는 권한 명령문을 보여 줍니다.

### ⚠ Important

Amazon ECR의 요구 사항에 따라 사용자가 레지스트리에 대해 인증하고 Amazon ECR 리포지토리에서 이미지를 푸시 또는 풀하기 전에 IAM 정책을 통해 `ecr:GetAuthorizationToken` API에 호출을 할 권한이 있어야 합니다. Amazon ECR은 다양한 수준에서 사용자 액세스를 제어하는 관리형 IAM 정책을 몇 가지 제공합니다. 자세한 내용은 [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#)(를) 참조하세요.

## 예제: 한 명 이상의 사용자 허용

다음 리포지토리 정책은 한 명 이상의 사용자가 리포지토리에 대해 이미지를 푸시하고 가져오도록 허용합니다.

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

```

    "Sid": "AllowPushPull",
    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::account-id:user/push-pull-user-1",
        "arn:aws:iam::account-id:user/push-pull-user-2"
      ]
    },
    "Action": [
      "ecr:BatchGetImage",
      "ecr:BatchCheckLayerAvailability",
      "ecr:CompleteLayerUpload",
      "ecr:GetDownloadUrlForLayer",
      "ecr:InitiateLayerUpload",
      "ecr:PutImage",
      "ecr:UploadLayerPart"
    ]
  }
]
}

```

## 예: 다른 계정 허용

다음 리포지토리 정책은 특정 계정이 이미지를 푸시하도록 허용합니다.

### Important

권한을 부여하려는 계정에는 리포지토리 정책을 만드는 리전이 활성화되어 있어야 합니다. 그렇지 않으면 오류가 발생합니다.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCrossAccountPush",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-id:root"
      },
      "Action": [
        "ecr:BatchCheckLayerAvailability",

```

```

        "ecr:CompleteLayerUpload",
        "ecr:InitiateLayerUpload",
        "ecr:PutImage",
        "ecr:UploadLayerPart"
    ]
}
]
}

```

다음 리포지토리 정책은 다른 사용자에게 전체 액세스 권한을 제공하면서(*admin-user*) 일부 사용자가 이미지를 가져오도록 허용합니다(*pull-user-1* and *pull-user-2*).

### Note

현재 에서 지원되지 않는 더 복잡한 리포지토리 정책의 AWS Management Console 경우 [set-repository-policy](#) AWS CLI 명령을 사용하여 정책을 적용할 수 있습니다.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPull",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::account-id:user/pull-user-1",
          "arn:aws:iam::account-id:user/pull-user-2"
        ]
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ]
    },
    {
      "Sid": "AllowAll",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-id:user/admin-user"
      },
      "Action": [

```



```

        "ecr:*"
    ]
}

```

## 예제: 모두 거부

다음 리포지토리 정책은 모든 계정의 모든 사용자가 이미지를 가져오는 기능을 거부합니다.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyPull",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ]
    }
  ]
}

```

## 예: 특정 IP 주소에 대한 액세스 제한

다음 예제에서는 특정 주소 범위에서 리포지토리에 적용할 때 어떠한 사용자에게도 Amazon ECR 작업을 수행할 수 있는 권한을 부여하지 않습니다.

이 문의 조건은 허용되는 IPv4(인터넷 프로토콜 버전 4) IP 주소의 54.240.143.\* 범위를 식별합니다.

Condition블록은 NotIpAddress 조건과 AWS-wide aws:SourceIp 조건 키인 조건 키를 사용합니다. 이 조건 키에 대한 자세한 내용은 [AWS 전역 조건 컨텍스트 키](#) 단원을 참조하십시오.

aws:sourceIp IPv4 값은 표준 CIDR 표기법을 사용합니다. 자세한 내용은 IAM 사용자 설명서의 [IP 주소 조건 연산자](#)를 참조합니다.

```

{
  "Version": "2012-10-17",
  "Id": "ECRPolicyId1",
  "Statement": [
    {

```

```

    "Sid": "IPAllow",
    "Effect": "Deny",
    "Principal": "*",
    "Action": "ecr:*",
    "Condition": {
      "NotIpAddress": {
        "aws:SourceIp": "54.240.143.0/24"
      }
    }
  }
]
}

```

## 예: 서비스 허용 AWS

다음 리포지토리 정책은 해당 서비스와의 통합에 필요한 Amazon ECR API 작업에 대한 AWS CodeBuild 액세스를 허용합니다. 다음 예제를 사용할 경우 `aws:SourceArn` 및 `aws:SourceAccount` 조건 키를 사용하여 이러한 권한을 수임할 수 있는 리소스의 범위를 지정해야 합니다. 자세한 내용은 [내용은 CodeBuild AWS CodeBuild 사용 설명서의 Amazon ECR 샘플을 참조하십시오.](#)

```

{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CodeBuildAccess",
      "Effect":"Allow",
      "Principal":{
        "Service":"codebuild.amazonaws.com"
      },
      "Action":[
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Condition":{
        "ArnLike":{
          "aws:SourceArn":"arn:aws:codebuild:region:123456789012:project/project-
name"
        },
        "StringEquals":{
          "aws:SourceAccount":"123456789012"
        }
      }
    }
  ]
}

```

```
]
}
```

## Amazon ECR에서 프라이빗 리포지토리 정책 설명 설정

아래 단계에 따라 리포지토리에 AWS Management Console 액세스 정책 설명을 추가할 수 있습니다. 리포지토리마다 정책 설명을 여러 개 추가할 수 있습니다. 예시 정책은 [Amazon ECR의 프라이빗 리포지토리 정책 예제](#) 섹션을 참조하세요.

### Important

Amazon ECR의 요구 사항에 따라 사용자가 레지스트리에 대해 인증하고 Amazon ECR 리포지토리에서 이미지를 푸시 또는 풀하기 전에 IAM 정책을 통해 `ecr:GetAuthorizationToken` API에 호출을 할 권한이 있어야 합니다. Amazon ECR은 다양한 수준에서 사용자 액세스를 제어하는 관리형 IAM 정책을 몇 가지 제공합니다. 자세한 내용은 [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#)을(를) 참조하십시오.

리포지토리 정책 설명을 설정하려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 정책 설명을 설정할 리포지토리가 들어 있는 리전을 선택합니다.
3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 리포지토리 콘텐츠를 보려면 정책 설명을 설정할 리포지토리를 선택합니다.
5. 리포지토리 이미지 목록 보기의 탐색 창에서 권한(Permissions), 편집(Edit)을 선택합니다.

### Note

탐색 창에 권한(Permissions) 옵션이 표시되지 않으면 리포지토리 이미지 목록 보기에 있는지 확인합니다.

6. 권한 편집(Edit permissions) 페이지에서 설명문 추가(Add statement)를 선택합니다.
7. 설명문 이름(Statement name)에 설명문 이름을 입력합니다.
8. 효과(Effect)에서 정책 설명문 결과가 허용 또는 명시적 거부인지 여부를 선택합니다.
9. 보안 주체(Principal)에서 정책 설명을 적용할 범위를 선택합니다. 자세한 내용은 IAM 사용 설명서의 [AWS JSON 정책 요소: 보안 주체](#)를 참조하세요.

- Everyone (\*) 확인란을 선택하여 인증된 모든 AWS 사용자에게 명령문을 적용할 수 있습니다.
- 정책 설명을 특정 서비스에 적용하려면 서비스 보안 주체(Service principal)에서 서비스 보안 주체(예: `ecs.amazonaws.com`)를 지정합니다.
- AWS 계정 ID의 경우 AWS 계정 번호 (예:111122223333) 를 지정하여 특정 AWS 계정의 모든 사용자에게 명세서를 적용하십시오. 심포로 구분된 목록을 사용하여 여러 계정을 지정할 수 있습니다.

#### Important

권한을 부여하려는 계정에는 리포지토리 정책을 만드는 리전이 활성화되어 있어야 합니다. 그렇지 않으면 오류가 발생합니다.

- IAM 엔티티의 경우 AWS 계정 내에서 명령문을 적용할 역할 또는 사용자를 선택합니다.

#### Note

현재 에서 지원되지 않는 더 복잡한 리포지토리 정책의 AWS Management Console경우 [set-repository-policy](#) AWS CLI 명령을 사용하여 정책을 적용할 수 있습니다.

10. 작업(Actions)의 경우 개별 API 작업 목록에서 정책 설명을 적용해야 하는 Amazon ECR API 작업의 범위를 선택합니다.
11. 완료되면 저장(Save)을 선택하여 정책을 설정합니다.
12. 추가할 각 리포지토리에 대해 이전 단계를 반복합니다.

## Amazon ECR의 프라이빗 리포지토리에 태그 지정

Amazon ECR 리포지토리를 관리하는 데 도움이 되도록 리소스 태그를 사용하여 새 Amazon ECR 리포지토리 또는 기존 Amazon ECR 리포지토리에 자체 메타데이터를 할당할 수 있습니다. AWS 예를 들어, 계정의 Amazon ECR 리포지토리에 대한 태그 집합을 정의하여 각 리포지토리의 소유자를 추적할 수 있습니다.

### 태그 기본 사항

태그는 Amazon ECR에는 의미가 없으며 엄격하게 문자열로 해석됩니다. 태그가 리소스에 자동으로 할당되는 것은 아닙니다. 태그 키와 값을 편집할 수 있으며 언제든지 리소스에서 태그를 제거할 수 있습니다. 태그의 값을 빈 문자열로 설정할 수 있지만 태그의 값을 Null로 설정할 수는 없습니다. 해당 리

소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다. 리소스를 삭제하면 리소스 태그도 삭제됩니다.

Amazon ECR 콘솔 AWS CLI, 및 Amazon ECR API를 사용하여 태그를 사용할 수 있습니다.

AWS Identity and Access Management (IAM) 을 사용하면 AWS 계정 내 어떤 사용자가 태그를 생성, 편집 또는 삭제할 수 있는지 제어할 수 있습니다. IAM 정책의 태그에 대한 자세한 내용은 [the section called “태그 기반 액세스 제어 사용”](#)을 참조하십시오.

## 리소스에 결제용 태그 지정

Amazon ECR 리포지토리에 추가하는 태그는 비용 및 사용 보고서에서 활성화 후 비용 할당을 검토할 때 유용합니다. 자세한 정보는 [Amazon ECR 사용 보고서](#)을 참조하세요.

결합된 리소스의 비용을 확인하려면 태그 키 값을 동일한 리소스에 따라 결제 정보를 구성할 수 있습니다. 예를 들어, 특정 애플리케이션 이름으로 여러 리소스에 태그를 지정한 다음 결제 정보를 구성하여 여러 서비스에 걸친 해당 애플리케이션의 총 비용을 볼 수 있습니다. 태그를 사용한 비용 할당 보고서 설정에 대한 자세한 내용은 AWS Billing 사용 설명서에서 [월간 비용 할당 보고서](#)를 참조하세요.

### Note

방금 보고서를 활성화한 경우, 24시간 후에 이번 달의 데이터를 볼 수 있습니다.

## Amazon ECR의 프라이빗 리포지토리에 태그 추가

개인 리포지토리에 태그를 추가할 수 있습니다.

이름 및 태그 모범 사례에 대한 자세한 내용은 AWS 리소스 태깅 사용 설명서의 [태그 이름 지정 제한 및 요구 사항](#) 및 [모범 사례](#)를 참조하십시오.

리포지토리에 태그 추가 ( )AWS Management Console

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 사용할 리전을 선택합니다.
3. 탐색 창에서 리포지토리(Repositories)를 선택합니다.
4. 리포지토리 페이지에서 태그를 지정하려는 리포지토리 옆의 확인란을 선택합니다.
5. 작업 메뉴에서 리포지토리 태그를 선택합니다.
6. 리포지토리 태그 페이지에서 태그 추가, 태그 추가를 선택합니다.

7. 리포지토리 태그 편집 페이지에서 각 태그의 키와 값을 지정한 다음 저장을 선택합니다.

리포지토리 (AWS CLI 또는 API) 에 태그 추가

또는 API를 사용하여 하나 이상의 태그를 추가하거나 덮어쓸 수 AWS CLI 있습니다.

- AWS CLI - [태그 리소스](#)
- API 액션 - [TagResource](#)

다음 예제는 를 사용하여 태그를 추가하는 방법을 보여줍니다 AWS CLI.

예 1: 리포지토리에 태그 지정

다음 명령은 리포지토리에 태그를 지정합니다.

```
aws ecr tag-resource \
  --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
  --tags Key=stack,Value=dev
```

예 2: 리포지토리에 여러 태그로 태그 지정

다음 명령은 리포지토리에 세 개의 태그를 추가합니다.

```
aws ecr tag-resource \
  --resource-arn arn:aws:ecr:region:account_id:repository/repository_name \
  --tags Key=key1,Value=value1 Key=key2,Value=value2 Key=key3,Value=value3
```

예 3: 리포지토리의 태그 목록

다음 명령은 리포지토리와 관련된 태그를 나열합니다.

```
aws ecr list-tags-for-resource \
  --resource-arn arn:aws:ecr:region:account_id:repository/repository_name
```

예 4: 리포지토리 생성 및 태그 추가

다음 명령은 test-repo라는 이름의 리포지토리를 생성하며 키 team 및 값 devs가 있는 태그를 추가합니다.

```
aws ecr create-repository \
```

```
--repository-name test-repo \  
--tags Key=team,Value=devs
```

## Amazon ECR의 프라이빗 리포지토리에서 태그 삭제

개인 저장소에서 태그를 삭제할 수 있습니다.

개인 리포지토리에서 태그를 삭제하려면 (AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 사용할 리전을 선택합니다.
3. 리포지토리 페이지에서 태그를 제거하려는 리포지토리 옆의 확인란을 선택합니다.
4. 작업 메뉴에서 리포지토리 태그를 선택합니다.
5. 리포지토리 태그 페이지에서 편집을 선택합니다.
6. 리포지토리 태그 편집 페이지에서 삭제할 각 태그에 대해 제거를 선택하고 저장을 선택합니다.

개인 저장소에서 태그를 삭제하려면 (AWS CLI)

또는 API를 사용하여 하나 이상의 태그를 삭제할 수 있습니다. AWS CLI

- AWS CLI - 리소스 태그 [해제](#)
- API 액션 - [UntagResource](#)

다음 예제는 를 사용하여 저장소에서 태그를 삭제하는 방법을 보여줍니다 AWS CLI.

```
aws ecr untag-resource \  
--resource-arn arn:aws:ecr:region:account_id:repository/repository_name \  
--tag-keys tag_key
```

# Amazon ECR의 프라이빗 이미지

Amazon ECR은 Docker 이미지, 오픈 컨테이너 이니셔티브 (OCI) 이미지 및 OCI 호환 아티팩트를 프라이빗 리포지토리에 저장합니다. Docker CLI를 사용하거나 선호하는 클라이언트를 사용하여 이미지를 리포지토리로 푸시 및 풀링할 수 있습니다.

## 주제

- [이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#)
- [Amazon ECR 사설 리포지토리에 저장된 이미지에 서명](#)
- [Amazon ECR 사설 리포지토리에서 서명 삭제](#)
- [Amazon ECR에서 이미지 세부 정보 보기](#)
- [Amazon ECR 사설 리포지토리에서 로컬 환경으로 이미지 가져오기](#)
- [아마존 리눅스 컨테이너 이미지 가져오기](#)
- [Amazon ECR에서 이미지 삭제](#)
- [Amazon ECR에서 이미지 태그 재지정](#)
- [Amazon ECR에서 이미지 태그의 덮어쓰기 방지](#)
- [Amazon ECR의 컨테이너 이미지 매니페스트 형식 지원](#)
- [Amazon ECS에서 Amazon ECR 이미지 사용](#)
- [Amazon EKS에서 Amazon ECR 이미지 사용](#)

## 이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기

Docker 이미지, 매니페스트 목록 및 Open Container Initiative(OCI) 이미지 및 호환 가능한 아티팩트를 프라이빗 리포지토리에 푸시할 수 있습니다.

Amazon ECR은 이미지를 다른 리포지토리로 복제하는 방법도 제공합니다. 프라이빗 레지스트리 설정에서 복제 구성을 지정하면 자체 레지스트리의 여러 지역과 다른 계정 간에 복제할 수 있습니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 설정](#)을 참조하세요.

## 주제

- [이미지를 Amazon ECR 사설 리포지토리로 푸시하기 위한 IAM 권한](#)
- [Docker 이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#)



- [다중 아키텍처 이미지를 Amazon ECR 사설 리포지토리로 푸시하기](#)
- [헬름 차트를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#)

## 이미지를 Amazon ECR 사설 리포지토리로 푸시하기 위한 IAM 권한

Amazon ECR 프라이빗 리포지토리로 이미지를 푸시하려면 IAM 권한이 필요합니다. 최소 권한 부여의 모범 사례에 따라 특정 리포지토리에 대한 액세스 권한을 부여할 수 있습니다. 모든 리포지토리에 액세스 권한을 부여할 수도 있습니다.

사용자는 인증 토큰을 요청하여 이미지를 푸시하려는 각 Amazon ECR 레지스트리에 대해 인증해야 합니다. Amazon ECR은 다양한 수준에서 사용자 액세스를 제어할 수 있는 몇 가지 AWS 관리형 정책을 제공합니다. 자세한 정보는 [AWS Amazon Elastic 컨테이너 레지스트리에 대한 관리형 정책](#)을 참조하세요.

자체 IAM 정책을 생성할 수도 있습니다. 다음 IAM 정책은 이미지를 특정 저장소로 푸시하는 데 필요한 권한을 부여합니다. 리포지토리를 전체 Amazon 리소스 이름(ARN)으로 지정해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:CompleteLayerUpload",
        "ecr:UploadLayerPart",
        "ecr:InitiateLayerUpload",
        "ecr:BatchCheckLayerAvailability",
        "ecr:PutImage"
      ],
      "Resource": "arn:aws:ecr:region:111122223333:repository/repository-name"
    },
    {
      "Effect": "Allow",
      "Action": "ecr:GetAuthorizationToken",
      "Resource": "*"
    }
  ]
}
```

다음 IAM 정책은 이미지를 모든 저장소로 푸시하는 데 필요한 권한을 부여합니다.

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

## Docker 이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기

docker push 명령을 사용하여 컨테이너 이미지를 Amazon ECR 리포지토리로 푸시할 수 있습니다.

또한 Amazon ECR은 다중 아키텍처 이미지에 사용되는 Docker 매니페스트 목록 생성 및 푸시를 지원합니다. 자세한 내용은 [다중 아키텍처 이미지를 Amazon ECR 사설 리포지토리로 푸시하기](#)을 참조하세요.

Amazon ECR 리포지토리에 Docker 이미지를 푸시하려면

이미지를 푸시하기 전에 Amazon ECR 리포지토리가 있어야 합니다. 자세한 정보는 [the section called “이미지를 저장할 리포지토리 생성”](#)을 참조하세요.

1. 이미지를 푸시하려는 Amazon ECR 레지스트리에 대해 Docker 클라이언트를 인증합니다. 인증 토큰은 사용되는 레지스트리마다 필요하며, 12시간 동안 유효합니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.

Amazon ECR 레지스트리에 대해 Docker를 인증하려면 aws ecr get-login-password 명령을 실행합니다. Amazon ECR 인증 토큰을 docker login 명령에 전달할 때 사용자 이름으로 AWS 값을 사용하고, 인증하려는 Amazon ECR 레지스트리 URI를 지정합니다. 여러 레지스트리에 대해 인증하는 경우 각 레지스트리에 대해 명령을 반복해야 합니다.

**⚠ Important**

오류가 발생하면 최신 버전의 AWS CLI를 설치하거나 업그레이드합니다. 자세한 내용을 알아보려면 AWS Command Line Interface 사용자 가이드에서 [AWS Command Line Interface 설치](#)를 참조하세요.

```
aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

2. 푸시하려는 레지스트리에 이미지 리포지토리가 아직 없으면 하나 생성합니다. 자세한 정보는 [이 이미지를 저장하기 위한 Amazon ECR 사설 리포지토리 생성](#)을 참조하세요.
3. 푸시할 로컬 이미지를 식별합니다. `docker images` 명령을 실행하여 시스템에 있는 컨테이너 이미지를 나열합니다.

```
docker images
```

명령 결과 출력에서 `repository:tag` 값 또는 이미지 ID를 확인하여 이미지를 식별할 수 있습니다.

4. 사용할 Amazon ECR 레지스트리, 리포지토리 및 이미지 태그 이름 조합(선택 사항)이 있는 이미지에 태그를 지정합니다. 레지스트리 형식은 `aws_account_id.dkr.ecr.us-west-2.amazonaws.com`입니다. 리포지토리 이름은 이미지에 대해 생성한 리포지토리와 일치해야 합니다. 이미지 태그를 생략하면 태그가 `latest`인 것으로 간주됩니다.

아래 예에서는 ID `e9ae3c220b23`을 `aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag`으로 사용하여 로컬 이미지에 태그를 지정합니다.

```
docker tag e9ae3c220b23 aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag
```

5. `docker push` 명령을 사용하여 이미지를 푸시합니다.

```
docker push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:tag
```

6. (선택 사항) [Step 4](#)과 [Step 5](#)(을)를 반복하여 이미지에 추가 태그를 적용하고 이러한 태그를 Amazon ECR에 푸시합니다.

## 다중 아키텍처 이미지를 Amazon ECR 사설 리포지토리로 푸시하기

Docker 매니페스트 목록을 생성하고 푸시하여 다중 아키텍처 이미지를 Amazon ECR 리포지토리로 푸시할 수 있습니다. 매니페스트 목록은 하나 이상의 이미지 이름을 지정하여 생성되는 이미지 목록입니다. 대부분의 경우 매니페스트 목록은 동일한 기능을 제공하지만 운영 체제 또는 아키텍처가 다른 이미지에서 생성됩니다. 매니페스트 목록은 필수는 아닙니다. 자세한 내용은 [도커 매니페스트](#)를 참조하세요.

매니페스트 목록은 다른 Amazon ECR 이미지와 마찬가지로 Amazon ECS 작업 정의 또는 Amazon EKS 포드 사양에서 가져오거나 참조할 수 있습니다.

### 사전 조건

- Docker CLI에서 실험용 기능을 켜세요. 실험 기능에 대한 자세한 내용은 Docker [설명서의 실험적 기능을](#) 참조하십시오.
- 이미지를 푸시하기 전에 Amazon ECR 리포지토리가 있어야 합니다. 자세한 정보는 [the section called “이미지를 저장할 리포지토리 생성”](#)을 참조하세요.
- Docker 매니페스트를 생성하기 전에 이미지를 리포지토리로 푸시해야 합니다. 이미지 푸시에 대한 자세한 내용은 [Docker 이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#) 단원을 참조하세요.

### 다중 아키텍처 Docker 이미지를 Amazon ECR 리포지토리에 푸시하려면

1. 이미지를 푸시하려는 Amazon ECR 레지스트리에 대해 Docker 클라이언트를 인증합니다. 인증 토큰은 사용되는 레지스트리마다 필요하며, 12시간 동안 유효합니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.

Amazon ECR 레지스트리에 대해 Docker를 인증하려면 `aws ecr get-login-password` 명령을 실행합니다. Amazon ECR 인증 토큰을 `docker login` 명령에 전달할 때 사용자 이름으로 AWS 값을 사용하고, 인증하려는 Amazon ECR 레지스트리 URI를 지정합니다. 여러 레지스트리에 대해 인증하는 경우 각 레지스트리에 대해 명령을 반복해야 합니다.

#### Important

오류가 발생하면 최신 버전의 AWS CLI를 설치하거나 업그레이드합니다. 자세한 내용을 알아보려면 AWS Command Line Interface 사용자 가이드에서 [AWS Command Line Interface설치](#)를 참조하세요.

```
aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

2. 이미지 태그를 확인하여 리포지토리의 이미지를 나열합니다.

```
aws ecr describe-images --repository-name my-repository
```

3. Docker 매니페스트 목록을 생성합니다. `manifest create` 명령은 참조된 이미지가 이미 리포지토리에 있는지 확인하고 로컬로 매니페스트를 생성합니다.

```
docker manifest create aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:image_one_tag aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository:image_two
```

4. (선택 사항) Docker 매니페스트 목록을 검사합니다. 이렇게 하면 매니페스트 목록에서 참조되는 각 이미지 매니페스트의 크기와 다이제스트를 확인할 수 있습니다.

```
docker manifest inspect aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository
```

5. Docker 매니페스트 목록을 Amazon ECR 리포지토리에 푸시합니다.

```
docker manifest push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/my-repository
```

## 헬름 차트를 Amazon ECR 프라이빗 리포지토리로 푸시하기

오픈 컨테이너 이니셔티브 (OCI) 아티팩트를 Amazon ECR 리포지토리로 푸시할 수 있습니다. 이 기능의 예를 보려면 다음 단계를 사용하여 Helm 차트를 Amazon ECR로 푸시하십시오.

Amazon EKS에서 Amazon ECR에 호스팅된 헬름 차트를 사용하는 방법에 대한 자세한 내용은 [참조하십시오. Amazon EKS 클러스터에 헬름 차트 설치](#)

Helm 차트를 Amazon ECR 리포지토리로 푸시하려면

1. Helm 클라이언트의 최신 버전을 설치합니다. 이 단계는 Helm 버전을 사용하여 작성되었습니다. 3.8.2. 자세한 정보는 [Helm 설치](#)를 참조하세요.

2. 다음 단계에 따라 테스트 Helm 차트를 생성합니다. 자세한 내용은 [Helm Docs - 시작하기](#)를 참조하세요.
  - a. helm-test-chart 라는 Helm 차트를 만들고 templates 디렉토리의 콘텐츠를 지웁니다.

```
helm create helm-test-chart
rm -rf ./helm-test-chart/templates/*
```

- b. ConfigMap 폴더에 를 생성하십시오. templates

```
cd helm-test-chart/templates
cat <<EOF > configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: helm-test-chart-configmap
data:
  myvalue: "Hello World"
EOF
```

3. 차트를 패키징합니다. 출력에는 Helm 차트를 푸시할 때 사용하는 패키지 차트의 파일 이름이 포함됩니다.

```
cd ../../
helm package helm-test-chart
```

#### 출력

```
Successfully packaged chart and saved it to: /Users/username/helm-test-chart-0.1.0.tgz
```

4. Helm 차트를 저장할 리포지토리를 생성합니다. 리포지토리 이름은 2단계에서 Helm 차트를 생성할 때 사용한 이름과 일치해야 합니다. 자세한 정보는 [이미지를 저장하기 위한 Amazon ECR 사설 리포지토리 생성](#)을 참조하세요.

```
aws ecr create-repository \
  --repository-name helm-test-chart \
  --region us-west-2
```

5. Helm 차트를 푸시하려는 Amazon ECR 레지스트리에 대해 Helm 클라이언트를 인증합니다. 인증 토큰은 사용되는 레지스트리마다 필요하며, 12시간 동안 유효합니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.

```
aws ecr get-login-password \
  --region us-west-2 | helm registry login \
  --username AWS \
  --password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
```

6. helm push 명령을 사용하여 Helm 차트를 푸시합니다. 출력에는 Amazon ECR 리포지토리 URI 및 SHA 다이제스트가 포함되어야 합니다.

```
helm push helm-test-chart-0.1.0.tgz oci://aws_account_id.dkr.ecr.us-west-2.amazonaws.com/
```

7. Helm 차트를 설명하세요.

```
aws ecr describe-images \
  --repository-name helm-test-chart \
  --region us-west-2
```

출력에서 artifactMediaType 파라미터가 적절한 아티팩트 유형을 나타내는지 확인합니다.

```
{
  "imageDetails": [
    {
      "registryId": "aws_account_id",
      "repositoryName": "helm-test-chart",
      "imageDigest":
      "sha256:dd8aebdda7df991a0ffe0b3d6c0cf315fd582cd26f9755a347a52adEXAMPLE",
      "imageTags": [
        "0.1.0"
      ],
      "imageSizeInBytes": 1620,
      "imagePushedAt": "2021-09-23T11:39:30-05:00",
      "imageManifestMediaType": "application/vnd.oci.image.manifest.v1+json",
      "artifactMediaType": "application/vnd.cncf.helm.config.v1+json"
    }
  ]
}
```

8. (선택 사항) 추가 단계로, Helm configmap을 설치하고 Amazon EKS를 시작합니다. 자세한 정보는 [Amazon EKS 클러스터에 헬름 차트 설치](#)를 참조하세요.

## Amazon ECR 사설 리포지토리에 저장된 이미지에 서명

Amazon ECR은 와 AWS Signer 통합되어 컨테이너 이미지에 서명할 수 있는 방법을 제공합니다. 컨테이너 이미지와 서명을 모두 프라이빗 리포지토리에 저장할 수 있습니다.

### 고려 사항

Amazon ECR 이미지 서명을 사용할 때는 다음 사항을 고려해야 합니다.

- 리포지토리에 저장된 서명은 리포지토리당 최대 이미지 수에 대한 서비스 할당량에 포함됩니다. 자세한 정보는 [Amazon ECR 서비스 할당량](#)을 참조하세요.
- Amazon ECR 수명 주기 정책을 사용할 때 규칙에 의해 OCI 이미지 인덱스를 만료 또는 삭제하는 작업을 수행하면 Amazon ECR은 해당 이미지 인덱스에서 참조하는 모든 서명을 24시간 이내에 삭제합니다.

### 필수 조건

시작하기 전에 다음 사전 요구 사항이 충족되어야 합니다.

- 최신 버전의 AWS CLI를 설치 및 구성합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 [AWS CLI최신 버전의 설치 또는 업데이트](#)를 참조하세요.
- 표기법 CLI와 표기법 플러그인을 설치합니다 AWS Signer . 자세한 내용은 AWS Signer 개발자 안내서의 [컨테이너 이미지 서명을 위한 사전 요구 사항](#)을 참조하세요.
- Amazon ECR 프라이빗 리포지토리에 저장된 컨테이너 이미지에 서명합니다. 자세한 정보는 [이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#)을 참조하세요.

## Notary 클라이언트에 대한 인증 구성

Notation CLI를 사용하여 서명을 만들려면 먼저 Amazon ECR에 인증할 수 있도록 클라이언트를 구성해야 합니다. Notation 클라이언트를 설치한 동일한 호스트에 Docker를 설치한 경우 Notation은 Docker 클라이언트에 사용하는 것과 동일한 인증 방법을 재사용합니다. Docker login 및 logout 명령을 사용하면 Notation sign 및 verify 명령에서 동일한 자격 증명을 사용할 수 있으며 Notation을



별도로 인증할 필요가 없습니다. 인증을 위한 Notation 클라이언트 구성에 대한 자세한 내용은 Notary Project 문서의 [OCI 준수 레지스트리를 통한 인증](#)을 참조하세요.

Docker 또는 Docker 자격 증명을 사용하는 다른 도구를 사용하지 않는 경우 Amazon ECR Docker 자격 증명 도우미를 자격 증명 스토어로 사용하는 것이 좋습니다. Amazon ECR 자격 증명 도우미를 설치하고 구성하는 방법에 대한 자세한 내용은 [Amazon ECR Docker 자격 증명 도우미](#)를 참조하세요.

## 이미지 서명

다음 단계를 사용하여 컨테이너 이미지에 대한 서명을 생성하고 Amazon ECR 프라이빗 리포지토리에 저장할 수 있습니다. Notation은 다이제스트를 사용하여 이미지에 서명합니다.

### 이미지에 서명

1. 서명 플랫폼을 사용하여 AWS Signer 서명 프로필을 생성합니다. Notation-OCI-SHA384-ECDSA --signature-validity-period 파라미터를 사용하여 서명 유효 기간을 선택적으로 지정할 수 있습니다. 이 값은 DAYS, MONTHS 또는 YEARS를 사용하여 지정할 수 있습니다. 유효 기간을 지정하지 않으면 기본값 135개월이 사용됩니다.

```
aws signer put-signing-profile --profile-name ecr_signing_profile --platform-id
Notation-OCI-SHA384-ECDSA
```

#### Note

서명 프로필 이름에는 영숫자와 밑줄(\_)만 사용할 수 있습니다.

2. Notation 클라이언트를 기본 레지스트리에 인증합니다. 다음 예제는 AWS CLI 를 사용하여 Amazon ECR 사설 레지스트리에 Notation CLI를 인증합니다.

```
aws ecr get-login-password --region region | notation login --username AWS --
password-stdin 111122223333.dkr.ecr.region.amazonaws.com
```

3. Notation CLI를 사용하여 이미지에 서명하고 리포지토리 이름과 SHA 다이제스트를 사용하여 이미지를 지정합니다. 그러면 서명이 생성되고 서명 중인 이미지가 있는 곳과 동일한 Amazon ECR 프라이빗 리포지토리로 푸시됩니다.

다음 예시에서는 SHA 다이제스

트 sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE이(가) 있는 curl 리포지토리의 이미지에 서명하고 있습니다.

**notation**

```
sign 111122223333.dkr.ecr.region.amazonaws.com/
curl@sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE --plugin
"com.amazonaws.signer.notation.plugin" --id "arn:aws:signer:region:111122223333:/
signing-profiles/ecrSigningProfileName"
```

## 다음 단계

컨테이너 이미지에 서명한 후 로컬에서 서명을 확인할 수 있습니다. 이미지 확인에 대한 지침은 AWS Signer 개발자 [안내서에 로그인한 후 로컬에서 이미지 확인](#)을 참조하십시오.

## Amazon ECR 사설 리포지토리에서 서명 삭제

Amazon ECR 사설 리포지토리에서 서명을 삭제할 수 있습니다. Notation CLI를 사용하여 서명을 생성하고 푸시하면 Amazon ECR 리포지토리에 OCI 이미지 인덱스가 생성됩니다. Amazon ECR API는 OCI 이미지 인덱스에서 참조하는 아티팩트 또는 이미지 삭제를 지원하지 않습니다. 이러한 아티팩트를 정리하는 데 사용할 수 있는 옵션은 다음과 같습니다.

- (권장) ORAS CLI를 사용하여 아티팩트를 삭제할 수 있으며 ORAS에서 이미지 인덱스 업데이트 또는 삭제를 처리합니다.
- Amazon ECR API 또는 콘솔을 사용하여 OCI 이미지 인덱스를 먼저 삭제한 다음 서명과 같은 참조 아티팩트를 삭제할 수 있습니다.

ORAS 클라이언트를 사용하여 서명 및 기타 참조 유형 아티팩트를 삭제할 때 ORAS는 OCI 이미지 인덱스를 관리합니다. ORAS는 먼저 인덱스에서 아티팩트에 대한 참조를 제거한 다음 매니페스트를 삭제합니다. 서명 아티팩트의 인덱스를 참조하여 `oras manifest delete` 명령을 사용할 수 있습니다.

ORAS CLI를 사용하여 서명을 삭제하려면

1. ORAS 클라이언트를 설치 및 구성합니다.

ORAS 클라이언트 설치 및 구성에 대한 자세한 내용은 ORAS [설명서의 설치](#)를 참조하십시오.

2. ORAS CLI를 사용하여 서명을 삭제하려면 다음 명령을 실행합니다.

```
oras manifest
delete 111122223333.dkr.ecr.region.amazonaws.com/
repository_name@sha256:ca78e5f730f9a789ef8c63bb55275ac12dfb9e8099e6EXAMPLE
```

## Amazon ECR에서 이미지 세부 정보 보기

이미지를 리포지토리로 푸시한 후 이미지에 대한 정보를 볼 수 있습니다. 포함된 세부 정보는 다음과 같습니다.

- 이미지 URI
- 이미지 태그
- 아티팩트 미디어 유형
- 이미지 매니페스트 유형
- 스캔 상태
- 이미지 크기(MB)
- 이미지가 리포지토리에 푸시된 시간
- 복제 상태

이미지 세부 정보를 보려면(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 이미지를 포함하는 리포지토리가 포함된 리전을 선택합니다.
3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 보려는 리포지토리를 선택합니다.
5. 리포지토리 : **repository\_name** 페이지에서 세부 사항을 볼 이미지를 선택합니다.

## Amazon ECR 사설 리포지토리에서 로컬 환경으로 이미지 가져오기

Amazon ECR에서 사용할 수 있는 Docker 이미지를 실행하려면 `docker pull` 명령을 사용하여 로컬 환경으로 이미지를 가져옵니다. 기본 레지스트리 또는 다른 AWS 계정과 연결된 레지스트리에서 이 작업을 수행할 수 있습니다.

Amazon ECS 작업 정의에서 Amazon ECR 이미지를 사용하려면 [Amazon ECS에서 Amazon ECR 이미지 사용](#) 섹션을 참조하세요.

### ⚠ Important

Amazon ECR의 요구 사항에 따라 사용자가 레지스트리에 대해 인증하고 Amazon ECR 리포지토리에서 이미지를 푸시 또는 풀하기 전에 IAM 정책을 통해 `ecr:GetAuthorizationToken` API에 호출을 할 권한이 있어야 합니다. Amazon ECR은 다양한 수준에서 사용자 액세스를 제어할 수 있는 몇 가지 AWS 관리형 정책을 제공합니다. Amazon ECR의 AWS 관리형 정책에 대한 자세한 내용은 [AWS Amazon Elastic 컨테이너 레지스트리에 대한 관리형 정책](#).

Amazon ECR 리포지토리에서 Docker 이미지를 가져오려면

1. 이미지를 가져오려는 Amazon ECR 레지스트리에 대해 Docker 클라이언트를 인증합니다. 인증 토큰은 사용되는 레지스트리마다 필요하며, 12시간 동안 유효합니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.
2. (선택 사항) 가져올 이미지를 식별합니다.
  - `aws ecr describe-repositories` 명령을 사용하여 레지스트리에 있는 리포지토리 목록을 표시할 수 있습니다.

```
aws ecr describe-repositories
```

위의 예제 레지스트리에는 `amazonlinux`이라는 리포지토리가 있습니다.

- `aws ecr describe-images` 명령을 사용하여 리포지토리 내에 있는 이미지 목록을 표시할 수 있습니다.

```
aws ecr describe-images --repository-name amazonlinux
```

위의 예제 리포지토리에는 이미지 다이제스트 `latest`과 함께 `2016.09` 및 `sha256:f1d4ae3f7261a72e98c6ebefe9985cf10a0ea5bd762585a43e0700ed99863807`라고 태그가 지정된 이미지가 있습니다.

3. `docker pull` 명령을 사용하여 이미지를 풀링합니다. 이미지 이름 형식은 태그를 기준으로 가져오는 경우 `registry/repository[:tag]`, 다이제스트를 기준으로 가져오는 경우 `registry/repository[@digest]`입니다.

```
docker pull aws_account_id.dkr.ecr.us-west-2.amazonaws.com/amazonlinux:latest
```

### Important

*repository-url* not found: does not exist or no pull access 오류가 표시되는 경우 Amazon ECR로 Docker 클라이언트를 인증해야 할 수 있습니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.

## 아마존 리눅스 컨테이너 이미지 가져오기

Amazon Linux 컨테이너 이미지는 Amazon Linux AMI에 포함된 동일한 소프트웨어 구성 요소로부터 빌드됩니다. Amazon Linux 컨테이너 이미지는 모든 환경에서 Docker 워크로드의 기본 이미지로 사용할 수 있습니다. Amazon EC2의 애플리케이션에 Amazon Linux AMI를 사용하는 경우 Amazon Linux 컨테이너 이미지를 사용하여 애플리케이션을 컨테이너화할 수 있습니다.

로컬 개발 환경에서 Amazon Linux 컨테이너 이미지를 AWS 사용한 다음 Amazon ECS를 사용하도록 애플리케이션을 푸시할 수 있습니다. 자세한 정보는 [Amazon ECS에서 Amazon ECR 이미지 사용](#)을 참조하세요.

Amazon Linux 컨테이너 이미지는 Amazon ECR Public 및 [Docker Hub](#)에서 사용할 수 있습니다. Amazon Linux 컨테이너 이미지에 대한 지원을 받으려면 [AWS 개발자 포럼을 방문하십시오](#).

Amazon ECR Public에서 Amazon Linux 컨테이너 이미지를 가져오려면

1. Docker 클라이언트를 Amazon Linux Public 레지스트리에 인증합니다. 인증 토큰은 12시간 동안 유효합니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.

### Note

ecr-public 명령은 버전 1.18.1.187 이상의 AWS CLI 에서 사용할 수 있지만, 최신 버전의 AWS CLI를 사용하는 것이 좋습니다. 자세한 내용을 알아보려면 AWS Command Line Interface 사용자 가이드에서 [AWS Command Line Interface설치](#)를 참조하세요.

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws
```

출력값은 다음과 같습니다.

```
Login succeeded
```

2. `docker pull` 명령을 사용하여 Amazon Linux 컨테이너 이미지를 가져옵니다. Amazon ECR Public Gallery에서 Amazon Linux 컨테이너 이미지를 보려면 [Amazon ECR Public Gallery - amazonlinux](#)를 참조하세요.

```
docker pull public.ecr.aws/amazonlinux/amazonlinux:latest
```

3. (선택 사항) 로컬에서 컨테이너를 실행합니다.

```
docker run -it public.ecr.aws/amazonlinux/amazonlinux /bin/bash
```

Docker Hub에서 Amazon Linux 컨테이너 이미지를 가져오려면

1. `docker pull` 명령을 사용하여 Amazon Linux 컨테이너 이미지를 가져옵니다.

```
docker pull amazonlinux
```

2. (선택 사항) 로컬에서 컨테이너를 실행합니다.

```
docker run -it amazonlinux:latest /bin/bash
```

## Amazon ECR에서 이미지 삭제

이미지 사용을 마치면 리포지토리에서 이를 삭제할 수 있습니다. 리포지토리 사용을 마치면 전체 리포지토리 및 리포지토리 내부의 이미지를 모두 삭제할 수 있습니다. 자세한 정보는 [Amazon ECR에서 프라이빗 리포지토리 삭제](#)을 참조하세요.

이미지를 수동으로 삭제하는 대신 리포지토리에 있는 이미지의 수명 주기 관리를 보다 효과적으로 제어할 수 있는 저장소 수명 주기 정책을 만들 수 있습니다. 수명 주기 정책은 이 프로세스를 자동화합니다. 자세한 정보는 [Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.](#)을 참조하세요.

## 이미지를 삭제하려면(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 삭제할 이미지가 들어 있는 리전을 선택합니다.
3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 삭제할 이미지가 들어 있는 리포지토리를 선택합니다.
5. 리포지토리: **repository\_name** 페이지에서 삭제할 이미지의 왼쪽에 있는 상자를 선택하고 삭제 (Delete)를 선택합니다.
6. 이미지 삭제>Delete image(s)) 대화 상자에서 삭제할 이미지가 선택되었는지 확인한 후 삭제 (Delete)를 선택합니다.

## 이미지를 삭제하려면(AWS CLI)

1. 리포지토리에 있는 이미지를 나열합니다. 태그가 지정된 이미지는 이미지 다이제스트와 관련 태그 목록을 모두 갖습니다. 태그가 지정되지 않은 이미지에는 이미지 다이제스트만 있습니다.

```
aws ecr list-images \
  --repository-name my-repo
```

2. (선택 사항) 삭제하려는 이미지와 연결된 태그를 지정하여 이미지에 대해 원치 않는 태그를 삭제합니다. 이미지에서 마지막 태그를 삭제하면 이미지도 삭제됩니다.

```
aws ecr batch-delete-image \
  --repository-name my-repo \
  --image-ids imageTag=tag1 imageTag=tag2
```

3. 이미지 다이제스트를 지정하여 태그가 지정되었거나 지정되지 않은 이미지를 삭제합니다. 다이제스트를 참조하여 이미지를 삭제하면 이미지와 이미지의 태그 모두가 삭제됩니다.

```
aws ecr batch-delete-image \
  --repository-name my-repo \
  --image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
```

여러 이미지를 삭제하려면 요청에서 여러 이미지 태그 또는 이미지 다이제스트를 지정할 수 있습니다.

```
aws ecr batch-delete-image \
  --repository-name my-repo \
```

```
--image-ids imageDigest=sha256:4f70ef7a4d29e8c0c302b13e25962d8f7a0bd304EXAMPLE
imageDigest=sha256:f5t0e245ssffc302b13e25962d8f7a0bd304EXAMPLE
```

## Amazon ECR에서 이미지 태그 재지정

Docker Image Manifest V2 Schema 2 이미지를 사용하면 `put-image` 명령의 `--image-tag` 옵션을 사용하여 기존 이미지에 태그를 다시 지정할 수 있습니다. Docker를 사용하여 이미지를 가져오거나 푸시하지 않고도 태그를 다시 지정할 수 있습니다. 크기가 큰 이미지의 경우 이렇게 하면 이미지에 태그를 다시 지정하는 데 드는 시간과 네트워크 대역폭을 크게 절약할 수 있습니다.

### 이미지에 태그를 다시 지정하려면(AWS CLI)

를 사용하여 이미지에 태그를 다시 지정하려면 AWS CLI

1. `batch-get-image` 명령을 사용하여 태그를 다시 지정할 이미지에 대한 이미지 매니페스트를 가져와 파일에 작성합니다. 이 예제에서는 리포지토리에서 `##` 태그가 있는 이미지의 매니페스트, `amazonlinux`가 `MANIFEST`라는 이름의 환경 변수에 작성됩니다.

```
MANIFEST=$(aws ecr batch-get-image --repository-name amazonlinux --image-ids
imageTag=latest --output text --query 'images[].imageManifest')
```

2. `put-image` 명령의 `--image-tag` 옵션을 사용하여 새로운 태그가 지정된 이미지 매니페스트를 Amazon ECR에 넣습니다. 이 예에서는 이미지가 `2017.03`로 태그 지정되어 있습니다.

#### Note

사용 중인 버전에서 `--image-tag` 옵션을 사용할 수 없는 경우 최신 버전으로 업그레이드하십시오. AWS CLI 자세한 내용은 AWS Command Line Interface 사용 설명서의 [설치 AWS Command Line Interface](#)를 참조하세요.

```
aws ecr put-image --repository-name amazonlinux --image-tag 2017.03 --image-manifest "$MANIFEST"
```

3. 새로운 이미지 태그가 이미지에 연결되어 있는지 확인합니다. 아래의 출력에서 이미지에 태그 `latest`와 `2017.03`가 있습니다.

```
aws ecr describe-images --repository-name amazonlinux
```



출력값은 다음과 같습니다.

```
{
  "imageDetails": [
    {
      "imageSizeInBytes": 98755613,
      "imageDigest":
"sha256:8d00af8f076eb15a33019c2a3e7f1f655375681c4e5be157a26EXAMPLE",
      "imageTags": [
        "latest",
        "2017.03"
      ],
      "registryId": "aws_account_id",
      "repositoryName": "amazonlinux",
      "imagePushedAt": 1499287667.0
    }
  ]
}
```

## 이미지에 태그를 다시 지정하려면(AWS Tools for Windows PowerShell)

를 사용하여 이미지에 태그를 다시 지정하려면 AWS Tools for Windows PowerShell

1. Get-ECRImageBatch cmdlet을 사용하여 태그를 다시 지정할 이미지의 설명을 가져온 다음 이를 환경 변수에 씁니다. 이 예제에서는 리포지토리에서 ## 태그가 있는 이미지, *amazonlinux*가 환경 변수 *\$Image*에 쓰여집니다.

### Note

시스템에서 Get-ECRImageBatch cmdlet을 사용할 수 없는 경우, AWS Tools for Windows PowerShell 사용 설명서의 [AWS Tools for Windows PowerShell 설정](#)을 참조합니다.

```
$Image = Get-ECRImageBatch -ImageId @{ imageTag="latest" } -  
RepositoryName amazonlinux
```

2. 이미지의 매니페스트를 *\$Manifest* 환경 변수에 씁니다.

```
$Manifest = $Image.Images[0].ImageManifest
```

3. Write-ECRImage cmdlet의 -ImageTag 옵션을 사용하여 이미지 매니페스트를 새로운 태그와 Amazon ECR에 넣습니다. 이 예에서는 이미지가 **2017.09**로 태그 지정되어 있습니다.

```
Write-ECRImage -RepositoryName amazonlinux -ImageManifest $Manifest -
ImageTag 2017.09
```

4. 새로운 이미지 태그가 이미지에 연결되어 있는지 확인합니다. 아래의 출력에서 이미지에 태그 latest와 2017.09가 있습니다.

```
Get-ECRImage -RepositoryName amazonlinux
```

출력값은 다음과 같습니다.

ImageDigest	ImageTag
-----	-----
sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497	latest
sha256:359b948ea8866817e94765822787cd482279eed0c17bc674a7707f4256d5d497	2017.09

## Amazon ECR에서 이미지 태그의 덮어쓰기 방지

리포지토리에서 태그 불변성을 활성화하여 이미지 태그가 덮어쓰여지는 것을 방지할 수 있습니다. 태그 불변성을 설정한 후 저장소에 이미 있는 태그가 포함된 이미지를 푸시하면 ImageTagAlreadyExistsException 오류가 반환됩니다. 태그 불변성은 모든 태그에 영향을 미칩니다. 일부 태그는 변경할 수 없고 다른 태그는 변경할 수 없게 만들 수 없습니다.

AWS Management Console 및 AWS CLI 도구를 사용하여 새 리포지토리 또는 기존 리포지토리의 이미지 태그 변경 가능성을 설정할 수 있습니다. 콘솔 단계를 사용하여 리포지토리를 만들려면 [참조하십시오](#) [이 이미지를 저장하기 위한 Amazon ECR 사설 리포지토리 생성](#).

### 이미지 태그 변경 가능성 설정 ()AWS Management Console

이미지 태그 가변성을 설정하려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 편집할 리포지토리가 포함된 리전을 선택합니다.

3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 프라이빗(Private) 탭을 선택한 다음 편집할 리포지토리를 선택하고 편집(Edit)을 선택합니다.
5. 태그 불변성(Tag immutability)에서 리포지토리의 태그 변경 가능 설정을 선택합니다. 변경 불가능 태그로 구성된 리포지토리는 이미지 태그를 덮어쓰는 것을 방지해 줍니다. 자세한 정보는 [Amazon ECR에서 이미지 태그의 덮어쓰기 방지](#)를 참조하세요.
6. 이미지 스캔 설정(Image scan settings)의 경우 기본 스캔을 위해 리포지토리 수준에서 스캔 설정을 지정할 수 있지만 프라이빗 레지스트리 수준에서 스캔 구성을 지정하는 것이 가장 좋습니다. 프라이빗 레지스트리에서 스캔 설정을 지정하면 고급 스캔 또는 기본 스캔을 사용하고 필터를 정의하여 스캔할 리포지토리를 지정할 수 있습니다. 자세한 정보는 [이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다.](#)를 참조하세요.
7. 암호화 설정(Encryption settings)의 경우 리포지토리가 생성되면 리포지토리에 대한 암호화 설정을 변경할 수 없으므로 이 항목은 보기 전용 필드입니다.
8. 저장(Save)을 선택하여 리포지토리 설정을 업데이트합니다.

## 이미지 태그 가변성 설정 (AWS CLI)

변경 불가능 태그로 구성된 리포지토리를 생성하려면

다음 명령 중 하나를 사용하여 변경 불가능 태그로 구성된 새 이미지 리포지토리를 생성합니다.

- [create-repository](#)(AWS CLI)

```
aws ecr create-repository --repository-name name --image-tag-mutability IMMUTABLE --region us-east-2
```

- [New-ECRRepository](#)(AWS Tools for Windows PowerShell)

```
New-ECRRepository -RepositoryName name -ImageTagMutability IMMUTABLE -Region us-east-2 -Force
```

리포지토리의 이미지 태그 변경 가능성 설정을 업데이트하려면

다음 명령 중 하나를 사용하여 기존 리포지토리의 이미지 태그 변경 가능성 설정을 업데이트합니다.

- [put-image-tag-mutability](#)(AWS CLI)

```
aws ecr put-image-tag-mutability --repository-name name --image-tag-
mutability IMMUTABLE --region us-east-2
```

- [쓰기 ImageTag ECR](#) 변경 가능성 ()AWS Tools for Windows PowerShell

```
Write-ECRImageTagMutability -RepositoryName name -ImageTagMutability IMMUTABLE -
Region us-east-2 -Force
```

## Amazon ECR의 컨테이너 이미지 매니페스트 형식 지원

Amazon ECR는 다음과 같은 컨테이너 이미지 매니페스트 형식을 지원합니다.

- Docker 이미지 매니페스트 V2 스키마 1(Docker 버전 1.9 이하에서 사용됨)
- 도커 이미지 매니페스트 V2 스키마 2(Docker 버전 1.10 이상에서 사용됨)
- Open Container Initiative(OCI) 사양(v1.0 이상)

도커 이미지 매니페스트 V2 스키마 2를 지원하면 다음 기능을 사용할 수 있습니다.

- 단일 이미지에 여러 개의 태그 사용 기능입니다.
- Windows 컨테이너 이미지 저장 지원.

## Amazon ECR 이미지 매니페스트 전환

Amazon ECR에 이미지를 푸시하고 가져오면, 컨테이너 엔진 클라이언트(예: Docker)가 레지스트리와 통신하여 이미지에 사용할 레지스트리 및 클라이언트가 인식할 수 있는 매니페스트 형식으로 일치시킵니다.

Docker 버전 1.9 이하를 사용하여 Amazon ECR에 이미지를 푸시하는 경우, Docker 이미지 매니페스트 V2 스키마 1 형식으로 이미지 매니페스트 형식이 저장됩니다. Docker 버전 1.10 이상을 사용하여 Amazon ECR에 이미지를 푸시하는 경우, Docker 이미지 매니페스트 V2 스키마 2 형식으로 이미지 매니페스트 형식이 저장됩니다.

태그로 Amazon ECR에서 이미지를 가져오는 경우 Amazon ECR은 리포지토리에 저장된 이미지 매니페스트 형식을 반환합니다. 해당 형식을 클라이언트에서 인식할 수 있는 경우에만 형식이 반환됩니다. 저장된 이미지 매니페스트 형식을 클라이언트에서 인식하지 못하면 Amazon ECR은 이미지 매니페스트를 클라이언트에서 인식하는 형식으로 변환합니다. 예를 들어 Docker 1.9 클라이언트는 Docker 이

이미지 매니페스트 V2 Schema 2로 저장되는 이미지 매니페스트를 요청하고, Amazon ECR은 Docker 이미지 매니페스트 V2 스키마 1 형식으로 매니페스트를 반환합니다. 다음 표에서는 태그로 이미지를 가져올 때 Amazon ECR에서 지원하는 사용 가능한 변환에 대해 설명합니다.

클라이언트에서 요청하는 스키마	V2 스키마 1 형식으로 ECR에 푸시함	V2 스키마 2 형식으로 ECR에 푸시함	OCI 형식으로 ECR에 푸시함
V2 스키마 1	변환이 필요하지 않음	V2 스키마 1 형식으로 변환됨	V2 스키마 1 형식으로 변환됨
V2 스키마 2	변환 불가능, 클라이언트가 V2 스키마 1로 폴백됨	변환이 필요하지 않음	V2 스키마 2 형식으로 변환됨
OCI	변환 불가능	OCI 형식으로 변환됨	변환이 필요하지 않음

#### Important

이미지를 다이제스트로 가져오는 경우, 사용할 수 있는 변환이 없습니다. 클라이언트는 Amazon ECR에 저장된 이미지 매니페스트 형식을 이해하고 있어야 합니다. 사용자가 Docker 1.9 이하 클라이언트에서 다이제스트를 기준으로 Docker 이미지 매니페스트 V2 스키마 2를 요청하는 경우 이미지 가져오기가 실패합니다. 자세한 내용은 Docker 설명서의 [레지스트리 호환성](#)을 참조하세요.

이 예에서 태그로 동일한 이미지를 요청하는 경우, Amazon ECR에서 클라이언트에서 인식할 수 있는 형식으로 이미지 매니페스트를 변환합니다. 그리고 이미지 가져오기가 성공합니다.

## Amazon ECS에서 Amazon ECR 이미지 사용

Amazon ECR 프라이빗 리포지토리를 사용하여 Amazon ECS 태스크에서 가져올 수 있는 컨테이너 이미지 및 아티팩트를 호스팅할 수 있습니다. 이 항목이 작동하려면 Amazon ECS 또는 Fargate, 컨테이너 에이전트에 `ecr:BatchGetImage`, `ecr:GetDownloadUrlForLayer` 및 `ecr:GetAuthorizationToken` API를 만들 수 있는 권한이 있어야 합니다.

## 필수 IAM 권한

다음 표에는 Amazon ECR 프라이빗 리포지토리에서 태스크를 가져오는 데 필요한 권한을 제공하는 각 시작 유형에 사용할 IAM 역할이 표시되어 있습니다. Amazon ECS는 필요한 권한이 포함된 관리형 IAM 정책을 제공합니다.

시작 유형	IAM 역할	AWS 관리형 IAM 정책
Amazon EC2 인스턴스의 Amazon ECS	Amazon ECS 클러스터에 등록된 Amazon EC2 인스턴스와 연결된 컨테이너 인스턴스 IAM 역할을 사용합니다. 자세한 내용을 알아보려면 Amazon Elastic Container Service 개발자 안내서의 <a href="#">컨테이너 인스턴스 IAM 역할</a> 을 참조하세요.	AmazonEC2ContainerServiceforEC2Role  자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 <a href="#">AmazonEC2ContainerServiceforEC2Role</a> 섹션을 참조하세요.
Fargate의 Amazon ECS	Amazon ECS 태스크 정의에서 참조하는 태스크 실행 IAM 역할을 사용하세요. 자세한 내용을 알아보려면 Amazon Elastic Container Service 개발자 안내서의 <a href="#">태스크 실행 IAM 역할</a> 을 참조하세요.	AmazonECSTaskExecutionRolePolicy  자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 <a href="#">AmazonECSTaskExecutionRolePolicy</a> 섹션을 참조하세요.
외부 인스턴스의 Amazon ECS	Amazon ECS 클러스터에 등록된 온프레미스 서버 또는 가상 머신(VM)과 연결된 컨테이너 인스턴스 IAM 역할을 사용하세요. 자세한 내용을 알아보려면 Amazon Elastic Container Service 개발자 안내서의 <a href="#">컨테이너 인스턴스 Amazon ECS 역할</a> 을 참조하세요.	AmazonEC2ContainerServiceforEC2Role  자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 <a href="#">AmazonEC2ContainerServiceforEC2Role</a> 섹션을 참조하세요.

**⚠ Important**

AWS 관리형 IAM 정책에는 사용에 필요하지 않을 수 있는 추가 권한이 포함되어 있습니다. 이 경우 이 항목들은 Amazon ECR 프라이빗 리포지토리에서 가져오는 데 필요한 최소 권한입니다.

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

## Amazon ECS 태스크 정의에서 Amazon ECR 이미지 지정

Amazon ECS 태스크 정의를 만들 때 Amazon ECR 프라이빗 리포지토리에서 호스팅되는 컨테이너 이미지를 지정할 수 있습니다. 태스크 정의에서 Amazon ECR 이미지에 대해 전체 `registry/repository:tag` 이름 지정을 사용하고 있는지 확인하세요. 예를 들어 `aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:latest`입니다.

다음 작업 정의 조각은 Amazon ECS 작업 정의의 Amazon ECR에서 호스트되는 컨테이너 이미지를 지정하는 데 사용할 구문을 보여줍니다.

```
{
  "family": "task-definition-name",
  ...
  "containerDefinitions": [
    {
      "name": "container-name",
      "image": "aws_account_id.dkr.ecr.region.amazonaws.com/my-
repository:latest",
```

```

    ...
  }
],
...
}

```

## Amazon EKS에서 Amazon ECR 이미지 사용

Amazon ECR 이미지를 Amazon EKS와 함께 사용할 수 있습니다.

Amazon ECR에서 이미지를 참조할 때는 이미지에 대해 전체 `registry/repository:tag` 이름 지정을 사용해야 합니다. 예: `aws_account_id.dkr.ecr.region.amazonaws.com/my-repository:latest`

### 필수 IAM 권한

Amazon EKS 워크로드를 관리형 노드 또는 AWS Fargate 자체 관리형 노드에서 호스팅하는 경우 다음을 검토하십시오.

- 관리형 또는 자체 관리형 노드에서 호스팅되는 Amazon EKS 워크로드: Amazon EKS 작업자 노드 IAM 역할 () 이 필요합니다. NodeInstanceRole Amazon EKS 작업자 노드 IAM 역할에는 Amazon ECR에 대해 다음의 IAM 정책 권한이 포함되어야 합니다.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    }
  ]
}

```



**Note**

Amazon EKS Getting [Starting of Amazon EKS의 AWS CloudFormation](#) 템플릿을 사용하여 eksctl 클러스터와 작업자 노드 그룹을 생성한 경우 이러한 IAM 권한은 기본적으로 작업자 노드 IAM 역할에 적용됩니다.

- 호스팅되는 Amazon EKS 워크로드 AWS Fargate: 프라이빗 Amazon ECR 리포지토리에서 이미지를 가져올 수 있는 권한을 파드에 제공하는 Fargate 포드 실행 역할을 사용하십시오. 자세한 내용은 [Fargate 포드 실행 역할 만들기](#)를 참조하세요.

## Amazon EKS 클러스터에 헬름 차트 설치

Amazon ECR에서 호스팅되는 헬름 차트는 Amazon EKS 클러스터에 설치할 수 있습니다.

### 사전 조건

- Helm 클라이언트의 최신 버전을 설치합니다. 이 단계는 Helm 버전을 사용하여 작성되었습니다 3.9.0. 자세한 정보는 [Helm 설치](#)를 참조하세요.
- 최소한 AWS CLI 의 1.23.9 또는 2.6.3 버전이 컴퓨터에 설치되어 있어야 합니다. 자세한 내용은 [최신 버전의 AWS CLI 설치 또는 업데이트](#)를 참조하세요.
- Amazon ECR 리포지토리에 Helm 차트를 푸시했습니다. 자세한 내용은 [헬름 차트를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#)(를) 참조하세요.
- Amazon EKS로 작업하기 위해 kubectl를 구성했습니다. 자세한 내용은 Amazon EKS 사용 설명서의 [Amazon EKS의 kubeconfig 생성](#)을 참조하세요. 클러스터에 대해 다음 명령이 성공한 경우 적절하게 구성한 것입니다.

```
kubectl get svc
```

Amazon EKS 클러스터에 헬름 차트를 설치하려면

- Helm 차트가 호스트되는 Amazon ECR 레지스트리에 Helm 클라이언트를 인증합니다. 인증 토큰은 사용되는 레지스트리마다 필요하며, 12시간 동안 유효합니다. 자세한 정보는 [Amazon ECR에서의 프라이빗 레지스트리 인증](#)을 참조하세요.

```
aws ecr get-login-password \
  --region us-west-2 | helm registry login \
```

```
--username AWS \  
--password-stdin aws_account_id.dkr.ecr.region.amazonaws.com
```

2. 차트를 설치합니다. 리포지토리로, *0.1.0#* 헬름 차트 태그로 *helm-test-chart* 바꾸십시오.

```
helm install ecr-chart-demo oci://aws_account_id.dkr.ecr.region.amazonaws.com/helm-test-chart --version 0.1.0
```

결과가 다음과 비슷할 것입니다.

```
NAME: ecr-chart-demo  
LAST DEPLOYED: Tue May 31 17:38:56 2022  
NAMESPACE: default  
STATUS: deployed  
REVISION: 1  
TEST SUITE: None
```

3. 차트 설치를 확인합니다.

```
helm list -n default
```

출력 예제:

NAME	NAMESPACE	REVISION	UPDATED
STATUS	CHART	APP VERSION	
ecr-chart-demo	default	1	2022-06-01 15:56:40.128669157 +0000
UTC deployed	helm-test-chart-0.1.0	1.16.0	

4. (선택 사항) 설치된 Helm 차트 ConfigMap을 참조하세요.

```
kubectl describe configmap helm-test-chart-configmap
```

5. 완료되면 클러스터에서 차트 릴리스를 제거할 수 있습니다.

```
helm uninstall ecr-chart-demo
```

## 이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다.

향상된 기본 스캔 기능은 Amazon ECR의 프리뷰 릴리즈 단계에 있으며 변경될 수 있습니다. 이 공개 미리 보기 동안에는 향상된 기본 스캔 버전을 AWS Management Console 옵트인하는 데만 사용할 수 있습니다.

Amazon ECR 이미지 스캔은 컨테이너 이미지의 소프트웨어 취약성을 식별하는 데 도움이 됩니다. 다음과 같은 스캔 유형이 제공됩니다.

### Important

향상된 스캔, 기본 스캔 및 개선된 기본 스캔 버전을 전환하면 이전에 설정된 스캔을 더 이상 사용할 수 없게 됩니다. 스캔을 다시 설정해야 합니다. 하지만 이전 스캔 버전으로 다시 전환하면 기존 스캔을 사용할 수 있습니다.

- 고급 스캔(Enhanced scanning) - Amazon ECR이 Amazon Inspector와 통합되어 리포지토리를 계속 해서 자동 스캔할 수 있는 기능을 제공합니다. 운영 체제 및 프로그래밍 언어 패키지 취약성 모두에 대해 컨테이너 이미지가 스캔됩니다. 새로운 취약성이 나타나면 스캔 결과가 업데이트되고 Amazon Inspector에서 이벤트를 EventBridge 생성하여 사용자에게 알립니다. 향상된 스캔은 다음을 제공합니다.
  - OS 및 프로그래밍 언어 패키지 취약성.
  - 두 가지 스캔 빈도: 푸시 스캔 및 연속 스캔.
- 기본 검사 —Amazon ECR은 일반적인 취약성 및 노출 (CVE) 데이터베이스를 사용하는 두 가지 버전의 기본 검사를 제공합니다. 하나는 오픈 소스 Clair 프로젝트를 사용하는 최신 GA 버전이고 다른 하나는 기본 기술을 사용하는 새롭게 개선된 기본 검사 버전 (미리 보기) 입니다. AWS 기본 스캔을 사용하면 푸시할 때 스캔하도록 리포지토리를 구성하거나 수동 스캔을 수행할 수 있으며, Amazon ECR에서 스캔 결과 목록을 제공합니다. 기본 스캔은 다음을 제공합니다.
  - OS 스캔.
  - 두 가지 스캔 주파수: 수동 및 푸시 스캔.

**⚠ Important**

새 버전의 기본 스캔은 `imageScanStatus DescribeImages` API에서 `imageScanFindingsSummary AND`를 지원하지 않습니다. 이를 보려면 `DescribeImageScanFindings` API를 사용하세요.

## Amazon ECR에서 스캔할 리포지토리를 선택하는 필터

사실 레지스트리에 대한 이미지 스캔을 구성할 때 필터를 사용하여 스캔할 리포지토리를 선택할 수 있습니다.

기본 스캔을 사용하는 경우 푸시 필터에서 스캔을 지정하여 새 이미지를 푸시할 때 이미지 스캔을 수행하도록 설정되는 리포지토리를 지정할 수 있습니다. 푸시 필터의 기본 스캔과 일치하지 않는 모든 리포지토리는 수동 스캔 빈도로 설정됩니다. 즉, 스캔을 수행하려면 수동으로 스캔을 트리거해야 합니다.

고급 스캔을 사용하는 경우 푸시 및 연속 스캔 시 스캔을 위한 별도의 필터를 지정할 수 있습니다. 고급 스캔 필터와 일치하지 않는 리포지토리는 스캔이 비활성화됩니다. 고급 스캔을 사용하며 여러 필터가 동일한 리포지토리와 일치하는 푸시 및 연속 스캔 시 스캔을 위한 별도의 필터를 지정하는 경우, Amazon ECR은 해당 리포지토리의 푸시 필터 스캔에 연속 스캔 필터를 적용합니다.

### 와일드카드 필터링

필터를 지정하면 와일드카드가 없는 필터는 필터를 포함하는 모든 리포지토리 이름과 일치합니다. 와일드카드가 있는 필터(\*)는 와일드카드가 리포지토리 이름에서 0개 이상의 문자를 대체하는 리포지토리 이름과 일치합니다.

다음 테이블에서는 리포지토리 이름이 가로축에 표시되고 세로축에 예제 필터가 지정되는 예제를 보여줍니다.

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
prod	예	예	예	예	예
*prod	예	예	아니요	아니요	아니요
prod*	예	아니요	예	아니요	예

	prod	repo-prod	prod-repo	repo-prod-repo	prodrepo
*prod*	예	예	예	예	예
prod*repo	아니요	아니요	예	아니요	예

## Amazon ECR의 OS 및 프로그래밍 언어 패키지 취약성이 있는지 이미지를 스캔합니다.

Amazon ECR 고급 스캔은 Amazon Inspector와 통합되어 컨테이너 이미지에 대한 취약성 스캔 기능을 제공합니다. 운영 체제 및 프로그래밍 언어 패키지 취약성 모두에 대해 컨테이너 이미지가 스캔됩니다. Amazon ECR과 Amazon Inspector 모두를 사용하여 스캔 결과를 직접 확인할 수 있습니다. Amazon Inspector에 대한 자세한 내용은 Amazon Inspector 사용 설명서의 [Amazon Inspector로 컨테이너 이미지 스캔](#)을 참조하세요.

고급 스캔을 사용하면 자동 연속 스캔을 위해 구성된 리포지토리와 푸시할 때 스캔하도록 구성된 리포지토리를 선택할 수 있습니다. 이 작업은 스캔 필터를 설정하여 수행됩니다.

## 고급 스캔 고려 사항

Amazon ECR 고급 스캔을 활성화하기 전에 다음 사항을 고려하십시오.

- Amazon ECR에서 이 기능을 사용하는 데 드는 추가 비용은 없지만 이미지를 스캔하려면 Amazon Inspector에서 비용을 지불해야 합니다. 자세한 내용은 [Amazon Inspector 요금](#)을 참조하세요.
- 다음 리전에서는 고급 스캔이 지원되지 않습니다.
  - 중동(UAE)(me-central-1)
  - 아시아 태평양(하이데라바드)(ap-south-2)
  - 이스라엘(텔아비브)(il-central-1)
  - 아시아 태평양(멜버른) (ap-southeast-4)
  - 유럽(스페인)(eu-south-2)
- Amazon Inspector는 특정 운영 체제에 대한 스캔을 지원합니다. 전체 목록은 Amazon Inspector 사용 설명서의 [지원되는 운영 체제 - Amazon ECR 스캔](#)을 참조하세요.
- Amazon Inspector는 리포지토리에 대한 고급 스캔 기능을 제공하는 데 필요한 권한을 지원하는 서비스 연결 IAM 역할을 사용합니다. 프라이빗 레지스트리에 대해 고급 스캔을 켜면 Amazon

Inspector에서 서비스 연결 IAM 역할이 자동으로 생성됩니다. 자세한 내용은 Amazon Inspector 사용 설명서의 [Amazon Inspector에 대한 서비스 연결 역할 사용](#)을 참조하세요.

- 프라이빗 레지스트리에 대한 향상된 스캔 기능을 처음 켜면 Amazon Inspector는 이미지 푸시 타임 스탬프를 기반으로 지난 30일 동안 Amazon ECR에 푸시되거나 지난 90일 동안 가져온 이미지만 인식합니다. 이전 이미지는 SCAN\_ELIGIBILITY\_EXPIRED 스캔 상태가 포함됩니다. Amazon Inspector에서 이러한 이미지를 스캔하려는 경우에는 이미지를 해당 리포지토리로 다시 푸시해야 합니다.
- 고급 스캔이 켜진 후에 Amazon ECR에 푸시된 모든 이미지는 구성된 기간 동안 지속적으로 스캔됩니다. 기간은 기본적으로 전체 기간으로 설정됩니다. 이 설정은 Amazon Inspector 콘솔을 사용하여 구성할 수 있습니다. 자세한 정보는 [Amazon Inspector에서 이미지의 향상된 스캔 지속 시간 변경](#)을 참조하세요.
- Amazon ECR 프라이빗 레지스트리에 대해 고급 검색이 켜져 있으면 검색 필터와 일치하는 리포지토리가 고급 검색만 사용하여 스캔됩니다. 필터와 일치하지 않는 리포지토리에는 Off 스캔 빈도가 설정되며 스캔되지 않습니다. 고급 스캔을 사용한 수동 스캔은 지원되지 않습니다. 자세한 정보는 [Amazon ECR에서 스캔할 리포지토리를 선택하는 필터](#)을 참조하세요.
- 여러 필터가 동일한 리포지토리와 일치하는 푸시 및 연속 스캔 시 스캔을 위한 별도의 필터를 지정하는 경우, Amazon ECR은 해당 리포지토리의 푸시 필터 스캔에 연속 스캔 필터를 적용합니다.
- 향상된 스캔이 켜져 있는 경우 Amazon ECR은 리포지토리의 스캔 빈도가 변경될 EventBridge 때 이벤트를 전송합니다. Amazon Inspector는 초기 스캔이 완료되고 이미지 스캔 결과가 생성, 업데이트 또는 달릴 때 이벤트를 내보냅니다. EventBridge

## Amazon ECR에서의 향상된 스캔을 위해 필요한 IAM 권한

Amazon ECR 고급 스캔에는 Amazon Inspector 서비스 연결 IAM 역할이 필요하며, 고급 스캔을 활성화 및 사용하는 IAM 주체에는 스캔에 필요한 Amazon Inspector API를 호출할 권한이 있습니다. 프라이빗 레지스트리에 대해 고급 스캔을 켜면 Amazon Inspector에서 Amazon Inspector 서비스 연결 IAM 역할이 자동으로 생성됩니다. 자세한 내용은 Amazon Inspector 사용 설명서의 [Amazon Inspector에 대한 서비스 연결 역할 사용](#)을 참조하세요.

다음 IAM 정책은 고급 스캔을 활성화하고 사용하기 위한 필수 권한을 부여합니다. 여기에는 Amazon Inspector가 서비스 연결 IAM 역할을 생성하는 데 필요한 권한과 고급 스캔을 켜거나 끄고 스캔 결과를 검색하는 데 필요한 Amazon Inspector API 권한이 포함되어 있습니다.

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

```

    {
      "Effect": "Allow",
      "Action": [
        "inspector2:Enable",
        "inspector2:Disable",
        "inspector2:ListFindings",
        "inspector2:ListAccountPermissions",
        "inspector2:ListCoverage"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": [
            "inspector2.amazonaws.com"
          ]
        }
      }
    }
  ]
}

```

## Amazon ECR에서 이미지에 대한 향상된 스캔 구성

사실 레지스트리의 지역별 향상된 검사를 구성하십시오.

향상된 검사를 구성할 수 있는 적절한 IAM 권한이 있는지 확인하십시오. 자세한 내용은 [Amazon ECR에서의 향상된 스캔을 위해 필요한 IAM 권한](#)을 참조하세요.

### AWS Management Console

프라이빗 레지스트리에 대한 향상된 검사 기능을 켜려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 스캔 구성을 설정할 리전을 선택합니다.
3. 탐색 창에서 개인 레지스트리, 설정, 스캔을 선택합니다.
4. 스캔 구성(Scanning configuration) 페이지의 스캔 유형(Scan type)에서 고급 스캔(Enhanced scanning)을 선택합니다.

기본적으로 향상된 검사를 선택하면 모든 리포지토리가 계속 스캔됩니다.

5. 연속 스캔할 특정 리포지토리를 선택하려면 모든 리포지토리 연속 스캔 확인란의 선택을 취소한 다음 필터를 정의하십시오.

#### Important

와일드카드가 없는 필터는 필터를 포함하는 모든 리포지토리 이름과 일치합니다. 와일드카드가 있는 필터(\*)는 와일드카드가 리포지토리 이름에서 0개 이상의 문자를 대체하는 리포지토리 이름과 일치합니다. 필터 작동 방식의 예를 보려면 [이 섹션](#)을 참조하십시오. [the section called “와일드카드 필터링”](#)

- a. 리포지토리 이름을 기반으로 필터를 입력한 다음 필터 추가를 선택합니다.
  - b. 이미지를 푸시할 때 스캔할 리포지토리를 결정하십시오.
    - 푸시 시 모든 리포지토리를 스캔하려면 모든 리포지토리를 푸시할 때 스캔을 선택합니다.
    - 푸시 시 스캔할 특정 리포지토리를 선택하려면 리포지토리 이름을 기반으로 필터를 입력한 다음 필터 추가를 선택합니다.
6. 저장을 선택합니다.
  7. 고급 스캔을 켜려는 각 리전에 대해 이 단계를 반복합니다.

## AWS CLI

다음 AWS CLI 명령을 사용하여 개인 레지스트리에 대한 향상된 검사 기능을 활성화하십시오. AWS CLIrules 객체를 사용하여 스캔 필터를 지정할 수 있습니다.

- [put-registry-scanning-configuration](#)(AWS CLI)

다음 예제에서는 프라이빗 레지스트리에 대한 고급 스캔을 켭니다. 기본적으로 rules가 지정되지 않은 경우 Amazon ECR은 스캔 구성을 모든 리포지토리에 대해 연속 스캔으로 설정합니다.

```
aws ecr put-registry-scanning-configuration \
  --scan-type ENHANCED \
  --region us-east-2
```



다음 예제에서는 프라이빗 레지스트리에 대해 고급 스캔을 켜고 스캔 필터를 지정합니다. 예제의 스캔 필터를 사용하면 이름에 prod가 있는 모든 리포지토리에 대해 연속 스캔을 켤 수 있습니다.

```
aws ecr put-registry-scanning-configuration \
  --scan-type ENHANCED \
  --rules '[{"repositoryFilters" : [{"filter": "prod", "filterType" :
  "WILDCARD"}], "scanFrequency" : "CONTINUOUS_SCAN"}]' \
  --region us-east-2
```

다음 예제에서는 프라이빗 레지스트리에 대해 고급 스캔을 켜고 다중 스캔 필터를 지정합니다. 예제의 스캔 필터는 이름에 prod가 포함된 모든 리포지토리에 대해 연속 스캔을 켜고 다른 모든 리포지토리에 대해서만 푸시할 때 스캔을 켵니다.

```
aws ecr put-registry-scanning-configuration \
  --scan-type ENHANCED \
  --rules '[{"repositoryFilters" : [{"filter": "prod", "filterType" :
  "WILDCARD"}], "scanFrequency" : "CONTINUOUS_SCAN"}, {"repositoryFilters" :
  [{"filter": "*", "filterType" : "WILDCARD"}], "scanFrequency" : "SCAN_ON_PUSH"}]' \
  --region us-west-2
```

## Amazon Inspector에서 이미지의 향상된 스캔 지속 시간 변경

Amazon Inspector가 Amazon ECR 프라이빗 리포지토리의 이미지를 지속적으로 스캔하는 기간 (일)을 변경할 수 있습니다. 기본적으로 Amazon ECR 프라이빗 레지스트리에 대해 고급 스캔을 켜면 Amazon Inspector 서비스는 이미지가 삭제되거나 고급 스캔이 비활성화될 때까지 리포지토리를 계속 모니터링합니다. Amazon Inspector가 이미지를 스캔하는 기간은 Amazon Inspector 설정을 사용하여 변경할 수 있습니다. 사용 가능한 스캔 기간은 전체 기간(기본값), 180일 및 30일입니다. 리포지토리에 대한 스캔 기간이 경과하면 스캔 취약성을 나열할 때 SCAN\_ELIGIBILITY\_EXPIRED라는 스캔 상태가 표시됩니다. 자세한 내용은 Amazon Inspector 사용 설명서의 [Amazon ECR 자동 재스캔 기간 변경](#)을 참조하세요.

고급 스캔 기간 설정을 변경하려면

1. Amazon Inspector 콘솔(<https://console.aws.amazon.com/inspector/v2/home>)을 엽니다.
2. 왼쪽 탐색 메뉴에서 설정(Settings)을 확장한 다음 일반(General)을 선택합니다.
3. 설정(Settings) 페이지의 ECR 재스캔 기간(ECR re-scan duration)에서 설정을 선택한 다음 저장(Save)을 선택합니다.

## EventBridge Amazon ECR에서 향상된 스캔을 위해 전송된 이벤트

향상된 스캔이 켜져 있는 경우 Amazon ECR은 리포지토리의 스캔 빈도가 변경될 EventBridge 때 이벤트를 전송합니다. Amazon Inspector는 초기 스캔이 완료되고 이미지 스캔 결과가 생성, 업데이트 또는 닫힐 때 이벤트를 전송합니다. EventBridge

리포지토리 스캔 빈도 변경에 대한 이벤트

레지스트리에 대해 고급 스캔을 켜 경우 고급 스캔이 켜진 리소스가 변경되면 Amazon ECR에서 다음 이벤트를 전송합니다. 여기에는 생성 중인 새 리포지토리, 변경 중인 리포지토리의 스캔 빈도 또는 고급 스캔 기능이 켜진 리포지토리에서 이미지를 생성하거나 삭제하는 시점이 포함됩니다. 자세한 정보는 [이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다.](#)을 참조하세요.

```
{
  "version": "0",
  "id": "0c18352a-a4d4-6853-ef53-0abEXAMPLE",
  "detail-type": "ECR Scan Resource Change",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2021-10-14T20:53:46Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "action-type": "SCAN_FREQUENCY_CHANGE",
    "repositories": [{
      "repository-name": "repository-1",
      "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
      "scan-frequency": "SCAN_ON_PUSH",
      "previous-scan-frequency": "MANUAL"
    },
    {
      "repository-name": "repository-2",
      "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
      "scan-frequency": "CONTINUOUS_SCAN",
      "previous-scan-frequency": "SCAN_ON_PUSH"
    },
    {
      "repository-name": "repository-3",
      "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
      "scan-frequency": "CONTINUOUS_SCAN",
      "previous-scan-frequency": "SCAN_ON_PUSH"
    }
  ]
}
```

```

"resource-type": "REPOSITORY",
"scan-type": "ENHANCED"
}
}

```

### 초기 이미지 스캔 이벤트(고급 스캔)

레지스트리에 대해 고급 스캔을 켜 준 경우 초기 이미지 스캔이 완료되면 Amazon Inspector에서 다음 이벤트를 전송합니다. `finding-severity-counts` 파라미터는 심각도 수준이 있는 경우에만 값을 반환합니다. 예를 들어, 이미지에 CRITICAL 수준의 결과가 없으면 심각 카운트가 반환되지 않습니다. 자세한 정보는 [Amazon ECR의 OS 및 프로그래밍 언어 패키지 취약성이 있는지 이미지를 스캔합니다.](#)을 참조하세요.

### 이벤트 패턴:

```

{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Scan"]
}

```

### 출력 예제:

```

{
  "version": "0",
  "id": "739c0d3c-4f02-85c7-5a88-94a9EXAMPLE",
  "detail-type": "Inspector2 Scan",
  "source": "aws.inspector2",
  "account": "123456789012",
  "time": "2021-12-03T18:03:16Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample"
  ],
  "detail": {
    "scan-status": "INITIAL_SCAN_COMPLETE",
    "repository-name": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample",
    "finding-severity-counts": {
      "CRITICAL": 7,
      "HIGH": 61,
      "MEDIUM": 62,
      "TOTAL": 158
    }
  }
}

```

```

    },
    "image-digest":
"sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
    "image-tags": [
        "latest"
    ]
}
}

```

### 이미지 스캔 결과 업데이트 이벤트(고급 스캔)

레지스트리에 대해 고급 스캔을 켜면 이미지 스캔 결과가 생성, 업데이트 또는 종료될 때 Amazon Inspector에서 다음 이벤트를 전송합니다. 자세한 정보는 [Amazon ECR의 OS 및 프로그래밍 언어 패키지 취약성이 있는지 이미지를 스캔합니다.](#)을 참조하세요.

### 이벤트 패턴:

```

{
  "source": ["aws.inspector2"],
  "detail-type": ["Inspector2 Finding"]
}

```

### 출력 예제:

```

{
  "version": "0",
  "id": "42dbea55-45ad-b2b4-87a8-afaEXAMPLE",
  "detail-type": "Inspector2 Finding",
  "source": "aws.inspector2",
  "account": "123456789012",
  "time": "2021-12-03T18:02:30Z",
  "region": "us-east-2",
  "resources": [
    "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-sample/sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77eEXAMPLE"
  ],
  "detail": {
    "awsAccountId": "123456789012",
    "description": "In libssh2 v1.9.0 and earlier versions, the SSH_MSG_DISCONNECT logic in packet.c has an integer overflow in a bounds check, enabling an attacker to specify an arbitrary (out-of-bounds) offset for a subsequent memory read. A crafted SSH server may be able to disclose sensitive information or cause a denial of service condition on the client system when a user connects to the server.",
  }
}

```

```
"findingArn": "arn:aws:inspector2:us-east-2:123456789012:finding/
be674aadd0f75ac632055EXAMPLE",
"firstObservedAt": "Dec 3, 2021, 6:02:30 PM",
"inspectorScore": 6.5,
"inspectorScoreDetails": {
  "adjustedCvss": {
    "adjustments": [],
    "cvssSource": "REDHAT_CVE",
    "score": 6.5,
    "scoreSource": "REDHAT_CVE",
    "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
    "version": "3.0"
  }
},
"lastObservedAt": "Dec 3, 2021, 6:02:30 PM",
"packageVulnerabilityDetails": {
  "cvss": [
    {
      "baseScore": 6.5,
      "scoringVector": "CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N",
      "source": "REDHAT_CVE",
      "version": "3.0"
    },
    {
      "baseScore": 5.8,
      "scoringVector": "AV:N/AC:M/Au:N/C:P/I:N/A:P",
      "source": "NVD",
      "version": "2.0"
    },
    {
      "baseScore": 8.1,
      "scoringVector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:H",
      "source": "NVD",
      "version": "3.1"
    }
  ],
  "referenceUrls": [
    "https://access.redhat.com/errata/RHSA-2020:3915"
  ],
  "source": "REDHAT_CVE",
  "sourceUrl": "https://access.redhat.com/security/cve/CVE-2019-17498",
  "vendorCreatedAt": "Oct 16, 2019, 12:00:00 AM",
  "vendorSeverity": "Moderate",
  "vulnerabilityId": "CVE-2019-17498",
```

```

    "vulnerablePackages": [
      {
        "arch": "X86_64",
        "epoch": 0,
        "name": "libssh2",
        "packageManager": "OS",
        "release": "12.amzn2.2",
        "sourceLayerHash":
"sha256:72d97abdfae3b3c933ff41e39779cc72853d7bd9dc1e4800c5294dEXAMPLE",
        "version": "1.4.3"
      }
    ],
    "remediation": {
      "recommendation": {
        "text": "Update all packages in the vulnerable packages section to
their latest versions."
      }
    },
    "resources": [
      {
        "details": {
          "awsEcrContainerImage": {
            "architecture": "amd64",
            "imageHash":
"sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77e5EXAMPLE",
            "imageTags": [
              "latest"
            ],
            "platform": "AMAZON_LINUX_2",
            "pushedAt": "Dec 3, 2021, 6:02:13 PM",
            "registry": "123456789012",
            "repositoryName": "amazon/amazon-ecs-sample"
          }
        },
        "id": "arn:aws:ecr:us-east-2:123456789012:repository/amazon/amazon-ecs-
sample/sha256:36c7b282abd0186e01419f2e58743e1bf635808231049bbc9d77EXAMPLE",
        "partition": "N/A",
        "region": "N/A",
        "type": "AWS_ECR_CONTAINER_IMAGE"
      }
    ],
    "severity": "MEDIUM",
    "status": "ACTIVE",

```

```

    "title": "CVE-2019-17498 - libssh2",
    "type": "PACKAGE_VULNERABILITY",
    "updatedAt": "Dec 3, 2021, 6:02:30 PM"
  }
}

```

## Amazon ECR에서 향상된 스캔을 위한 결과 검색

마지막으로 완료된 향상된 이미지 스캔에 대한 스캔 결과를 검색한 다음 Amazon Inspector에서 결과를 열어 자세한 내용을 볼 수 있습니다. 발견된 소프트웨어 취약성은 CVE (일반 취약성 및 노출) 데이터베이스를 기반으로 심각도별로 나열됩니다.

이미지 스캔 시 몇 가지 일반적인 문제에 대한 문제 해결 세부 정보를 보려면 [Amazon ECR에서의 이미지 스캔 문제 해결](#) 단원을 참조하세요.

### AWS Management Console

AWS Management Console을 사용하여 이미지 스캔 결과를 검색하려면 다음 단계를 따르세요.

이미지 스캔 결과를 검색하려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 리포지토리가 있는 리전을 선택합니다.
3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 스캔 결과를 검색할 이미지가 포함된 리포지토리를 선택합니다.
5. 이미지(Images) 페이지의 취약성(Vulnerabilities) 열에서 스캔 결과를 검색할 이미지의 결과 보기(See findings)를 선택합니다.
6. Amazon Inspector 콘솔에서 자세한 내용을 보려면 이름 열에서 취약성 이름을 선택하십시오.

### AWS CLI

다음 AWS CLI 명령을 사용하여 이미지 스캔 결과를 검색하십시오. AWS CLI imageTag 또는 imageDigest를 사용하여 이미지를 지정할 수 있으며, 둘 다 [list-images](#) CLI 명령을 사용하여 가져올 수 있습니다.

- [describe-image-scan-findings](#)(AWS CLI)

다음 예에서는 이미지 태그를 사용합니다.

```
aws ecr describe-image-scan-findings \  
  --repository-name name \  
  --image-id imageTag=tag_name \  
  --region us-east-2
```

다음 예에서는 이미지 다이제스트를 사용합니다.

```
aws ecr describe-image-scan-findings \  
  --repository-name name \  
  --image-id imageDigest=sha256_hash \  
  --region us-east-2
```

## Amazon ECR의 OS 취약성에 대한 이미지 스캔

향상된 기본 스캔 기능은 Amazon ECR의 프리뷰 릴리즈 단계에 있으며 변경될 수 있습니다. 이 공개 미리 보기 동안에는 향상된 기본 스캔 버전을 AWS Management Console 옵트인하는 데만 사용할 수 있습니다.

Amazon ECR은 일반적인 취약성 및 노출 (CVE) 데이터베이스를 사용하는 두 가지 버전의 기본 검사를 제공합니다.

- 오픈 소스 Clair 프로젝트를 사용하는 최신 GA 버전입니다. [Clair에 대한 자세한 내용은 Clair on을 참조하십시오.](#) GitHub
- 기본 기술을 사용하는 AWS 새롭게 개선된 기본 스캔 버전 (미리 보기).

Amazon ECR은 가능한 경우 업스트림 배포 소스의 CVE 심각도를 사용합니다. 그렇지 않으면 일반 취약점 평가 시스템 (CVSS) 점수가 사용됩니다. CVSS 점수를 사용하여 NVD 취약성 심각도 등급을 얻을 수 있습니다. 자세한 내용은 [NVD 취약성 심각도 등급](#)을 참조하세요.

Amazon ECR 기본 검사 두 버전 모두 푸시 시 스캔할 저장소를 지정하는 필터를 지원합니다. 푸시 필터 스캔과 일치하지 않는 모든 리포지토리는 수동 스캔 빈도로 설정되므로 수동으로 스캔을 시작해야 합니다. 이미지는 24시간에 한 번 스캔할 수 있습니다. 24시간에는 푸시 시 초기 스캔 (구성된 경우) 과 모든 수동 스캔이 포함됩니다.



각 이미지에 대해 마지막으로 완료된 이미지 스캔 결과를 검색할 수 있습니다. 이미지 스캔이 완료되면 Amazon ECR은 Amazon에 이벤트를 보냅니다. EventBridge 자세한 정보는 [아마존 ECR 이벤트 및 EventBridge](#)을 참조하세요.

## 향상된 기본 스캔을 위한 지역 지원

기본 스캔의 향상된 버전은 다음 지역에서 지원됩니다.

- 아시아 태평양(홍콩)(ap-east-1)
- EU(스톡홀름)(eu-north-1)
- 중동(바레인)(me-south-1)
- 아시아 태평양(뭄바이) (ap-south-1)
- EU(파리)(eu-west-3)
- AWS GovCloud (미국 동부) (us-gov-east-1)
- 아프리카(케이프타운)(af-south-1)
- 아시아 태평양(자카르타)(ap-southeast-3)
- EU (프랑크푸르트)(eu-central-1)
- EU (아일랜드)(eu-west-1)
- 남아메리카(상파울루)(sa-east-1)
- 미국 동부 (오하이오)(us-east-2)
- AWS GovCloud (미국 서부) () us-gov-west-1
- 아시아 태평양(도쿄) (ap-northeast-1)
- 아시아 태평양(서울) (ap-northeast-2)
- 아시아 태평양(오사카)(ap-northeast-3)
- EU(밀라노)(eu-south-1)
- EU(런던)(eu-west-2)
- 미국 동부 (버지니아 북부) (us-east-1)
- 아시아 태평양(싱가포르) (ap-southeast-1)
- 아시아 태평양(시드니) (ap-southeast-2)
- 캐나다(중부) (ca-central-1)
- 미국 서부(캘리포니아 북부)(us-west-1)

- 미국 서부 (오리건) (us-west-2)
- 유럽(취리히)(eu-central-2)

## 기본 스캔 및 향상된 기본 스캔을 위한 운영 체제 지원

보안 모범 사례이자 지속적인 적용 범위를 유지하기 위해 지원되는 운영 체제 버전을 계속 사용하는 것이 좋습니다. 공급업체 정책에 따라 단종된 운영 체제는 더 이상 패치로 업데이트되지 않으며, 대부분의 경우 해당 운영 체제에 대한 새로운 보안 권고가 더 이상 발표되지 않습니다. 또한 일부 공급업체의 경우 영향을 받는 운영 체제의 표준 지원이 종료되면 피드에서 기존 보안 권고 및 탐지를 제거합니다. 배포에 공급업체의 지원이 중단되면 Amazon ECR은 더 이상 취약성 검사를 지원하지 않을 수 있습니다. Amazon ECR이 단종된 운영 체제에 대해 생성하는 모든 결과는 정보 제공 목적으로만 사용해야 합니다. 현재 지원되는 운영 체제 및 버전은 다음과 같습니다.

운영 체제	버전
Alpine Linux(Alpine)	3.19
Alpine Linux(Alpine)	3.18
Alpine Linux(Alpine)	3.17
Alpine Linux(Alpine)	3.16
Amazon Linux 2(AL2)	AL2
아마존 리눅스 2023 (AL2023)	AL2023
CentOS Linux(CentOS)	7
Debian Server(Bookworm)	12
Debian Server(Bullseye)	11
Debian Server(Buster)	10
Oracle Linux(Oracle)	9
Oracle Linux(Oracle)	8
Oracle Linux(Oracle)	7

운영 체제	버전
우분투 (음력)	23.04
Ubuntu(Jammy)	22.04(LTS)
Ubuntu(Focal)	20.04(LTS)
Ubuntu(Bionic)	18.04(ESM)
Ubuntu(Xenial)	16.04(ESM)
Ubuntu(Trusty)	14.04(ESM)
Red Hat Enterprise Linux(RHEL)	7
Red Hat Enterprise Linux(RHEL)	8
Red Hat Enterprise Linux(RHEL)	9

## Amazon ECR에서 이미지에 대한 향상된 기본 스캔 구성

이제 Amazon ECR 기본 스캔의 향상된 버전을 미리 보기로 사용할 수 있습니다. 향상된 기본 스캔은 AWS 기본 기술을 사용합니다.

개인 리포지토리의 지역별 기본 검사를 개선하도록 구성하십시오. 향상된 기본 검사를 지원하는 지역 목록은 [을 참조하십시오 향상된 기본 스캔을 위한 지역 지원](#).

개인 레지스트리에 대한 향상된 기본 검사를 켜려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 스캔 구성을 설정할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 스캔을 선택합니다.
4. 스캔 구성 페이지에서 스캔 유형으로 향상된 기본 스캔 (미리 보기 중) - 새로 만들기를 선택합니다.
5. 기본적으로 모든 리포지토리는 수동 스캔으로 설정됩니다. 필요에 따라 필터 푸시 시 스캔을 지정하여 푸시할 때 스캔을 구성할 수 있습니다. 모든 리포지토리 또는 개별 리포지토리에 대해 푸시할 때 스캔을 설정할 수 있습니다. 자세한 정보는 [Amazon ECR에서 스캔할 리포지토리를 선택하는 필터](#)를 참조하세요.

## Amazon ECR에서 이미지에 대한 기본 스캔 구성

Amazon ECR은 기본적으로 모든 프라이빗 레지스트리에 대해 기본 스캔 기능을 활성화합니다. 따라서 프라이빗 레지스트리의 스캔 설정을 변경하지 않는 한 기본 스캔을 켜 필요 없습니다. 기본 스캔은 오픈 소스 Clair 프로젝트를 사용합니다.

다음 단계를 사용하여 푸시 필터에서 하나 이상의 스캔을 정의할 수 있습니다.

개인 레지스트리에 대한 기본 검사를 켜려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 스캔 구성을 설정할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 스캔을 선택합니다.
4. 스캔 구성 페이지의 스캔 유형에서 기본 스캔을 선택합니다.
5. 기본적으로 모든 리포지토리는 수동 스캔으로 설정됩니다. 필요에 따라 필터 푸시 시 스캔을 지정하여 푸시할 때 스캔을 구성할 수 있습니다. 모든 리포지토리 또는 개별 리포지토리에 대해 푸시할 때 스캔을 설정할 수 있습니다. 자세한 정보는 [Amazon ECR에서 스캔할 리포지토리를 선택하는 필터](#)를 참조하세요.

## Amazon ECR의 OS 취약성을 찾기 위해 이미지를 수동으로 스캔합니다.

리포지토리가 푸시 시 스캔하도록 구성되지 않은 경우 수동으로 이미지 스캔을 시작할 수 있습니다. 이미지는 24시간에 한 번 스캔할 수 있습니다. 24시간에는 푸시 시 초기 스캔 (구성된 경우) 과 모든 수동 스캔이 포함됩니다.

이미지 스캔 시 몇 가지 일반적인 문제에 대한 문제 해결 세부 정보를 보려면 [Amazon ECR에서의 이미지 스캔 문제 해결](#) 단원을 참조하세요.

### AWS Management Console

AWS Management Console을 사용하여 수동 이미지 스캔을 시작하려면 다음 단계를 따르세요.

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 리포지토리를 생성할 리전을 선택합니다.
3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 스캔할 이미지가 들어 있는 리포지토리를 선택합니다.
5. 이미지(Images) 페이지에서 스캔할 이미지를 선택한 다음 스캔(Scan)을 선택합니다.

## AWS CLI

- [start-image-scan](#)(AWS CLI)

다음 예에서는 이미지 태그를 사용합니다.

```
aws ecr start-image-scan --repository-name name --image-id imageTag=tag_name --region us-east-2
```

다음 예에서는 이미지 다이제스트를 사용합니다.

```
aws ecr start-image-scan --repository-name name --image-id imageDigest=sha256_hash --region us-east-2
```

## AWS Tools for Windows PowerShell

- [Get-ECR 검색 결과 \(\) ImageScan](#)AWS Tools for Windows PowerShell

다음 예에서는 이미지 태그를 사용합니다.

```
Start-ECRImageScan -RepositoryName name -ImageId_ImageTag tag_name -Region us-east-2 -Force
```

다음 예에서는 이미지 다이제스트를 사용합니다.

```
Start-ECRImageScan -RepositoryName name -ImageId_ImageDigest sha256_hash -Region us-east-2 -Force
```

## Amazon ECR에서 기본 스캔을 위한 결과 검색

마지막으로 완료된 기본 이미지 스캔의 스캔 결과를 검색할 수 있습니다. 발견된 소프트웨어 취약성은 일반적인 취약성 및 노출 (CVE) 데이터베이스를 기반으로 심각도별로 나열됩니다.

이미지 스캔 시 몇 가지 일반적인 문제에 대한 문제 해결 세부 정보를 보려면 [Amazon ECR에서의 이미지 스캔 문제 해결](#) 단원을 참조하세요.

## AWS Management Console

AWS Management Console을 사용하여 이미지 스캔 결과를 검색하려면 다음 단계를 따르세요.

이미지 스캔 결과를 검색하려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 리포지토리를 생성할 리전을 선택합니다.
3. 탐색 창에서 리포지토리를 선택합니다.
4. 리포지토리(Repositories) 페이지에서 스캔 결과를 검색할 이미지가 포함된 리포지토리를 선택합니다.
5. 이미지(Images) 페이지의 취약성(Vulnerabilities) 열에서 스캔 결과를 검색할 이미지의 세부 정보(Details)를 선택합니다.

## AWS CLI

다음 AWS CLI 명령을 사용하여 이미지 스캔 결과를 검색하십시오 AWS CLI. `imageTag` 또는 `imageDigest`를 사용하여 이미지를 지정할 수 있으며, 둘 다 [list-images](#) CLI 명령을 사용하여 가져올 수 있습니다.

- [describe-image-scan-findings](#)(AWS CLI)

다음 예에서는 이미지 태그를 사용합니다.

```
aws ecr describe-image-scan-findings --repository-name name --image-id
imageTag=tag_name --region us-east-2
```

다음 예에서는 이미지 다이제스트를 사용합니다.

```
aws ecr describe-image-scan-findings --repository-name name --image-id
imageDigest=sha256_hash --region us-east-2
```

## AWS Tools for Windows PowerShell

- [Get-ECR 검색 결과 \(\) ImageScan](#) AWS Tools for Windows PowerShell

다음 예에서는 이미지 태그를 사용합니다.

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageTag tag_name -
Region us-east-2
```

다음 예에서는 이미지 다이제스트를 사용합니다.

```
Get-ECRImageScanFinding -RepositoryName name -ImageId_ImageDigest sha256_hash -
Region us-east-2
```

## Amazon ECR에서의 이미지 스캔 문제 해결

다음은 일반적인 이미지 스캔 오류입니다. Amazon ECR 콘솔에서 이미지 세부 정보를 표시하거나 API 또는 AWS CLI API를 사용하여 이와 같은 오류를 볼 수 있습니다. DescribeImageScanFindings

### UnsupportedImage오류

Amazon ECR이 기본 이미지 스캔을 지원하지 않는 운영 체제를 사용하여 구축된 이미지에 대해 기본 스캔을 수행하려고 하면 UnsupportedImageError 오류가 발생할 수 있습니다. Amazon ECR은 Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine, and RHEL Linux 배포판의 주 버전에 대한 패키지 취약점 검색을 지원합니다. 배포판이 판매업체의 지원을 받지 않게 되면 Amazon ECR에서 취약성 검색을 더 이상 지원하지 않을 수 있습니다. Amazon ECR은 [Docker scratch](#) 이미지에서 구축된 이미지의 스캔을 지원하지 않습니다.

#### Important

고급 스캔을 사용할 경우 Amazon Inspector는 특정 운영 체제 및 미디어 유형에 대한 스캔을 지원합니다. 전체 목록은 Amazon Inspector 사용 설명서의 [지원되는 운영 체제 및 미디어 유형](#)을 참조하세요.

UNDEFINED 심각도 수준이 반환됩니다.

심각도 수준이 UNDEFINED인 스캔 결과를 받을 수 있습니다. 이에 대한 일반적인 원인은 다음과 같습니다.

- 이 취약성에는 CVE 소스에 의해 우선 순위가 할당되지 않았습니다.
- 이 취약성에는 Amazon ECR이 인식하지 못한 우선 순위가 할당되었습니다.

취약성의 심각도 및 설명을 확인하려면 소스에서 직접 CVE를 확인하면 됩니다.

## 스캔 상태 **SCAN\_ELIGIBILITY\_EXPIRED** 이해

프라이빗 레지스트리에 대해 Amazon Inspector를 사용한 고급 스캔이 활성화되어 있고 스캔 취약성을 보고 있는 경우 스캔 상태가 **SCAN\_ELIGIBILITY\_EXPIRED**로 표시될 수 있습니다. 이에 대한 가장 일반적인 원인은 다음과 같습니다.

- 프라이빗 레지스트리에 대해 고급 스캔을 처음 켜면 Amazon Inspector는 이미지 푸시 타임스탬프를 근거로 지난 30일 동안 Amazon ECR에 푸시된 이미지만 인식합니다. 이전 이미지에 **SCAN\_ELIGIBILITY\_EXPIRED** 스캔 상태가 포함됩니다. Amazon Inspector에서 이러한 이미지를 스캔하려는 경우에는 이미지를 해당 리포지토리로 다시 푸시해야 합니다.
- Amazon Inspector 콘솔에서 ECR 재스캔 기간(ECR re-scan duration)이 변경되고 이 시간이 경과하면 이미지의 스캔 상태가 사유 코드 **inactive**와 함께 **expired**로 변경되고 이미지에 대한 모든 관련 검색 결과가 닫히도록 예약됩니다. 그 결과 Amazon ECR 콘솔에 스캔 상태가 **SCAN\_ELIGIBILITY\_EXPIRED**로 표시됩니다.



# 업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화

풀 스루 캐시 규칙을 사용하여 업스트림 레지스트리의 콘텐츠를 Amazon ECR 사설 레지스트리와 동기화할 수 있습니다.

Amazon ECR은 현재 다음 업스트림 레지스트리에 대한 풀스루 캐시 규칙 생성을 지원합니다.

- Docker Hub, Microsoft Azure 컨테이너 레지스트리, GitHub 컨테이너 레지스트리, GitLab 컨테이너 레지스트리 (인증 필요)
- Amazon ECR 퍼블릭, Kubernetes 컨테이너 이미지 레지스트리, Quay(인증 필요 없음)

GitLab 컨테이너 레지스트리의 경우 Amazon ECR은 GitLab .com GitLab software-as-a-service 오퍼링을 통해서만 풀스루 캐시를 지원합니다.

인증이 필요한 업스트림 레지스트리의 경우 자격 증명을 비밀로 저장해야 합니다. AWS Secrets Manager Amazon ECR 콘솔을 사용하면 인증된 업스트림 레지스트리 각각에 대해 Secrets Manager 보안 암호를 쉽게 생성할 수 있습니다. Secrets Manager 콘솔을 사용하여 Secrets Manager 시크릿을 생성하는 방법에 대한 자세한 내용은 [업스트림 리포지토리 자격 증명을 비밀로 AWS Secrets Manager 저장](#).

업스트림 레지스트리에 대한 풀스루 캐시 규칙을 생성한 후에는 Amazon ECR 프라이빗 레지스트리 URI를 사용하여 해당 업스트림 레지스트리에서 이미지를 가져옵니다. 그러면 Amazon ECR이 리포지토리를 생성하고 해당 이미지를 프라이빗 레지스트리에 캐시합니다. 지정된 태그가 있는 캐시된 이미지에 대한 후속 pull 요청에서 Amazon ECR은 업스트림 레지스트리를 확인하여 특정 태그가 포함된 새 버전의 이미지가 있는지 확인하고 최소 24시간에 한 번씩 프라이빗 레지스트리의 이미지 업데이트를 시도합니다.

## 리포지토리 생성 템플릿

Amazon ECR은 현재 평가판으로 제공되며 리포지토리 생성 템플릿에 대한 지원을 추가했습니다. 이 템플릿에서는 사용자 대신 풀스루 캐시 규칙을 사용하여 Amazon ECR에서 생성한 새 리포지토리의 초기 구성을 지정할 수 있습니다. 각 템플릿에는 새 리포지토리를 특정 템플릿에 매칭하는 데 사용되는 리포지토리 네임스페이스 접두사가 포함되어 있습니다. 템플릿은 리소스 기반 액세스 정책, 태그 불변성, 암호화, 수명 주기 정책을 비롯한 모든 리포지토리 설정의 구성을 지정할 수 있습니다. 리포지토리 생성 템플릿의 설정은 리포지토리를 생성하는 동안에만 적용되며 기존 리포지토리나 다른 방법을 사

용하여 생성한 리포지토리에는 영향을 주지 않습니다. 자세한 정보는 [폴 스루 캐시 작업 중에 생성된 리포지토리를 제어하기 위한 템플릿](#)을 참조하세요.

## 폴스루 캐시 규칙 사용 시 고려 사항

Amazon ECR 폴스루 캐시 규칙을 사용할 때는 다음 사항을 고려하십시오.

- 다음 리전에서는 폴스루 캐시 규칙 생성이 지원되지 않습니다.
  - 중국(베이징)(cn-north-1)
  - 중국(닝샤)(cn-northwest-1)
  - AWS GovCloud (미국 동부) () us-gov-east-1
  - AWS GovCloud (미국 서부) () us-gov-west-1
- AWS Lambda 폴스루 캐시 규칙을 사용하여 Amazon ECR에서 컨테이너 이미지를 가져오는 것은 지원하지 않습니다.
- 폴스루 캐시를 사용하여 이미지를 가져올 경우 이미지를 처음 가져올 때 Amazon ECR FIPS 서비스 엔드포인트가 지원되지 않습니다. Amazon ECR FIPS 서비스 엔드포인트를 사용하면 후속 폴스루에서 작동합니다.
- Amazon ECR 사설 레지스트리 URI를 통해 캐시된 이미지를 가져오면 IP 주소를 기준으로 이미지 가져오기가 시작됩니다. AWS 이렇게 하면 이미지 풀이 업스트림 레지스트리에서 구현한 풀 레이트 할당량에 포함되지 않습니다.
- Amazon ECR 프라이빗 레지스트리 URI를 통해 캐시된 이미지를 가져오면 Amazon ECR은 업스트림 리포지토리를 24시간마다 한 번 이상 점검하여 캐시된 이미지가 최신 버전인지 확인합니다. 업스트림 레지스트리에 새 이미지가 있는 경우 Amazon ECR은 캐시된 이미지를 업데이트하려고 시도합니다. 이 타이머는 캐시된 이미지의 마지막 풀을 기반으로 합니다.
- 어떤 이유로든 Amazon ECR이 업스트림 레지스트리에서 이미지를 업데이트할 수 없고 이미지를 가져오더라도 마지막으로 캐시된 이미지는 여전히 가져옵니다.
- 업스트림 레지스트리 보안 인증 정보가 포함된 Secrets Manager 보안 암호를 만들 때는 보안 암호 이름에 `ecr-pullthroughcache/` 접두사를 사용해야 합니다. 또한 보안 암호는 폴스루 캐시 규칙이 생성된 동일한 계정 및 리전에 있어야 합니다.
- 폴스루 캐시 규칙을 사용하여 다중 아키텍처 이미지를 가져오면, Amazon ECR 리포지토리로 매니페스트 목록과 매니페스트 목록에서 참조되는 각 이미지를 가져옵니다. 특정 아키텍처만 가져오려는 경우 매니페스트 목록과 연결된 태그 대신 아키텍처와 연결된 이미지 다이제스트 또는 태그를 사용하여 이미지를 가져올 수 있습니다.
- Amazon ECR은 서비스 연결 IAM 역할을 사용합니다. 이 역할은 Amazon ECR이 리포지토리를 생성하고 Secrets Manager 보안 암호 값을 검색하여 인증하며 사용자를 대신하여 캐시된 이미지를 푸시

하는 데 필요한 권한을 제공합니다. 서비스 연결 IAM 역할은 풀스루 캐시 규칙이 만들어질 때 자동으로 생성됩니다. 자세한 정보는 [풀스루 캐시에 대한 Amazon ECR 서비스 연결 역할](#)을 참조하세요.

- 기본적으로 캐시된 이미지를 가져오는 IAM 보안 주체에는 IAM 정책을 통해 부여된 권한이 있습니다. Amazon ECR 프라이빗 레지스트리 권한 정책을 사용하여 IAM 엔터티의 권한 범위를 추가로 지정할 수 있습니다. 자세한 정보는 [레지스트리 권한 사용](#)을 참조하세요.
- 풀스루 캐시 워크플로를 사용하여 생성된 Amazon ECR 리포지토리는 다른 모든 Amazon ECR 리포지토리처럼 취급됩니다. 복제 및 이미지 스캔과 같은 모든 리포지토리 기능이 지원됩니다.
- Amazon ECR이 풀스루 캐시 작업을 사용하여 사용자 대신 새 리포지토리를 생성하는 경우 일치하는 리포지토리 생성 템플릿이 없는 한 다음 기본 설정이 리포지토리에 적용됩니다. 리포지토리 생성 템플릿을 사용하여 Amazon ECR에서 생성한 리포지토리에 적용되는 설정을 사용자 대신 정의할 수 있습니다. 자세한 정보는 [풀스루 캐시 작업 중에 생성된 리포지토리를 제어하기 위한 템플릿](#)을 참조하세요.
- 태그 불변성 - 이 기능을 비활성화하면 태그는 변경 가능하며 덮어쓸 수 있습니다.
- 암호화 - 기본 AES256 암호화가 사용됩니다.
- 리포지토리 권한 - 생략됨, 리포지토리 권한 정책이 적용되지 않습니다.
- 수명 주기 정책 - 생략됨, 수명 주기 정책이 적용되지 않습니다.
- 리소스 태그 - 생략됨, 리소스 태그가 적용되지 않습니다.
- 풀스루 캐시 규칙을 사용하여 리포지토리의 이미지 태그 불변성을 활성화하면 Amazon ECR에서 동일한 태그를 사용하는 이미지를 업데이트할 수 없게 됩니다.
- 풀스루 캐시 규칙을 사용하여 이미지를 처음으로 가져오는 경우 인터넷으로의 경로가 필요할 수 있습니다. 특정 상황에서는 인터넷 경로가 필요하므로 장애가 발생하지 않도록 경로를 설정하는 것이 가장 좋습니다. 따라서 AWS PrivateLink 사용하는 인터페이스 VPC 엔드포인트를 사용하도록 Amazon ECR을 구성한 경우 첫 번째 pull 시 인터넷 경로가 있는지 확인해야 합니다. 이를 위한 한 가지 방법은 인터넷 게이트웨이가 있는 동일한 VPC에 퍼블릭 서브넷을 만든 다음, 모든 아웃바운드 트래픽을 프라이빗 서브넷에서 퍼블릭 서브넷으로 인터넷으로 라우팅하는 것입니다. 풀스루 캐시 규칙을 사용한 후속 이미지 풀에는 이 설정이 필요하지 않습니다. 자세한 내용을 알아보려면 Amazon Virtual Private Cloud 사용 설명서의 [라우팅 옵션 예](#)을 참조하세요.

## 업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화하는 데 필요한 IAM 권한

풀스루 캐시 규칙을 효과적으로 사용하기 위해서는 프라이빗 레지스트리에 인증하고 이미지를 푸시하고 가져오는 데 필요한 Amazon ECR API 권한과 더불어 다음과 같은 추가 권한이 필요합니다.

- `ecr:CreatePullThroughCacheRule` – 풀스루 캐시 규칙을 생성할 수 있는 권한을 부여합니다. 이 권한은 자격 증명 기반 IAM 정책을 통해 부여되어야 합니다.
- `ecr:BatchImportUpstreamImage` – 외부 이미지를 검색하고 이 이미지를 프라이빗 레지스트리로 가져올 수 있는 권한을 부여합니다. 이 권한은 프라이빗 레지스트리 권한 정책, 자격 증명 기반 IAM 정책을 사용하거나 리소스 기반 리포지토리 권한 정책을 사용하여 부여될 수 있습니다. 리포지토리 권한에 대한 자세한 내용은 [Amazon ECR의 프라이빗 리포지토리 정책](#) 섹션을 참조하세요.
- `ecr:CreateRepository` – 프라이빗 레지스트리에 리포지토리를 생성할 수 있는 권한을 부여합니다. 이 권한은 캐시된 이미지를 저장하는 리포지토리가 이미 존재하지 않는 경우에 필요합니다. 이 권한은 자격 증명 기반 IAM 정책 또는 프라이빗 레지스트리 권한 정책에 의해 부여할 수 있습니다.
- `ecr:TagResource` - Amazon ECR 리소스에 메타데이터 태그를 추가할 수 있는 권한을 부여합니다. 이 권한은 리포지토리에 리소스 태그를 추가하도록 구성된 관련 리포지토리 생성 템플릿이 있는 풀스루 캐시 규칙을 사용하는 이미지를 가져오는 경우에만 필요합니다. 이 권한은 자격 증명 기반 IAM 정책을 통해 부여되어야 합니다.

## 레지스트리 권한 사용

Amazon ECR 프라이빗 레지스트리 권한은 풀스루 캐시를 사용하도록 개별 IAM 엔터티의 권한 범위를 지정하는 데 활용할 수 있습니다. IAM 엔터티에 레지스트리 권한 정책에서 허용하는 권한보다 IAM 정책에서 부여한 권한이 많을 경우 IAM 정책이 우선합니다. 예를 들어, 사용자에게 `ecr:*` 권한이 부여된 경우에는 레지스트리 수준에서 추가 권한이 필요하지 않습니다.

프라이빗 레지스트리에 대한 권한 정책을 생성하는 방법(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 프라이빗 레지스트리 권한 문을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 레지스트리 권한을 선택합니다.
4. 레지스트리 권한(Registry permissions) 페이지에서 문 생성(Generate statement)을 선택합니다.
5. 생성하려는 각 풀스루 캐시 권한 정책 문에 대해 다음을 수행합니다.
  - a. 정책 유형(Policy type)으로 풀스루 캐시 정책(Pull through cache policy)을 선택합니다.
  - b. 문 ID(Statement id)에서 풀스루 캐시 문 정책의 이름을 입력합니다.
  - c. IAM 엔터티(IAM entities)에서 정책에 포함할 사용자, 그룹 또는 역할을 지정합니다.
  - d. 리포지토리 네임스페이스(Repository namespace)로 정책을 연결할 풀스루 캐시 규칙을 선택합니다.

- e. 리포지토리 이름(Repository names)에서 규칙을 적용할 리포지토리 기본 이름을 지정합니다. 예를 들어, Amazon ECR Public에서 Amazon Linux 리포지토리를 지정하려는 경우 리포지토리 이름은 `amazonlinux`입니다.

프라이빗 레지스트리에 대한 권한 정책을 생성하는 방법(AWS CLI)

다음 AWS CLI 명령을 사용하여 프라이빗 레지스트리 권한을 지정합니다. AWS CLI

1. 레지스트리 정책의 내용과 `ptc-registry-policy.json`이라는 이름의 로컬 파일을 생성합니다. 다음 예제에서는 사용자에게 리포지토리를 생성하고 과거에 생성된 풀스루 캐시 규칙에 연결된 업스트림 소스인 Amazon ECR 퍼블릭에서 이미지를 가져올 수 있는 `ecr-pull-through-cache-user` 권한을 부여합니다.

```
{
  "Sid": "PullThroughCacheFromReadOnlyRole",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/ecr-pull-through-cache-user"
  },
  "Action": [
    "ecr:CreateRepository",
    "ecr:BatchImportUpstreamImage"
  ],
  "Resource": "arn:aws:ecr:us-east-1:111122223333:repository/ecr-public/*"
}
```

#### Important

`ecr:CreateRepository` 권한은 캐시된 이미지를 저장하는 리포지토리가 이미 존재하지 않는 경우에만 필요합니다. 예를 들어 리포지토리 생성 작업과 이미지 풀 작업이 관리자 및 개발자와 같은 별도의 IAM 보안 주체에 의해 수행되는 경우가 해당됩니다.

2. [put-registry-policy](#) 명령을 사용하여 레지스트리 정책을 설정합니다.

```
aws ecr put-registry-policy \
  --policy-text file://ptc-registry.policy.json
```

## 다음 단계

풀스루 캐시 규칙을 사용할 준비가 되었으면 다음 단계는 다음과 같습니다.

- 풀스루 캐시 규칙을 생성합니다. 자세한 정보는 [Amazon ECR에서 풀스루 캐시 규칙 생성](#)을 참조하세요.
- 리포지토리 생성 템플릿을 생성합니다. 리포지토리 생성 템플릿을 사용하면 풀스루 캐시 작업 중에 Amazon ECR에서 사용자 대신 생성한 새 리포지토리에 사용할 설정을 정의할 수 있습니다. 자세한 정보는 [풀스루 캐시 작업 중에 생성된 리포지토리를 제어하기 위한 템플릿](#)을 참조하세요.

## Amazon ECR에서 풀스루 캐시 규칙 생성

Amazon ECR 사설 레지스트리에 캐시하려는 이미지가 포함된 각 업스트림 레지스트리에 대해 풀스루 캐시 규칙을 생성해야 합니다.

인증이 필요한 업스트림 레지스트리의 경우 자격 증명을 Secrets Manager 암호에 저장해야 합니다. 기존의 암호를 사용하거나 새 암호를 생성할 수 있습니다. Amazon ECR 콘솔 또는 Secrets Manager 콘솔에서 Secrets Manager 시크릿을 생성할 수 있습니다. Amazon ECR 콘솔 대신 Secrets Manager 콘솔을 사용하여 Secrets Manager 시크릿을 생성하려면 [참조하십시오 업스트림 리포지토리 자격 증명을 비밀로 AWS Secrets Manager 저장](#).

## 필수 조건

- 풀스루 캐시 규칙을 생성할 수 있는 적절한 IAM 권한이 있는지 확인하십시오. 자세한 내용은 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화하는 데 필요한 IAM 권한](#)을 참조하세요.
- 인증이 필요한 업스트림 레지스트리의 경우: 기존 암호를 사용하려면 Secrets Manager 암호가 다음 요구 사항을 충족하는지 확인하십시오.
  - 시크릿 이름은 로 시작합니다. ecr-pullthroughcache/ ecr-pullthroughcache/접두사가 있는 Secrets Manager AWS Management Console 암호만 표시됩니다.
  - 암호가 있는 계정 및 지역은 풀스루 캐시 규칙이 속한 계정 및 지역과 일치해야 합니다.

## 풀스루 캐시 규칙을 생성하는 방법(AWS Management Console)

다음 단계는 Amazon ECR 콘솔을 사용하여 풀스루 캐시 규칙 및 Secrets Manager 보안 암호를 생성하는 방법을 보여줍니다. Secrets Manager 콘솔을 사용하여 시크릿을 만들려면 [을 참조하십시오](#) [업스트림 리포지토리 자격 증명을 비밀로 AWS Secrets Manager 저장](#).


Amazon ECR 퍼블릭, Kubernetes 컨테이너 레지스트리 또는 Quay의 경우

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 프라이빗 레지스트리 설정을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 풀스루 캐시를 선택합니다.
4. 풀스루 캐시 구성(Pull through cache configuration) 페이지에서 규칙 추가(Add rule)를 선택합니다.
5. 1단계: 소스 지정 페이지에서 레지스트리에 대해 업스트림 레지스트리 목록에서 Amazon ECR 퍼블릭, Kubernetes 또는 Quay를 선택한 후 다음을 선택합니다.
6. 2단계: 대상 지정 페이지에서 Amazon ECR 리포지토리 접두사에 대해 소스 퍼블릭 레지스트리에서 가져온 이미지를 캐싱할 때 사용할 리포지토리 네임스페이스 접두사를 지정하고 다음을 선택합니다. 네임스페이스는 기본적으로 작성되지만, 사용자 지정 네임스페이스를 지정할 수도 있습니다.
7. 3단계: 검토 및 생성 페이지에서 풀스루 캐시 규칙 구성을 검토한 다음 생성을 선택합니다.
8. 생성하려는 각 풀스루 캐시에 대해 이전 단계를 반복합니다. 풀스루 캐시 규칙은 각 리전에 대해 별도로 생성됩니다.

Docker Hub의 경우

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 프라이빗 레지스트리 설정을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 풀스루 캐시를 선택합니다.
4. 풀스루 캐시 구성(Pull through cache configuration) 페이지에서 규칙 추가(Add rule)를 선택합니다.
5. 1단계: 소스 지정 페이지에서 레지스트리에 대해 Docker Hub, 다음을 선택합니다.
6. 2단계: 인증 구성 페이지에서 업스트림 보안 인증 정보에 대해 Docker Hub의 인증 관련 보안 인증 정보를 AWS Secrets Manager 보안 암호로 저장해야 합니다. 기존 보안 암호를 지정하거나 Amazon ECR 콘솔을 사용하여 새 보안 암호를 생성할 수 있습니다.

- a. 기존 암호를 사용하려면 기존 암호 사용을 선택합니다. AWS 보안 암호 이름의 경우 드롭다운을 사용하여 기존 보안 암호를 선택한 후 다음을 선택합니다.

 Note

ecr-pullthroughcache/접두사를 사용하는 이름을 가진 Secrets Manager AWS Management Console 암호만 표시합니다. 또한 보안 암호는 풀스루 캐시 규칙이 생성된 동일한 계정 및 리전에 있어야 합니다.

- b. 새 보안 암호를 만들려면 AWS 보안 암호 생성을 선택하고 다음 작업을 수행한 후 다음을 선택합니다.
  - i. 보안 암호 이름에서 보안 암호를 설명하는 이름을 지정합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함해야 합니다.
  - ii. Docker Hub 사용자 이름에서 Docker Hub 사용자 이름을 지정합니다.
  - iii. Docker Hub 액세스 토크에서 Docker Hub 액세스 토크를 지정합니다. Docker Hub 액세스 토크 생성에 대한 자세한 내용은 Docker 설명서의 [Create and manage access tokens](#)를 참조하세요.
7. 3단계: 대상 지정 페이지에서 Amazon ECR 리포지토리 접두사에 대해 소스 퍼블릭 레지스트리에서 가져온 이미지를 캐싱할 때 사용할 리포지토리 네임스페이스를 지정하고 다음을 선택합니다.
 


네임스페이스는 기본적으로 작성되지만, 사용자 지정 네임스페이스를 지정할 수도 있습니다.
8. 4단계: 검토 및 생성 페이지에서 풀스루 캐시 규칙 구성을 검토한 다음 생성을 선택합니다.
9. 생성하려는 각 풀스루 캐시에 대해 이전 단계를 반복합니다. 풀스루 캐시 규칙은 각 리전에 대해 별도로 생성됩니다.

## GitHub 컨테이너 레지스트리용

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 프라이빗 레지스트리 설정을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 풀스루 캐시를 선택합니다.
4. 풀스루 캐시 구성(Pull through cache configuration) 페이지에서 규칙 추가(Add rule)를 선택합니다.
5. 1단계: 소스 페이지 지정에서 레지스트리에 대해 GitHub 컨테이너 레지스트리, 다음을 선택합니다.



6. 2단계: 인증 구성 페이지에서 업스트림 자격 증명의 경우 GitHub 컨테이너 레지스트리에 대한 인증 자격 증명을 AWS Secrets Manager 비밀로 저장해야 합니다. 기존 보안 암호를 지정하거나 Amazon ECR 콘솔을 사용하여 새 보안 암호를 생성할 수 있습니다.
  - a. 기존 암호를 사용하려면 기존 암호 사용을 선택합니다. AWS 보안 암호 이름의 경우 드롭다운을 사용하여 기존 보안 암호를 선택한 후 다음을 선택합니다.

 Note

ecr-pullthroughcache/접두사를 사용하는 이름을 가진 Secrets Manager AWS Management Console 암호만 표시합니다. 또한 보안 암호는 풀스루 캐시 규칙이 생성된 동일한 계정 및 리전에 있어야 합니다.

- b. 새 보안 암호를 만들려면 AWS 보안 암호 생성을 선택하고 다음 작업을 수행한 후 다음을 선택합니다.
    - i. 보안 암호 이름에서 보안 암호를 설명하는 이름을 지정합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함해야 합니다.
    - ii. GitHub 컨테이너 레지스트리 사용자 이름에는 GitHub 컨테이너 레지스트리 사용자 이름을 지정하십시오.
    - iii. GitHub 컨테이너 레지스트리 액세스 토큰의 경우 GitHub 컨테이너 레지스트리 액세스 토큰을 지정하십시오. GitHub 액세스 토큰 생성에 대한 자세한 내용은 GitHub 설명서의 [개인 액세스 토큰 관리](#)를 참조하십시오.
7. 3단계: 대상 지정 페이지에서 Amazon ECR 리포지토리 접두사에 대해 소스 퍼블릭 레지스트리에서 가져온 이미지를 캐싱할 때 사용할 리포지토리 네임스페이스를 지정하고 다음을 선택합니다.

네임스페이스는 기본적으로 작성되지만, 사용자 지정 네임스페이스를 지정할 수도 있습니다.

8. 4단계: 검토 및 생성 페이지에서 풀스루 캐시 규칙 구성을 검토한 다음 생성을 선택합니다.
9. 생성하려는 각 풀스루 캐시에 대해 이전 단계를 반복합니다. 풀스루 캐시 규칙은 각 리전에 대해 별도로 생성됩니다.

### Microsoft Azure 컨테이너 레지스트리의 경우

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 프라이빗 레지스트리 설정을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 풀스루 캐시를 선택합니다.

4. 풀스루 캐시 구성(Pull through cache configuration) 페이지에서 규칙 추가(Add rule)를 선택합니다.
5. 1단계: 소스 지정 페이지에서 다음 작업을 수행합니다.
  - a. 레지스트리에서 Microsoft Azure 컨테이너 레지스트리를 선택합니다.
  - b. 소스 레지스트리 URL에서 Microsoft Azure 컨테이너 레지스트리의 이름을 지정하고 다음을 선택합니다.

**⚠ Important**

.azurecr.io 접미사는 자동으로 채워지므로 접두사만 지정하면 됩니다.

6. 2단계: 인증 구성 페이지의 업스트림 보안 인증 정보에서 Microsoft Azure 컨테이너 레지스트리의 인증 관련 보안 인증 정보를 AWS Secrets Manager 보안 암호로 저장해야 합니다. 기존 보안 암호를 지정하거나 Amazon ECR 콘솔을 사용하여 새 보안 암호를 생성할 수 있습니다.
  - a. 기존 암호를 사용하려면 기존 암호 사용을 선택합니다. AWS 보안 암호 이름의 경우 드롭다운을 사용하여 기존 보안 암호를 선택한 후 다음을 선택합니다.

**i Note**

ecr-pullthroughcache/접두사를 사용하는 이름을 가진 Secrets Manager AWS Management Console 암호만 표시합니다. 또한 보안 암호는 풀스루 캐시 규칙이 생성된 동일한 계정 및 리전에 있어야 합니다.

- b. 새 보안 암호를 만들려면 AWS 보안 암호 생성을 선택하고 다음 작업을 수행한 후 다음을 선택합니다.
  - i. 보안 암호 이름에서 보안 암호를 설명하는 이름을 지정합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함해야 합니다.
  - ii. Microsoft Azure 컨테이너 레지스트리 사용자 이름에서 Microsoft Azure 컨테이너 레지스트리 사용자 이름을 지정합니다.
  - iii. Microsoft Azure 컨테이너 레지스트리 액세스 토큰에서 Microsoft Azure 컨테이너 레지스트리 액세스 토큰을 지정합니다. Microsoft Azure 컨테이너 레지스트리 액세스 토큰을 만드는 방법에 대한 자세한 내용은 Microsoft Azure 설명서의 [Create token - portal](#)을 참조하세요.

7. 3단계: 대상 지정 페이지에서 Amazon ECR 리포지토리 접두사에 대해 소스 퍼블릭 레지스트리에서 가져온 이미지를 캐싱할 때 사용할 리포지토리 네임스페이스를 지정하고 다음을 선택합니다.  
  
네임스페이스는 기본적으로 작성되지만, 사용자 지정 네임스페이스를 지정할 수도 있습니다.
8. 4단계: 검토 및 생성 페이지에서 풀스루 캐시 규칙 구성을 검토한 다음 생성을 선택합니다.
9. 생성하려는 각 풀스루 캐시에 대해 이전 단계를 반복합니다. 풀스루 캐시 규칙은 각 리전에 대해 별도로 생성됩니다.

## GitLab 컨테이너 레지스트리용

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 프라이빗 레지스트리 설정을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 풀스루 캐시를 선택합니다.
4. 풀스루 캐시 구성(Pull through cache configuration) 페이지에서 규칙 추가(Add rule)를 선택합니다.
5. 1단계: 소스 페이지 지정에서 레지스트리에 대해 GitLab 컨테이너 레지스트리, 다음을 선택합니다.
6. 2단계: 인증 구성 페이지에서 업스트림 자격 증명의 경우 GitLab 컨테이너 레지스트리에 대한 인증 자격 증명을 AWS Secrets Manager 비밀로 저장해야 합니다. 기존 보안 암호를 지정하거나 Amazon ECR 콘솔을 사용하여 새 보안 암호를 생성할 수 있습니다.
  - a. 기존 암호를 사용하려면 기존 암호 사용을 선택합니다. AWS 보안 암호 이름의 경우 드롭다운을 사용하여 기존 보안 암호를 선택한 후 다음을 선택합니다. Secrets Manager 콘솔을 사용하여 Secrets Manager 보안 암호를 생성하는 방법에 대한 자세한 내용은 [업스트림 리포지토리 자격 증명을 비밀로 AWS Secrets Manager 저장](#)을 참조하세요.

### Note

ecr-pullthroughcache/접두사를 사용하는 이름을 가진 Secrets Manager AWS Management Console 암호만 표시합니다. 또한 보안 암호는 풀스루 캐시 규칙이 생성된 동일한 계정 및 리전에 있어야 합니다.

- b. 새 보안 암호를 만들려면 AWS 보안 암호 생성을 선택하고 다음 작업을 수행한 후 다음을 선택합니다.
  - i. 보안 암호 이름에서 보안 암호를 설명하는 이름을 지정합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함해야 합니다.

- ii. GitLab 컨테이너 레지스트리 사용자 이름에는 GitLab 컨테이너 레지스트리 사용자 이름을 지정하십시오.
  - iii. GitLab 컨테이너 레지스트리 액세스 토큰의 경우 GitLab 컨테이너 레지스트리 액세스 토큰을 지정하십시오. GitLab 컨테이너 레지스트리 액세스 토큰 생성에 대한 자세한 내용은 GitLab 설명서의 [개인용 액세스 토큰](#), [그룹 액세스 토큰](#) 또는 [프로젝트 액세스 토큰](#)을 참조하십시오.
7. 3단계: 대상 지정 페이지에서 Amazon ECR 리포지토리 접두사에 대해 소스 퍼블릭 레지스트리에서 가져온 이미지를 캐싱할 때 사용할 리포지토리 네임스페이스를 지정하고 다음을 선택합니다.
- 네임스페이스는 기본적으로 작성되지만, 사용자 지정 네임스페이스를 지정할 수도 있습니다.
8. 4단계: 검토 및 생성 페이지에서 풀스루 캐시 규칙 구성을 검토한 다음 생성을 선택합니다.
9. 생성하려는 각 풀스루 캐시에 대해 이전 단계를 반복합니다. 풀스루 캐시 규칙은 각 리전에 대해 별도로 생성됩니다.

## 풀스루 캐시 규칙을 생성하는 방법(AWS CLI)

[create-pull-through cache-rule AWS CLI 명령을 사용하여 Amazon ECR 프라이빗 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다.](#) 인증이 필요한 업스트림 레지스트리의 경우 보안 인증 정보를 Secrets Manager 보안 암호로 저장해야 합니다. Secrets Manager 콘솔을 사용하여 시크릿을 만들려면 [여기](#)를 참조하십시오. [업스트림 리포지토리 자격 증명을 비밀로 AWS Secrets Manager 저장.](#)

지원되는 각 업스트림 레지스트리에 대해 다음 예제가 제공됩니다.

### Amazon ECR 퍼블릭의 경우

다음 예제에서는 Amazon ECR 퍼블릭 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 `ecr-public`으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 `ecr-public/upstream-repository-name`의 명명 체계를 사용하게 됩니다.

```
aws ecr create-pull-through-cache-rule \
  --ecr-repository-prefix ecr-public \
  --upstream-registry-url public.ecr.aws \
  --region us-east-2
```

## Kubernetes 컨테이너 레지스트리의 경우

다음 예제에서는 Kubernetes 퍼블릭 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 `kubernetes`으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 `kubernetes/upstream-repository-name`의 명명 체계를 사용하게 됩니다.

```
aws ecr create-pull-through-cache-rule \
  --ecr-repository-prefix kubernetes \
  --upstream-registry-url registry.k8s.io \
  --region us-east-2
```

## Quay의 경우

다음 예제에서는 Quay 퍼블릭 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 `quay`로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 `quay/upstream-repository-name`의 명명 체계를 사용하게 됩니다.

```
aws ecr create-pull-through-cache-rule \
  --ecr-repository-prefix quay \
  --upstream-registry-url quay.io \
  --region us-east-2
```

## Docker Hub의 경우

다음 예제에서는 Docker Hub 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 `docker-hub`으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 `docker-hub/upstream-repository-name`의 명명 체계를 사용하게 됩니다. Docker Hub 보안 인증 정보가 포함된 보안 암호의 전체 Amazon 리소스 이름(ARN)을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
  --ecr-repository-prefix docker-hub \
  --upstream-registry-url registry-1.docker.io \
  --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
  --region us-east-2
```

## GitHub 컨테이너 레지스트리의 경우

다음 예제는 GitHub 컨테이너 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 `docker-hub`으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가

github/*upstream-repository-name*의 명명 체계를 사용하게 됩니다. GitHub 컨테이너 레지스트리 자격 증명에 포함된 암호의 전체 Amazon 리소스 이름 (ARN) 을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
  --ecr-repository-prefix github \
  --upstream-registry-url ghcr.io \
  --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
  --region us-east-2
```

Microsoft Azure 컨테이너 레지스트리의 경우

다음 예에서는 Microsoft Azure 컨테이너 레지스트리에 대한 풀스루 캐시 규칙을 만듭니다. 리포지토리 접두사를 azure으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 azure/*upstream-repository-name*의 명명 체계를 사용하게 됩니다. Microsoft Azure 컨테이너 레지스트리 보안 인증 정보가 포함된 보안 암호의 전체 Amazon 리소스 이름(ARN)을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
  --ecr-repository-prefix azure \
  --upstream-registry-url myregistry.azurecr.io \
  --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
  --region us-east-2
```

GitLab 컨테이너 레지스트리의 경우

다음 예제는 GitLab 컨테이너 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 gitlab으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 gitlab/*upstream-repository-name*의 명명 체계를 사용하게 됩니다. GitLab 컨테이너 레지스트리 자격 증명에 포함된 암호의 전체 Amazon 리소스 이름 (ARN) 을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
  --ecr-repository-prefix gitlab \
  --upstream-registry-url registry.gitlab.com \
  --credential-arn arn:aws:secretsmanager:us-east-2:111122223333:secret:ecr-pullthroughcache/example1234 \
  --region us-east-2
```

## 다음 단계

풀스루 캐시 규칙을 생성한 후 다음 단계는 다음과 같습니다.

- 리포지토리 생성 템플릿을 생성합니다. 리포지토리 생성 템플릿을 사용하면 풀스루 캐시 작업 중에 Amazon ECR에서 사용자 대신 생성한 새 리포지토리에 사용할 설정을 정의할 수 있습니다. 자세한 정보는 [풀 스루 캐시 작업 중에 생성된 리포지토리를 제어하기 위한 템플릿](#)을 참조하세요.
- 풀스루 캐시 규칙의 유효성을 검증합니다. 풀스루 캐시 규칙의 유효성을 검증할 때 Amazon ECR은 업스트림 레지스트리와 네트워크 연결을 만들고 업스트림 레지스트리의 보안 인증 정보가 포함된 Secrets Manager 보안 암호에 액세스할 수 있는지와 인증이 성공했는지 확인합니다. 자세한 정보는 [Amazon ECR의 풀스루 캐시 규칙 검증](#)을 참조하세요.
- 풀스루 캐시 규칙 사용을 시작합니다. 자세한 정보는 [Amazon ECR에서 풀스루 캐시 규칙을 사용하여 이미지 가져오기](#)을 참조하세요.

## 풀 스루 캐시 작업 중에 생성된 리포지토리를 제어하기 위한 템플릿

리포지토리 생성 템플릿 기능은 Amazon ECR용 평가판 릴리스에 포함되어 있으며 변경될 수 있습니다. 이 공개 미리 보기에서는 리포지토리 생성 템플릿을 관리하는 데만 사용할 AWS Management Console 수 있습니다.

Amazon ECR 리포지토리 생성 템플릿을 사용하여 풀스루 캐시 작업 중에 Amazon ECR에서 사용자를 대신하여 생성한 리포지토리의 설정을 정의합니다. 리포지토리 생성 템플릿의 설정은 리포지토리를 생성하는 동안에만 적용되며 기존 리포지토리나 다른 방법을 사용하여 생성한 리포지토리에는 영향을 주지 않습니다.

다음 지역에서는 리포지토리 생성 템플릿이 지원되지 않습니다.

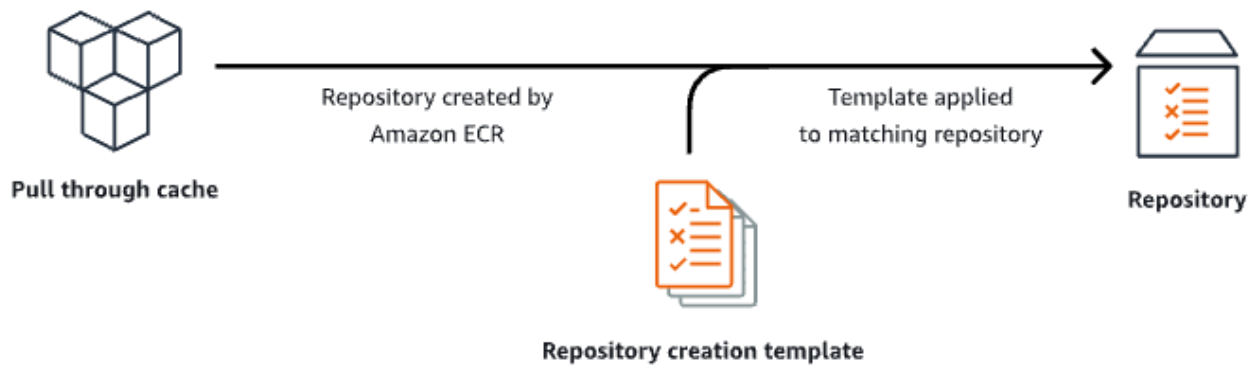
- 중국(베이징)(cn-north-1)
- 중국(닝샤)(cn-northwest-1)
- AWS GovCloud (미국 동부) (us-gov-east-1)
- AWS GovCloud (미국 서부) () us-gov-west-1

## 리포지토리 생성 템플릿 작동 방식

Amazon ECR에서 사용자를 대신하여 새 프라이빗 리포지토리를 생성해야 하는 경우가 있습니다. 풀스루 캐시 규칙을 처음으로 사용하여 업스트림 리포지토리의 콘텐츠를 검색하고 Amazon ECR 프라이빗 레지스트리에 저장하는 경우를 예로 들 수 있습니다. 풀스루 캐시 규칙과 일치하는 리포지토리 생성 템플릿이 없는 경우 Amazon ECR은 새 리포지토리의 기본 설정을 사용합니다. 이러한 기본 설정에는 태그 불변성 해제, AES-256 암호화 사용, 리포지토리 또는 수명 주기 정책 적용 안 함 등이 포함됩니다.

풀스루 캐시 규칙과 일치하는 접두사가 있는 리포지토리 생성 템플릿을 사용하면 풀스루 캐시 작업을 통해 생성된 새 리포지토리에 Amazon ECR이 적용하는 설정을 정의할 수 있습니다. 새 리포지토리의 태그 불변성, 암호화 구성, 리포지토리 권한, 수명 주기 정책, 리소스 태그를 정의할 수 있습니다.

다음 다이어그램은 리포지토리 생성 템플릿을 사용할 때 Amazon ECR에서 사용하는 워크플로를 보여줍니다.



다음은 리포지토리 생성 템플릿의 각 파라미터에 대해 자세히 설명합니다.

### 접두사

접두사는 템플릿과 연결할 리포지토리 네임스페이스 접두사입니다. 이 접두사를 사용하여 생성한 모든 리포지토리에 이 템플릿에 정의된 설정이 적용됩니다. 예를 들어 접두사 `prod`는 `prod/`로 시작하는 모든 리포지토리에 적용됩니다. 마찬가지로 접두사 `prod/team`은 `prod/team/`으로 시작하는 모든 리포지토리에 적용됩니다.

연결된 생성 템플릿이 없는 레지스트리의 모든 리포지토리에 템플릿을 적용하려면 `ROOT`를 접두사로 사용할 수 있습니다.



**⚠ Important**

접두사 끝에는 항상 위임된 /가 적용됩니다. ecr-public을 접두사로 지정하는 경우 Amazon ECR은 ecr-public/을 접두사로 취급합니다. 풀스루 캐시 규칙을 사용하는 경우 규칙 생성 중에 지정하는 리포지토리 접두사를 리포지토리 생성 템플릿 접두사로도 지정해야 합니다.

**설명**

이 템플릿 설명은 선택 사항이며 리포지토리 생성 템플릿의 용도를 설명하는 데 사용됩니다.

**템플릿 버전**

사용할 리포지토리 생성 템플릿 버전. 현재, TV1 템플릿 버전만 지원됩니다.

**구성 버전**

사용할 템플릿의 리포지토리 구성 버전. 각 템플릿에는 리포지토리 구성이 포함되어야 합니다. 기본 구성 버전은 CV1이고 이미지 태그 변경 가능성, 리포지토리 정책, 수명 주기 정책 설정으로 구성되어 있습니다.

**이미지 태그 변경 가능성**

템플릿을 사용하여 생성한 리포지토리에 사용할 태그 변경 가능성 설정입니다. 이 파라미터를 생략하면 이미지 태그를 덮어쓸 수 있는 변경 가능성이 기본 설정으로 사용됩니다. 이 기본 설정은 풀스루 캐시 작업으로 생성된 리포지토리에 사용되는 템플릿에 사용하는 것이 좋습니다. 이렇게 하면 태그가 동일할 때 Amazon ECR에서 캐시된 이미지를 업데이트할 수 있습니다.

변경 불가능을 지정하면 리포지토리 내의 모든 이미지 태그가 변경 불가능하므로 덮어쓰기가 방지됩니다.

**암호화 구성**

템플릿을 사용하여 생성한 리포지토리에 사용할 암호화 구성.

KMS 암호화 유형을 사용하면 AWS KMS에 저장된 AWS Key Management Service 키로 서버 측 암호화를 사용하여 리포지토리의 콘텐츠를 암호화합니다. 를 AWS KMS 사용하여 데이터를 암호화할 때는 Amazon ECR의 기본 AWS 관리 AWS KMS 키를 사용하거나 이미 생성한 자체 AWS KMS 키를 지정할 수 있습니다. 자세한 내용은 Amazon 심플 스토리지 서비스 사용 설명서의 [AWS Key Management Service \(SSE-KMS\) 에 저장된 AWS Key Management Service 키를 사용한 서버 측 암호화를 사용한 데이터 보호를](#) 참조하십시오.

AES256 암호화 유형을 사용하는 경우 Amazon ECR은 AES-256 암호화 알고리즘을 사용하여 리포지토리의 이미지를 암호화하는 Amazon S3 관리형 암호화 키로 서버 측 암호화를 사용합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 관리형 암호화 키\(SSE-S3\)로 서버 측 암호화를 사용하여 데이터 보호](#) 섹션을 참조하세요.

## 리포지토리 권한

템플릿을 사용하여 생성한 리포지토리에 적용할 리포지토리 정책입니다. 리포지토리 정책은 리소스 기반 권한을 사용하여 리포지토리에 대한 액세스를 제어합니다. 리소스 기반 권한을 사용하면 리포지토리에 액세스할 수 있는 IAM 사용자 또는 역할과 해당 리포지토리에서 수행 가능한 작업을 지정할 수 있습니다. 기본적으로 리포지토리를 생성한 AWS 계정만 리포지토리에 액세스할 수 있습니다. 사용자는 정책 문서를 적용하여 리포지토리에 대한 추가 권한을 허용하거나 거부할 수 있습니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리 정책](#)을 참조하세요.

## 리포지토리 수명 주기 정책

템플릿을 사용하여 생성한 리포지토리에 사용할 수명 주기 정책입니다. 수명 주기 정책은 프라이빗 리포지토리의 이미지에 대한 수명 주기 관리를 보다 효과적으로 제어할 수 있도록 합니다. 수명 주기 정책은 1개 이상의 규칙을 포함하며 각 규칙은 Amazon ECR에 대한 작업을 정의합니다. 이러한 규칙을 사용하면 연한 또는 수를 기반으로 만료하는 방식을 통해 컨테이너 이미지 정리 작업을 자동화할 수 있습니다. 자세한 정보는 [Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.](#)을 참조하세요.

## 리소스 태그

리소스 태그는 리포지토리에 적용하여 분류하고 구성하는 데 도움이 되는 메타데이터입니다. 각 태그는 사용자가 정의하는 키와 선택적 값으로 구성됩니다.

## 리포지토리 생성 템플릿 생성을 위한 IAM 권한

IAM 보안 주체가 리포지토리 생성 템플릿을 관리하려면 다음 권한이 필요합니다. 이 권한은 자격 증명 기반 IAM 정책을 사용하여 부여되어야 합니다.

- `ecr:CreateRepositoryCreationTemplate` - 리포지토리 생성 템플릿을 생성할 수 있는 권한을 부여합니다.
- `ecr>DeleteRepositoryCreationTemplate` - 리포지토리 생성 템플릿을 삭제할 수 있는 권한을 부여합니다.
- `ecr:PutLifecyclePolicy` - 수명 주기 정책을 생성하여 리포지토리에 적용할 수 있는 권한을 부여합니다. 이 권한은 리포지토리 생성 템플릿에 수명 주기 정책이 포함된 경우에만 필요합니다.

- `ecr:SetRepositoryPolicy` - 리포지토리에 대한 권한 정책을 생성할 수 있는 권한을 부여합니다. 이 권한은 리포지토리 생성 템플릿에 리포지토리 정책이 포함된 경우에만 필요합니다.
- `ecr:TagResource` - 리소스에 메타데이터 태그를 추가할 수 있는 권한을 부여합니다. 이 권한은 리포지토리 생성 템플릿에 리소스 태그가 포함된 경우에만 필요합니다.

## Amazon ECR에서 리포지토리 생성 템플릿 생성

리포지토리 생성 템플릿을 만들어 풀스루 캐시 작업 중에 Amazon ECR에서 사용자 대신 생성한 리포지토리에 사용할 설정을 정의할 수 있습니다. 리포지토리 생성 템플릿이 생성되면 풀스루 캐시 작업 중에 생성되는 모든 새 리포지토리에 설정이 적용됩니다. 이는 이전에 만든 리포지토리는 영향을 주지 않습니다.

리포지토리 생성 템플릿(AWS Management Console)을 만들려면

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 리포지토리 생성 템플릿을 만들려는 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리, 리포지토리 생성 템플릿을 선택합니다.
4. 리포지토리 생성 템플릿 페이지에서 템플릿 생성을 선택합니다.
5. 1단계: 템플릿 정의 페이지에서 템플릿 세부 정보에 대해 특정 리포지토리 네임스페이스 접두사에 템플릿을 적용할 특정 접두사를 선택하고, ECR 레지스트리의 모든 접두사를 선택하여 해당 리전의 다른 템플릿과 일치하지 않는 모든 리포지토리에 템플릿을 적용합니다.
  - a. 특정 접두사를 선택한 경우 접두사에 템플릿을 적용할 리포지토리 네임스페이스 접두사를 지정합니다. 접두사 끝에는 항상 위임된 /가 적용됩니다. 예를 들어 접두사 `prod`는 `prod/`로 시작하는 모든 리포지토리에 적용됩니다. 마찬가지로 접두사 `prod/team`은 `prod/team/`으로 시작하는 모든 리포지토리에 적용됩니다.
  - b. ECR 레지스트리에서 어떤 접두사를 선택하든 해당 접두사가 ROOT로 설정됩니다.
6. 템플릿 설명에서 템플릿에 대한 설명(선택 사항)을 지정하고 다음을 선택합니다.
7. 2단계: 리포지토리 생성 구성 추가 페이지에서 템플릿을 사용하여 생성한 리포지토리에 적용되는 리포지토리 설정 구성을 지정합니다.
  - a. 이미지 태그 변경 가능성에서 사용할 태그 변경 가능성 설정을 선택합니다. 자세한 정보는 [Amazon ECR에서 이미지 태그의 덮어쓰기 방지](#)를 참조하세요.


변경 가능을 선택하면 이미지 태그를 덮어쓸 수 있습니다. 이 기본 설정은 풀 스루 캐시 작업으로 생성된 리포지토리에 사용되는 템플릿에 사용하는 것이 좋습니다. 이렇게 하면 태그가 동일할 때 Amazon ECR에서 캐시된 이미지를 업데이트할 수 있습니다.

변경 불가능을 선택하면 이미지 태그를 덮어쓸 수 없습니다. 리포지토리가 변경 불가능 태그로 구성된 후 리포지토리에 이미 존재하는 태그가 지정된 이미지를 푸시하려고 시도할 때 ImageTagAlreadyExistsException 오류가 반환됩니다. 리포지토리에 대해 태그 불변성을 켜면 모든 태그에 영향을 미치며 일부 태그는 변경할 수 있지만 다른 태그는 변경할 수 없습니다.

- b. 암호화 구성에서 사용할 암호화 설정을 선택합니다. 자세한 정보는 [저장된 데이터 암호화](#)을 참조하세요.

AES-256을 선택하면 Amazon ECR은 Amazon Simple Storage Service 관리형 암호화 키로 서버 측 암호화를 사용합니다. 이 암호화 키는 업계 표준 AES-256 암호화 알고리즘을 사용하여 저장 데이터를 암호화합니다. 이 서비스는 추가 비용 없이 제공됩니다.

AWS KMS를 선택하면 Amazon ECR은 키가 () 에 저장된 서버 측 암호화를 사용합니다. AWS Key Management Service AWS KMS를 사용하여 데이터를 AWS KMS 암호화하는 경우 Amazon ECR에서 AWS 관리하는 기본 관리 키를 사용하거나 사용자 고유의 키 (고객 관리 AWS KMS 키라고 함) 를 지정할 수 있습니다.

 Note

리포지토리가 생성된 후에는 리포지토리의 암호화 설정을 변경할 수 없습니다.

- c. 리포지토리 권한에서 이 템플릿을 사용하여 생성한 리포지토리에 적용할 리포지토리 권한 정책을 지정합니다. 선택적으로 드롭다운을 사용하여 가장 일반적인 사용 사례의 JSON 샘플 중 하나를 선택할 수 있습니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리 정책](#)을 참조하세요.
- d. 리포지토리 수명 주기 정책에서 이 템플릿을 사용하여 생성한 리포지토리에 적용할 리포지토리 수명 주기 정책을 지정합니다. 선택적으로 드롭다운을 사용하여 가장 일반적인 사용 사례의 JSON 샘플 중 하나를 선택할 수 있습니다. 자세한 정보는 [Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.](#)을 참조하세요.
- e. 리포지토리 AWS 태그의 경우 이 템플릿을 사용하여 생성된 리포지토리와 연결할 메타데이터를 키-값 쌍의 형태로 지정한 후 다음을 선택합니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리에 태그 지정](#)을 참조하세요.

- 3단계: 검토 및 생성 페이지에서 리포지토리 생성 템플릿에 지정한 설정을 검토합니다. 편집 옵션을 선택하여 변경합니다. 완료하면 생성을 선택합니다.

## Amazon ECR에서 리포지토리 생성 템플릿 삭제

리포지토리 생성 템플릿 사용을 완료하면 이를 삭제할 수 있습니다. 리포지토리 생성 템플릿이 삭제되면 풀스루 캐시 작업 중에 생성되는 모든 리포지토리에 기본 설정이 적용됩니다.

리포지토리 생성 템플릿(AWS Management Console)을 삭제하려면

- Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
- 탐색 모음에서 삭제하려는 리포지토리 생성 템플릿이 있는 리전을 선택합니다.
- 탐색 창에서 프라이빗 레지스트리, 리포지토리 생성 템플릿을 선택합니다.
- 리포지토리 생성 템플릿 페이지에서 삭제할 리포지토리 생성 템플릿을 선택합니다.
- 작업 드롭다운 메뉴에서 삭제를 선택합니다.

## Amazon ECR의 풀스루 캐시 규칙 검증

풀스루 캐시 규칙을 생성한 후 인증이 필요한 업스트림 레지스트리의 경우 규칙이 제대로 작동하는지 확인할 수 있습니다. 풀스루 캐시 규칙을 검증할 때 Amazon ECR은 업스트림 레지스트리와 네트워크 연결을 만들고 업스트림 레지스트리의 자격 증명에 포함된 Secrets Manager 암호에 액세스할 수 있는지 확인하고 인증이 성공했는지 확인합니다.

풀스루 캐시 규칙 작업을 시작하기 전에 적절한 IAM 권한이 있는지 확인하십시오. 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화하는 데 필요한 IAM 권한](#)을 참조하세요.

풀스루 캐시 규칙(AWS Management Console)을 검증하려면

다음 단계는 Amazon ECR 콘솔을 사용하여 풀스루 캐시 규칙을 검증하는 방법을 보여줍니다.

- Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
- 탐색 모음에서 검증할 풀스루 캐시 규칙이 포함된 리전을 선택합니다.
- 탐색 창에서 프라이빗 레지스트리, 풀스루 캐시를 선택합니다.
- 풀스루 캐시 구성 페이지에서 검증할 풀스루 캐시 규칙을 선택합니다. 그런 다음 작업 드롭다운 메뉴를 사용하여 세부 정보 보기를 선택합니다.

5. 풀스루 캐시 규칙 세부 정보 페이지에서 작업 드롭다운 메뉴를 사용하고 인증 확인을 선택합니다. Amazon ECR은 결과가 포함된 배너를 표시합니다.
6. 검증하려는 각 풀스루 캐시 규칙에 대해 이러한 단계를 반복합니다.

## 풀스루 캐시 규칙(AWS CLI)을 검증하려면

[validate-pull-through-cache-rule](#) AWS CLI 명령은 Amazon ECR 사실 레지스트리의 풀스루 캐시 규칙을 검증하는 데 사용됩니다. 다음 예시에서는 `ecr-public` 네임스페이스 접두사를 사용합니다. 이 값을 검증할 풀스루 캐시 규칙의 접두사 값으로 바꿉니다.

```
aws ecr validate-pull-through-cache-rule \
  --ecr-repository-prefix ecr-public \
  --region us-east-2
```

응답에서 `isValid` 파라미터는 검증의 성공 여부를 나타냅니다. `true`인 경우 Amazon ECR은 업스트림 레지스트리에 연결할 수 있으며 인증에 성공했습니다. `false`인 경우 문제가 발생하여 검증에 실패했습니다. `failure` 파라미터는 원인을 나타냅니다.

## Amazon ECR에서 풀스루 캐시 규칙을 사용하여 이미지 가져오기

다음 예제는 풀스루 캐시 규칙을 사용하여 이미지를 가져올 때 사용하는 명령 구문을 보여줍니다. 풀스루 캐시 규칙을 사용하여 업스트림 이미지를 가져오는 도중에 오류가 발생하면 [Amazon ECR의 풀스루 캐시 문제 해결](#) 섹션을 참조하여 가장 일반적인 오류 및 해결 방법을 확인하세요.

풀스루 캐시 규칙 작업을 시작하기 전에 적절한 IAM 권한이 있는지 확인하세요. 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화하는 데 필요한 IAM 권한](#)을 참조하세요.

### Note

다음 예제는 에서 사용하는 기본 Amazon ECR 리포지토리 네임스페이스 값을 사용합니다. AWS Management Console 직접 구성한 Amazon ECR 프라이빗 리포지토리 URI를 사용해야 합니다.

## Amazon ECR 퍼블릭의 경우

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/ecr-public/repository_name/image_name:tag
```

## Kubernetes 컨테이너 레지스트리

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/kubernetes/repository_name/  
image_name:tag
```

## Quay

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/quay/repository_name/  
image_name:tag
```

## Docker Hub

Docker Hub 공식 이미지의 경우:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/  
library/image_name:tag
```

### Note

Docker Hub 공식 이미지의 경우 /library 접두사를 포함해야 합니다. 다른 모든 Docker Hub 리포지토리의 경우 /library 접두사를 생략해야 합니다.

기타 모든 Docker Hub 이미지의 경우:

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/repository_name/  
image_name:tag
```

## GitHub 컨테이너 레지스트리

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/github/repository_name/  
image_name:tag
```

## Microsoft Azure 컨테이너 레지스트리

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/azure/repository_name/  
image_name:tag
```

## GitLab 컨테이너 레지스트리

```
docker pull aws_account_id.dkr.ecr.region.amazonaws.com/gitlab/repository_name/image_name:tag
```

## 업스트림 리포지토리 자격 증명을 비밀로 AWS Secrets Manager 저장

인증이 필요한 업스트림 리포지토리에 대한 풀스루 캐시 규칙을 생성할 때는 보안 인증 정보를 Secrets Manager 보안 암호로 저장해야 합니다. Secrets Manager 보안 암호를 사용하면 비용이 발생할 수 있습니다. 자세한 내용은 [AWS Secrets Manager 요금](#)을 참조하십시오.

다음 절차는 지원되는 각 업스트림 리포지토리에 대해 Secrets Manager 보안 암호를 생성하는 방법을 안내합니다. Secrets Manager 콘솔을 사용하여 보안 암호를 생성하는 대신 Amazon ECR 콘솔의 풀스루 캐시 규칙 생성 워크플로를 사용하여 보안 암호를 생성할 수도 있습니다. 자세한 정보는 [Amazon ECR에서 풀스루 캐시 규칙 생성](#)을 참조하세요.

### Docker Hub

Docker Hub 보안 인증 정보(AWS Management Console)를 위한 Secrets Manager 보안 암호를 생성하려면

1. <https://console.aws.amazon.com/secretsmanager/>에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택 페이지에서 다음을 수행합니다.
  - a. 보안 암호 유형(Secret type)에서 다른 유형의 보안 암호(Other type of secret)를 선택합니다.
  - b. 키/값 페어에서 Docker Hub 보안 인증 정보를 위한 행을 두 개 생성합니다. 보안 암호에는 최대 65,536바이트까지 저장할 수 있습니다.
    - i. 첫 번째 키/값 페어의 경우 username을 키로 지정하고 Docker Hub 사용자 이름을 값으로 지정합니다.
    - ii. 두 번째 키/값 페어의 경우 accessToken을 키로 지정하고 Docker Hub 액세스 토큰을 값으로 지정합니다. Docker Hub 액세스 토큰 생성에 대한 자세한 내용은 Docker 설명서의 [Create and manage access tokens](#)를 참조하세요.



- c. 암호화 키의 경우 기본 aws/secretsmanager AWS KMS key 값을 유지하고 다음을 선택합니다. 이 키를 사용하는 데 드는 비용은 없습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager의 보안 암호 암호화 및 복호화](#)를 참조하세요.

**⚠ Important**

보안 암호를 암호화하려면 기본 aws/secretsmanager 암호화 키를 사용해야 합니다. Amazon ECR은 이를 위한 고객 관리형 키(CMK) 사용을 지원하지 않습니다.

- 4. 보안 암호 구성 페이지에서 다음을 수행합니다.
  - a. 설명이 포함된 Secret name(보안 암호 이름)과 Description(설명)을 입력합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함하고 ecr-pullthroughcache/가 접두사로 지정되어야 합니다.

**⚠ Important**

Amazon ECR은 접두사를 사용하는 이름을 가진 ecr-pullthroughcache/ Secrets Manager AWS Management Console 비밀만 표시합니다.

- b. (선택 사항) Tags(태그) 섹션에서 보안 암호에 태그를 추가합니다. 태깅 전략은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 태그 지정](#)을 참조하세요. 민감한 정보는 암호화되지 않으므로 태그에 저장하지 마세요.
  - c. (선택 사항) 리소스 권한(Resource permissions)에서 리소스 정책을 보안 암호에 추가하려면 권한 편집(Edit permissions)을 선택합니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호에 권한 정책 연결](#)을 참조하세요.
  - d. (선택 사항) 암호 복제에서 암호를 다른 AWS 리전암호로 복제하려면 암호 복제를 선택합니다. 보안 암호를 지금 복제하거나 페이지로 다시 돌아와서 나중에 복제할 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [보안 암호를 다른 리전으로 복제](#)를 참조하세요.
  - e. Next(다음)를 선택합니다.
- 5. (선택 사항) 교체 구성(Configure rotation) 페이지에서 자동 교체를 켤 수 있습니다. 현재 교체를 끈 다음 나중에 켤 수도 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 교체](#)를 참조하세요. 다음을 선택합니다.
  - 6. Review(검토) 페이지에서 보안 암호 세부 정보를 검토한 후 Store(저장)를 선택합니다.

Secrets Manager는 보안 암호 목록으로 돌아갑니다. 암호가 표시되지 않으면 Refresh(새로 고침)를 선택합니다.

## GitHub Container Registry

GitHub 컨테이너 레지스트리 자격 증명을 위한 Secrets Manager 시크릿을 만들려면 (AWS Management Console)

1. <https://console.aws.amazon.com/secretsmanager/>에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택 페이지에서 다음을 수행합니다.
  - a. 보안 암호 유형(Secret type)에서 다른 유형의 보안 암호(Other type of secret)를 선택합니다.
  - b. 키/값 쌍으로 자격 증명을 위한 행을 두 개 생성합니다. GitHub 보안 암호에는 최대 65,536 바이트까지 저장할 수 있습니다.
    - i. 첫 번째 키/값 쌍의 경우 키로 지정하고 GitHub 사용자 이름을 username 값으로 지정합니다.
    - ii. 두 번째 키/값 쌍의 경우 키로 지정하고 GitHub 액세스 토큰을 accessToken 값으로 지정합니다. GitHub 액세스 토큰 생성에 대한 자세한 내용은 설명서의 [개인용 액세스 토큰 관리](#)를 참조하십시오. GitHub
  - c. 암호화 키의 경우 기본 aws/secretsmanager AWS KMS key 값을 유지하고 다음을 선택합니다. 이 키를 사용하는 데 드는 비용은 없습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager의 보안 암호 암호화 및 복호화](#)를 참조하세요.

### Important

보안 암호를 암호화하려면 기본 aws/secretsmanager 암호화 키를 사용해야 합니다. Amazon ECR은 이를 위한 고객 관리형 키(CMK) 사용을 지원하지 않습니다.

4. 보안 구성(Configure secret) 페이지에서 다음을 수행합니다.
  - a. 설명이 포함된 Secret name(보안 암호 이름)과 Description(설명)을 입력합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함하고 ecr-pullthroughcache/가 접두사로 지정되어야 합니다.

**⚠ Important**

Amazon ECR은 접두사를 사용하는 이름을 가진 `ecr-pullthroughcache/Secrets Manager` AWS Management Console 비밀만 표시합니다.

- b. (선택 사항) Tags(태그) 섹션에서 보안 암호에 태그를 추가합니다. 태깅 전략은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 태그 지정](#)을 참조하세요. 민감한 정보는 암호화되지 않으므로 태그에 저장하지 마세요.
  - c. (선택 사항) 리소스 권한(Resource permissions)에서 리소스 정책을 보안 암호에 추가하려면 권한 편집(Edit permissions)을 선택합니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호에 권한 정책 연결](#)을 참조하세요.
  - d. (선택 사항) 암호 복제에서 암호를 다른 AWS 리전암호로 복제하려면 암호 복제를 선택합니다. 보안 암호를 지금 복제하거나 페이지로 다시 돌아와서 나중에 복제할 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [보안 암호를 다른 리전으로 복제](#)를 참조하세요.
  - e. Next(다음)를 선택합니다.
5. (선택 사항) 교체 구성(Configure rotation) 페이지에서 자동 교체를 켤 수 있습니다. 현재 교체를 끈 다음 나중에 켤 수도 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 교체](#)를 참조하세요. 다음을 선택합니다.
  6. Review(검토) 페이지에서 보안 암호 세부 정보를 검토한 후 Store(저장)를 선택합니다.

Secrets Manager는 보안 암호 목록으로 돌아갑니다. 암호가 표시되지 않으면 Refresh(새로 고침)를 선택합니다.

## Microsoft Azure Container Registry

Microsoft Azure 컨테이너 레지스트리 보안 인증 정보(AWS Management Console)를 위한 Secrets Manager 보안 암호를 만들려면

1. <https://console.aws.amazon.com/secretsmanager/>에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택 페이지에서 다음을 수행합니다.
  - a. 보안 암호 유형(Secret type)에서 다른 유형의 보안 암호(Other type of secret)를 선택합니다.

- b. 키/값 페어에서 Microsoft Azure 보안 인증 정보를 위한 행을 두 개 생성합니다. 보안 암호에는 최대 65,536바이트까지 저장할 수 있습니다.
  - i. 첫 번째 키/값 페어의 경우 username을 키로 지정하고 Microsoft Azure 컨테이너 레지스트리 사용자 이름을 값으로 지정합니다.
  - ii. 두 번째 키/값 페어의 경우 accessToken을 키로 지정하고 Microsoft Azure 컨테이너 레지스트리 액세스 토큰을 값으로 지정합니다. Microsoft Azure 액세스 토큰을 만드는 방법에 대한 자세한 내용은 Microsoft Azure 설명서의 [Create token - portal](#)을 참조하세요.
- c. 암호화 키의 경우 기본 aws/secretsmanager AWS KMS key 값을 유지하고 다음을 선택합니다. 이 키를 사용하는 데 드는 비용은 없습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager의 보안 암호 암호화 및 복호화](#)를 참조하세요.

**⚠ Important**

보안 암호를 암호화하려면 기본 aws/secretsmanager 암호화 키를 사용해야 합니다. Amazon ECR은 이를 위한 고객 관리형 키(CMK) 사용을 지원하지 않습니다.

4. 보안 구성(Configure secret) 페이지에서 다음을 수행합니다.
  - a. 설명이 포함된 Secret name(보안 암호 이름)과 Description(설명)을 입력합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함하고 ecr-pullthroughcache/가 접두사로 지정되어야 합니다.

**⚠ Important**

Amazon ECR은 접두사를 사용하는 이름을 가진 ecr-pullthroughcache/ Secrets Manager AWS Management Console 비밀만 표시합니다.

- b. (선택 사항) Tags(태그) 섹션에서 보안 암호에 태그를 추가합니다. 태깅 전략은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 태그 지정](#)을 참조하세요. 민감한 정보는 암호화되지 않으므로 태그에 저장하지 마세요.
- c. (선택 사항) 리소스 권한(Resource permissions)에서 리소스 정책을 보안 암호에 추가하려면 권한 편집(Edit permissions)을 선택합니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호에 권한 정책 연결](#)을 참조하세요.

- d. (선택 사항) 암호 복제에서 암호를 다른 AWS 리전암호로 복제하려면 암호 복제를 선택합니다. 보안 암호를 지금 복제하거나 페이지로 다시 돌아와서 나중에 복제할 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [보안 암호를 다른 리전으로 복제를 참조](#)하세요.
  - e. Next(다음)를 선택합니다.
5. (선택 사항) 교체 구성(Configure rotation) 페이지에서 자동 교체를 켤 수 있습니다. 현재 교체를 끈 다음 나중에 켤 수도 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 교체](#)를 참조하세요. 다음을 선택합니다.
  6. Review(검토) 페이지에서 보안 암호 세부 정보를 검토한 후 Store(저장)를 선택합니다.

Secrets Manager는 보안 암호 목록으로 돌아갑니다. 암호가 표시되지 않으면 Refresh(새로 고침)를 선택합니다.

## GitLab Container Registry

GitLab 컨테이너 레지스트리 자격 증명을 위한 Secrets Manager 시크릿을 만들려면 (AWS Management Console)

1. <https://console.aws.amazon.com/secretsmanager/>에서 Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택 페이지에서 다음을 수행합니다.
  - a. 보안 암호 유형(Secret type)에서 다른 유형의 보안 암호(Other type of secret)를 선택합니다.
  - b. 키/값 쌍으로 자격 증명을 위한 행을 두 개 생성합니다. GitLab 보안 암호에는 최대 65,536 바이트까지 저장할 수 있습니다.
    - i. 첫 번째 키/값 쌍의 경우 키로 지정하고 GitLab 컨테이너 레지스트리 사용자 이름을 username 값으로 지정합니다.
    - ii. 두 번째 키/값 쌍의 경우 키로 지정하고 GitLab 컨테이너 레지스트리 액세스 토큰을 accessToken 값으로 지정합니다. GitLab 컨테이너 레지스트리 액세스 토큰 생성에 대한 자세한 내용은 설명서의 [개인용 액세스 토큰](#), [그룹 액세스 토큰](#) 또는 [프로젝트 액세스 토큰](#)을 참조하십시오. GitLab
  - c. 암호화 키의 경우 기본 aws/secretsmanager AWS KMS key 값을 유지하고 다음을 선택합니다. 이 키를 사용하는 데 드는 비용은 없습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager의 보안 암호 암호화 및 복호화](#)를 참조하세요.

**⚠ Important**

보안 암호를 암호화하려면 기본 `aws/secretsmanager` 암호화 키를 사용해야 합니다. Amazon ECR은 이를 위한 고객 관리형 키(CMK) 사용을 지원하지 않습니다.

4. 보안 구성(Configure secret) 페이지에서 다음을 수행합니다.

- a. 설명이 포함된 Secret name(보안 암호 이름)과 Description(설명)을 입력합니다. 보안 암호 이름은 1~512자의 유니코드 문자를 포함하고 `ecr-pullthroughcache/`가 접두사로 지정되어야 합니다.

**⚠ Important**

Amazon ECR은 접두사를 사용하는 이름을 가진 `ecr-pullthroughcache/` Secrets Manager AWS Management Console 비밀만 표시합니다.

- b. (선택 사항) Tags(태그) 섹션에서 보안 암호에 태그를 추가합니다. 태깅 전략은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 태그 지정](#)을 참조하세요. 민감한 정보는 암호화되지 않으므로 태그에 저장하지 마세요.
  - c. (선택 사항) 리소스 권한(Resource permissions)에서 리소스 정책을 보안 암호에 추가하려면 권한 편집(Edit permissions)을 선택합니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호에 권한 정책 연결](#)을 참조하세요.
  - d. (선택 사항) 암호 복제에서 암호를 다른 AWS 리전암호로 복제하려면 암호 복제를 선택합니다. 보안 암호를 지금 복제하거나 페이지로 다시 돌아와서 나중에 복제할 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [보안 암호를 다른 리전으로 복제](#)를 참조하세요.
  - e. Next(다음)를 선택합니다.
5. (선택 사항) 교체 구성(Configure rotation) 페이지에서 자동 교체를 켤 수 있습니다. 현재 교체를 끈 다음 나중에 켤 수도 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [Secrets Manager 보안 암호 교체](#)를 참조하세요. 다음을 선택합니다.
6. Review(검토) 페이지에서 보안 암호 세부 정보를 검토한 후 Store(저장)를 선택합니다.

Secrets Manager는 보안 암호 목록으로 돌아갑니다. 암호가 표시되지 않으면 Refresh(새로 고침)를 선택합니다.

## Amazon ECR의 풀스루 캐시 문제 해결

풀스루 캐시 규칙을 사용하여 업스트림 이미지를 가져올 때 수신할 수 있는 가장 일반적인 오류는 다음과 같습니다.

리포지토리가 존재하지 않습니다.

리포지토리가 존재하지 않는다는 오류는 Amazon ECR 프라이빗 레지스트리에 리포지토리가 없거나 업스트림 이미지를 가져오는 IAM 보안 주체에게 `ecr:CreateRepository` 권한이 부여되지 않았기 때문에 가장 자주 발생합니다. 이 오류를 해결하려면 `pull` 명령의 리포지토리 URI가 올바른지, 업스트림 이미지를 가져오는 IAM 보안 주체에 필요한 IAM 권한이 부여되었는지, 또는 푸시될 업스트림 이미지에 대한 리포지토리가 업스트림 이미지 풀을 수행하기 전에 Amazon ECR 프라이빗 레지스트리에 생성되었는지 확인해야 합니다. 필요한 IAM 권한에 대한 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화하는 데 필요한 IAM 권한](#) 섹션을 참조하세요.

다음은 이 오류의 예입니다.

```
Error response from daemon: repository 111122223333.dkr.ecr.us-east-1.amazonaws.com/ecr-public/amazonlinux/amazonlinux not found: name unknown: The repository with name 'ecr-public/amazonlinux/amazonlinux' does not exist in the registry with id '111122223333'
```

요청한 이미지를 찾을 수 없음

이미지를 찾을 수 없음을 나타내는 오류는 이미지가 업스트림 레지스트리에 존재하지 않거나 업스트림 이미지를 가져오는 IAM 보안 주체에게 `ecr:BatchImportUpstreamImage` 권한이 부여되지 않았지만 리포지토리가 이미 Amazon ECR 프라이빗에서 생성되고 있기 때문에 가장 자주 발생합니다. 이 오류를 해결하려면 업스트림 이미지 및 이미지 태그 이름이 올바른지, 해당 항목이 존재하는지, 그리고 업스트림 이미지를 가져오는 IAM 보안 주체에 필요한 IAM 권한이 부여되었는지를 확인해야 합니다. 필요한 IAM 권한에 대한 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화하는 데 필요한 IAM 권한](#) 섹션을 참조하세요.

다음은 이 오류의 예입니다.

```
Error response from daemon: manifest for 111122223333.dkr.ecr.us-east-1.amazonaws.com/ecr-public/amazonlinux/amazonlinux:latest not found: manifest unknown: Requested image not found
```

### 4.0.3 Docker Hub 저장소에서 가져올 때 금지됨

Docker 공식 이미지로 태그가 지정된 Docker Hub 리포지토리에서 가져올 때는 사용하는 URI에 `/library/`를 포함해야 합니다. 예를 들어 `aws_account_id.dkr.ecr.region.amazonaws.com/docker-hub/library/image_name:tag`입니다. Docker Hub 공식 이미지에 대해 `/library/`를 생략할 경우 폴스루 캐시 규칙을 사용하여 이미지를 가져오려고 하면 403 Forbidden 오류가 반환됩니다. 자세한 정보는 [Amazon ECR에서 폴스루 캐시 규칙을 사용하여 이미지 가져오기](#)을 참조하세요.

다음은 이 오류의 예입니다.

```
Error response from daemon: failed to resolve reference "111122223333.dkr.ecr.us-west-2.amazonaws.com/docker-hub/amazonlinux:2023": pulling from host
111122223333.dkr.ecr.us-west-2.amazonaws.com failed with status code [manifests 2023]: 403 Forbidden
```



# Amazon ECR에서의 프라이빗 이미지 복제

리포지토리 복제를 지원하도록 Amazon ECR 프라이빗 레지스트리를 구성할 수 있습니다. Amazon ECR은 리전 간 복제와 교차 계정 복제를 모두 지원합니다. 교차 계정 복제가 발생하려면 대상 계정이 소스 레지스트리에서 복제가 수행되도록 레지스트리 권한 정책을 구성해야 합니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 권한](#)을 참조하세요.

## 주제

- [프라이빗 이미지 복제 관련 고려 사항](#)
- [Amazon ECR의 프라이빗 이미지 복제 예제](#)
- [Amazon ECR에서 프라이빗 이미지 복제 구성](#)

## 프라이빗 이미지 복제 관련 고려 사항

프라이빗 이미지 복제를 사용할 때는 다음 사항을 고려해야 합니다.

- 복제가 구성된 이후에 리포지토리로 푸시된 리포지토리 콘텐츠만 복제됩니다. 리포지토리의 기존 콘텐츠는 복제되지 않습니다. 리포지토리에 대한 복제가 구성되면 Amazon ECR은 대상과 소스를 동기화된 상태로 유지합니다.
- 리포지토리 이름은 복제가 수행된 경우 여러 리전과 계정에서 동일하게 유지됩니다. Amazon ECR은 복제 중에 리포지토리 이름 변경을 지원하지 않습니다.
- Amazon ECR은 복제에 대한 프라이빗 레지스트리를 처음 구성할 때 사용자를 대신하여 서비스 연결 IAM 역할을 생성합니다. 서비스 연결 IAM 역할은 Amazon ECR 복제 서비스에 리포지토리를 생성하고 레지스트리에서 이미지를 복제하는 데 필요한 권한을 부여합니다. 자세한 정보는 [Amazon ECR에 대한 서비스 연결 역할 사용](#)을 참조하세요.
- 교차 계정 복제가 발생하려면 프라이빗 레지스트리 대상에서 원본 레지스트리에서 해당 이미지를 복제할 수 있는 권한을 부여해야 합니다. 이 작업은 프라이빗 레지스트리 권한 정책을 설정하여 수행됩니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 권한](#)을 참조하세요.
- 프라이빗 레지스트리에 대한 권한 정책이 권한을 제거하도록 변경되면 이전에 부여된 진행 중인 복제가 완료될 수 있습니다.
- 리전 간 복제를 수행하려면 해당 리전 내에서 또는 리전에 대한 복제 작업을 수행하기 전에 소스 계정과 대상 계정을 모두 리전에 옵트인해야 합니다. 자세한 내용은 Amazon Web Services 일반 참조의 [AWS 리전 관리](#)를 참조하세요.

- 파티션 간에는 지역 간 복제가 지원되지 않습니다. AWS 예를 들어, us-west-2의 리포지토리는 cn-north-1에 복제할 수 없습니다. AWS 파티션에 대한 자세한 내용은 AWS 일반 참조의 [ARN 형식](#)을 참조하십시오.
- 프라이빗 레지스트리에 대한 복제 구성에는 모든 규칙에서 최대 25개의 고유 대상이 포함될 수 있으며 최대 10개의 규칙이 포함됩니다. 각 규칙에는 최대 100개의 필터가 포함될 수 있습니다. 이를 통해 예를 들어, 프로덕션 및 테스트에 사용되는 이미지가 포함된 리포지토리에 대해 별도의 규칙을 지정할 수 있습니다.
- 복제 구성은 리포지토리 접두사를 지정하여 프라이빗 레지스트리에서 복제되는 리포지토리 필터링을 지원합니다. 예시는 [예제: 리포지토리 필터를 사용하여 교차 리전 복제 구성](#) 단원을 참조하세요.
- 복제 작업은 이미지 푸시당 한 번만 발생합니다. 예를 들어, us-west-2에서 us-east-1로 그리고 us-east-1에서 us-east-2로 교차 리전 복제를 구성한 경우 us-west-2로 푸시된 이미지는 us-east-1으로만 복제를 하고 us-east-2로 다시 복제하지 않습니다. 이 동작은 교차 리전 및 교차 계정 복제에 모두 적용됩니다.
- 대부분의 이미지는 30분 이내에 복제되지만 드문 경우 복제에 시간이 더 오래 걸릴 수 있습니다.
- 레지스트리 복제는 삭제 작업을 수행하지 않습니다. 복제된 이미지와 리포지토리는 더 이상 사용되지 않을 때 수동으로 삭제할 수 있습니다.
- IAM 정책 및 수명 주기 정책을 비롯한 리포지토리 정책은 복제되지 않으며 정의된 리포지토리 외에는 영향을 주지 않습니다.
- 리포지토리 설정은 복제되지 않습니다. 태그 불변성, 이미지 스캔 및 KMS 암호화 설정은 복제 작업으로 인해 생성된 모든 리포지토리에서 기본적으로 비활성화됩니다. 리포지토리 생성 후 태그 불변성 및 이미지 스캔 설정을 변경할 수 있습니다. 그러나 설정이 변경된 후 푸시된 이미지에만 설정이 적용됩니다.
- 리포지토리에서 태그 불변성이 활성화되어 있고 기존 이미지와 동일한 태그를 사용하는 이미지가 복제되면 이미지가 복제되지만 중복된 태그는 포함되지 않습니다. 이로 인해 이미지에 태그가 지정되지 않을 수 있습니다.

## Amazon ECR의 프라이빗 이미지 복제 예제

다음 예제는 프라이빗 이미지 복제에 대한 일반적인 사용 사례를 보여줍니다. 를 사용하여 복제를 구성하면 JSON 파일을 생성할 때 JSON 예제를 시작점으로 사용할 수 있습니다. AWS CLI를 사용하여 복제를 구성한 경우 검토 및 제출 페이지에서 복제 규칙을 검토할 때 유사한 JSON이 표시됩니다. AWS Management Console

## 예제: 단일 대상 리전에 대한 교차 리전 복제 구성

다음은 단일 레지스트리 내에서 교차 리전 복제를 구성하는 예를 보여줍니다. 이 예제에서는 계정 ID가 111122223333이고 us-west-2 이외의 리전에서 이 복제 구성을 지정한다는 것을 가정합니다.

```
{
  "rules": [
    {
      "destinations": [
        {
          "region": "us-west-2",
          "registryId": "111122223333"
        }
      ]
    }
  ]
}
```

## 예제: 리포지토리 필터를 사용하여 교차 리전 복제 구성

다음은 접두사 이름 값과 일치하는 리포지토리에 대해 교차 리전 복제를 구성하는 예제입니다. 이 예제에서는 계정 ID가 111122223333이고 us-west-1 이외의 리전에서 이 복제 구성을 지정하고 접두사가 prod인 리포지토리가 있다는 것을 가정합니다.

```
{
  "rules": [{
    "destinations": [{
      "region": "us-west-1",
      "registryId": "111122223333"
    }],
    "repositoryFilters": [{
      "filter": "prod",
      "filterType": "PREFIX_MATCH"
    }]
  }]
}
```

## 예제: 단일 대상 리전에 대한 교차 리전 복제 구성

다음은 단일 레지스트리 내에서 교차 리전 복제를 구성하는 예를 보여줍니다. 이 예제에서는 계정 ID가 111122223333이고 us-west-1 혹은 us-west-2 이외의 리전에서 이 복제 구성을 지정한다는 것을 가정합니다.

```
{
  "rules": [
    {
      "destinations": [
        {
          "region": "us-west-1",
          "registryId": "111122223333"
        },
        {
          "region": "us-west-2",
          "registryId": "111122223333"
        }
      ]
    }
  ]
}
```

## 예제: 교차 계정 복제 구성

다음은 레지스트리에 대한 교차 계정 복제를 구성하는 예를 보여 줍니다. 이 예제에서는 444455556666 계정 및 us-west-2 리전에 대한 복제를 구성합니다.

### Important

교차 계정 복제가 발생하려면 대상 계정에서 복제를 허용하도록 레지스트리 권한 정책을 구성해야 합니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 권한](#)을 참조하세요.

```
{
  "rules": [
    {
      "destinations": [
        {
          "region": "us-west-2",
```

```

        "registryId": "444455556666"
      }
    ]
  }
}

```

## 예제: 구성에서 여러 규칙 지정

다음은 레지스트리에 대해 여러 복제 규칙을 구성하는 예제를 소개합니다. 이 예제에서는 us-west-2 리전에 대해 접두사가 prod인 리포지토리와 us-east-2 리전에 대해 접두사가 test인 리포지토리를 복제하는 하나의 규칙으로 **111122223333** 계정에 대한 복제를 구성합니다. 복제 구성에는 최대 10개의 규칙이 포함될 수 있으며 각 규칙은 최대 25개의 대상을 지정합니다.

```

{
  "rules": [{
    "destinations": [{
      "region": "us-west-2",
      "registryId": "111122223333"
    }],
    "repositoryFilters": [{
      "filter": "prod",
      "filterType": "PREFIX_MATCH"
    }]
  },
  {
    "destinations": [{
      "region": "us-east-2",
      "registryId": "111122223333"
    }],
    "repositoryFilters": [{
      "filter": "test",
      "filterType": "PREFIX_MATCH"
    }]
  }
]
}

```

## Amazon ECR에서 프라이빗 이미지 복제 구성

사실 레지스트리의 지역별 복제를 구성합니다. 지역 간 복제 또는 계정 간 복제를 구성할 수 있습니다.

복제가 일반적으로 사용되는 방법의 예는 [Amazon ECR의 프라이빗 이미지 복제 예제](#)(를) 참조하십시오.

## 레지스트리 복제 설정을 구성하려면(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 레지스트리 복제 설정을 구성할 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리(Private registry)를 선택합니다.
4. 프라이빗 레지스트리(Private registry) 페이지의 복제(Replication) 섹션에서 편집(Edit)을 선택합니다.
5. 복제(Replication) 페이지에서 복제 규칙 추가(Add replication rule)를 선택합니다.
6. 대상 유형(Destination types) 페이지에서 교차 리전 복제, 교차 계정 복제 또는 둘 다를 사용하도록 설정할지 선택한 후, 다음(Next)을 선택합니다.
7. 교차 리전 복제가 활성화된 경우 대상 리전 구성(Configure destination regions)에서 하나 이상의 대상 리전(Destination regions)을 선택한 후 다음(Next)을 선택합니다.
8. 교차 계정 복제가 활성화된 경우 교차 계정 복제(Cross-account replication)에서 레지스트리에 대한 교차 계정 복제 설정을 선택합니다. 대상 계정(Destination account)에 대상 계정의 계정 ID와 하나 이상의 복제할 대상 리전(Destination regions)을 입력합니다. 대상 계정 +(Destination account +)를 선택하여 추가 계정을 복제 대상으로 구성합니다.

### Important

교차 계정 복제가 발생하려면 대상 계정에서 복제를 허용하도록 레지스트리 권한 정책을 구성해야 합니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 권한](#)을 참조하십시오.

9. (선택 사항) 필터 추가(Add filters)페이지에서 복제 규칙에 대해 하나 이상의 필터를 지정한 다음 추가(Add)를 선택합니다. 복제 작업과 연결할 각 필터에 대해 이 단계를 반복합니다. 필터는 리포지토리 이름 접두사로 지정되어야 합니다. 필터가 추가되지 않으면 모든 리포지토리 내용이 복제됩니다. 모든 필터가 추가되면 다음(Next)을 선택합니다.
10. 검토 및 제출(Review and submit) 페이지에서 복제 규칙 구성을 검토한 다음 규칙 제출(Submit rule)을 선택합니다.

## 레지스트리 복제 설정을 구성하려면(AWS CLI)

1. 레지스트리에 대해 정의할 복제 규칙이 포함된 JSON 파일을 만듭니다. 복제 구성에는 모든 규칙에 대해 최대 25개의 고유 대상과 각 규칙당 100개의 필터가 있는 최대 10개의 규칙이 포함될 수 있습니다. 자체 계정 내에서 교차 리전 복제를 구성하려면 고유의 계정 ID를 지정합니다. 더 많은 예제는 [Amazon ECR의 프라이빗 이미지 복제 예제](#)를 참조합니다.

```
{
  "rules": [{
    "destinations": [{
      "region": "destination_region",
      "registryId": "destination_accountId"
    }],
    "repositoryFilters": [{
      "filter": "repository_prefix_name",
      "filterType": "PREFIX_MATCH"
    }]
  }]
}
```

2. 레지스트리에 대한 복제 구성을 생성합니다.

```
aws ecr put-replication-configuration \
  --replication-configuration file://replication-settings.json \
  --region us-west-2
```

3. 레지스트리 설정을 확인합니다.

```
aws ecr describe-registry \
  --region us-west-2
```

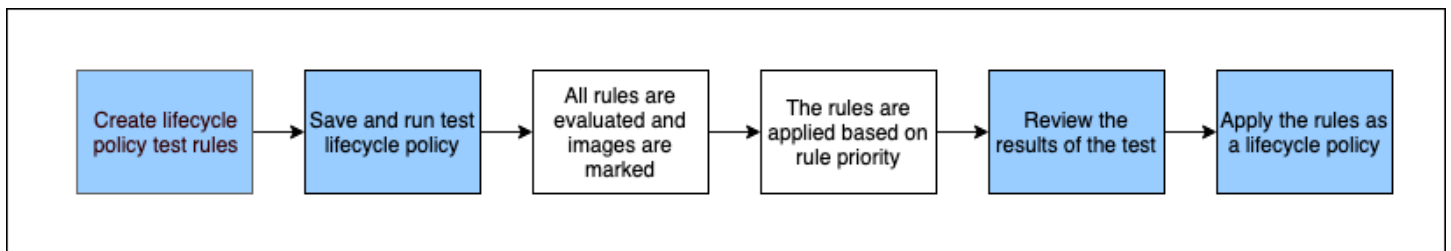
# Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.

Amazon ECR 수명 주기 정책은 프라이빗 리포지토리의 이미지에 대한 수명 주기 관리를 보다 효과적으로 제어할 수 있도록 합니다. 수명 주기 정책에는 하나 이상의 규칙이 포함되며, 각 규칙은 Amazon ECR에 대한 작업을 정의합니다. 수명 주기 정책의 만료 기준에 따라 이미지는 연령 또는 개수를 기준으로 24시간 이내에 만료됩니다. Amazon ECR이 수명 주기 정책을 기반으로 작업을 수행하는 경우 이 작업은 [에서 AWS CloudTrail이벤트로 캡처됩니다.](#) 자세한 정보는 [를 사용하여 Amazon ECR 작업 로깅 AWS CloudTrail](#)을 참조하세요.

## 수명 주기 정책의 작동 방식

수명 주기 정책은 리포지토리에서 만료되어야 하는 이미지를 결정하는 하나 이상의 규칙으로 구성됩니다. 수명 주기 정책의 사용을 고려할 때는 수명 주기 정책 미리 보기를 사용하여 수명 주기 정책을 리포지토리에 적용하기 전에 정책이 만료시키는 이미지를 확인하는 것이 중요합니다. 리포지토리에 수명 주기 정책을 적용하면, 해당 이미지는 만료 기준에 부합한 뒤 24시간 이내에 만료될 것으로 예상해야 합니다. Amazon ECR이 수명 주기 정책에 따라 작업을 수행할 때 이 작업은 AWS CloudTrail의 이벤트로 캡처됩니다. 자세한 정보는 [를 사용하여 Amazon ECR 작업 로깅 AWS CloudTrail](#)을 참조하세요.

다음 다이어그램은 수명 주기 정책 워크플로를 보여줍니다.



1. 하나 이상의 테스트 규칙을 생성합니다.
2. 테스트 규칙을 저장하고 미리 보기를 실행합니다.
3. 수명 주기 정책 평가자는 모든 규칙을 살펴보고 각 규칙이 영향을 주는 이미지를 표시합니다.
4. 그런 다음 수명 주기 정책 평가기는 규칙 우선 순위에 따라 규칙을 적용하고 리포지토리에서 만료되도록 설정된 이미지를 표시합니다.
5. 테스트 결과를 검토하여 만료될 것으로 표시된 이미지가 의도한 이미지인지 확인합니다.
6. 리포지토리에 대한 수명 주기 정책으로 테스트 규칙을 적용합니다.



7. 수명 주기 정책을 생성하면, 이미지가 만료 기준에 부합한 뒤 24시간 이내에 만료될 것으로 예상해야 합니다.

## 수명 주기 정책 평가 규칙

수명 주기 정책 평가자는 수명 주기 정책의 평문 JSON의 구문을 분석하여 모든 규칙을 평가한 다음 규칙 우선 순위에 따라 이러한 규칙을 리포지토리 내의 이미지에 적용할 책임이 있습니다. 다음은 수명 주기 정책 평가자의 논리에 대해 자세히 설명합니다. 예를 보려면 [Amazon ECR의 수명 주기 정책 예제](#)를 참조하세요.

- 규칙 우선 순위에 관계없이 모든 규칙이 동시에 평가됩니다. 모든 규칙이 평가되면 규칙 우선 순위에 따라 규칙이 적용됩니다.
- 정확히 한 개 또는 제로 규칙에 의해 이미지가 만료됩니다.
- 규칙의 태그 지정 요건에 일치하는 이미지는 이보다 우선순위가 낮은 규칙에 의해 만료되지 않습니다.
- 규칙은 우선순위가 더 높은 규칙으로 표시된 이미지에 표시할 수 없으나, 만료되지 않은 것처럼 식별할 수는 있습니다.
- 규칙 집합에는 고유한 태그 접두사 집합이 포함되어야 합니다.
- 오직 한 개의 규칙만 태그되지 않은 이미지를 선택할 수 있습니다.
- 매니페스트 목록에서 이미지를 참조하는 경우 이미지가 만료되려면 먼저 매니페스트 목록을 삭제해야 합니다.
- 만료는 항상 `pushed_at_time`이 명령하며, 항상 새 이미지보다 오래된 이미지가 먼저 만료합니다.
- 수명 주기 정책 규칙은 `tagPatternList` 또는 `tagPrefixList` 중 하나를 지정할 수 있지만 둘 다 지정할 수는 없습니다. 하지만 수명 주기 정책에는 여러 규칙이 포함될 수 있으며 다양한 규칙에서 패턴과 접두사 목록을 모두 사용합니다.
- `tagStatus0(가)` `tagged`인 경우 `tagPatternList` 또는 `tagPrefixList` 매개변수만 사용할 수 있습니다.
- `tagPatternList`을(를) 사용할 때 이미지가 와일드카드 필터와 일치하면 이미지가 성공적으로 일치하는 것입니다. 예를 들어, `prod*` 필터를 적용하면 이름이 `prod`, `prod1` 또는 `production-team1`와(과) 같은 `prod(으)`로 시작하는 리포지토리와 일치하게 됩니다. 마찬가지로 `*prod*` 필터를 적용하면 이름에 `repo-production` 또는 `prod-team`와(과) 같은 `prod0(가)` 포함된 리포지토리와 일치하게 됩니다.

**⚠ Important**

문자열당 와일드카드(\*) 는 최대 4개로 제한됩니다. 예를 들어, ["\*test\*1\*2\*3", "test\*1\*2\*3\*"]은(는) 유효하지만 ["test\*1\*2\*3\*4\*5\*6"]은(는) 유효하지 않습니다.

- tagPrefixList 사용 시 tagPrefixList 값의 모든 태그가 이미지의 태그와 일치하는 경우, 이미지가 성공적으로 일치합니다.
- countUnit 파라미터는 countType가 sinceImagePushed인 경우에만 사용됩니다.
- countType = imageCountMoreThan에서는 pushed\_at\_time을 바탕으로 새 것부터 오래된 것으로 이미지를 분류한 다음 지정 카운트보다 큰 모든 이미지가 만료됩니다.
- countType = sinceImagePushed에서는 pushed\_at\_time가 countNumber를 바탕으로 지정 날짜 수보다 오래된 모든 이미지가 만료됩니다.

## Amazon ECR에서 수명 주기 정책 미리 보기 생성

수명 주기 정책 미리 보기를 사용하면 수명 주기 정책을 적용하기 전에 이미지 저장소에 미치는 영향을 확인할 수 있습니다. 수명 주기 정책을 저장소에 적용하기 전에 미리 보기를 수행하는 것이 모범 사례입니다.

**i Note**

Amazon ECR 복제를 사용하여 여러 지역 또는 계정에 걸쳐 리포지토리를 복사하는 경우, 수명 주기 정책은 해당 정책이 생성된 리전의 리포지토리에서만 작업을 수행할 수 있다는 점에 유의하십시오. 따라서 복제가 활성화된 경우 리포지토리를 복제하려는 각 리전 및 계정에 수명 주기 정책을 생성하고자 할 수 있습니다.

### 수명 주기 정책 생성 방법(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 수명 주기 정책 미리 보기를 실행할 리포지토리가 들어 있는 리전을 선택합니다.
3. 탐색 창의 프라이빗 레지스트리(Private registry)에서 리포지토리(Repositories)를 선택합니다.
4. 프라이빗 리포지토리(Private repositories) 페이지에서 리포지토리를 선택하고 작업 드롭다운을 사용하여 수명 주기 정책(Lifecycle policies)을 선택합니다.

5. 리포지토리에 대한 수명 주기 정책 규칙 페이지에서 테스트 규칙 편집(Edit test rules), 규칙 생성(Create rule)을 선택합니다.
  6. 각 수명 주기 정책 테스트 규칙에 대한 다음 세부 정보를 지정하세요.
    - a. 규칙 우선순위(Rule priority)에 규칙 우선순위 번호를 입력합니다. 규칙 우선 순위에 따라 수명 주기 정책 규칙이 적용되는 순서가 결정됩니다.
    - b. 규칙 설명(Rule description)에 수명 주기 정책 규칙의 설명을 입력합니다.
    - c. 이미지 상태에 대해 태그 지정됨(와일드카드 일치), 태그 지정됨(접두사 일치), 태그 없음 또는 모두를 선택합니다.
    - d. 이미지 상태에 대해 태그 지정됨(와일드카드 일치)를 선택한 경우 와일드카드 일치에 대한 태그 지정에서 대해 수명 주기 정책에 따라 조치를 취할 와일드카드(\*)를 사용하여 이미지 태그 목록을 지정할 수 있습니다. 예를 들어 prod, prod1, prod2 등으로 이미지가 태그되면 prod\*을(를) 지정하여 모든 이미지에 대한 조치를 취해야 합니다. 여러 개의 태그를 지정하면, 지정된 태그가 있는 모든 이미지들만 선택됩니다.
- ⚠ Important**

문자열당 와일드카드(\*)는 최대 4개로 제한됩니다. 예를 들어, ["\*test\*1\*2\*3", "test\*1\*2\*3\*"]은(는) 유효하지만 ["test\*1\*2\*3\*4\*5\*6"]은(는) 유효하지 않습니다.
- e. 이미지 상태에 대해 태그 지정됨(접두사 일치)를 선택한 경우 접두사 일치에 대한 태그 지정에 대해 수명 주기 정책에 따라 조치를 취할 이미지 태그 목록을 지정할 수 있습니다.
  - f. 일치 기준에서 이미지가 푸시된 이후 또는 이미지 개수 초과를 선택한 다음, 값을 지정합니다.
  - g. 저장을 선택합니다.
7. 5~7단계를 반복하여 수명 주기 정책 테스트 규칙을 추가합니다.
  8. 수명 주기 정책 미리 보기를 실행하려면 테스트 저장 실행(Save and run test)을 선택합니다.
  9. 테스트 수명 주기 규칙과 일치하는 이미지(Image matches for test lifecycle rules)에서 수명 주기 정책 미리 보기의 영향을 검토합니다.
  10. 미리 보기 결과가 만족스러우면 수명 주기 정책으로 적용(Apply as lifecycle policy)을 선택하여 지정 규칙으로 수명 주기 정책을 만듭니다. 수명 주기 정책을 적용 후에는 24시간 이내에 영향을 받는 이미지가 만료됨을 예상해야 합니다.
  11. 미리 보기 결과가 만족스럽지 않으면 하나 이상의 테스트 수명 주기 규칙을 삭제하고 하나 이상의 규칙을 생성하여 대체한 다음 테스트를 반복하면 됩니다.

## Amazon ECR의 리포지토리에 대한 수명 주기 정책 생성

수명 주기 정책을 사용하여 사용하지 않은 저장소 이미지를 만료시키는 일련의 규칙을 만들 수 있습니다. 수명 주기 정책을 생성한 후 영향을 받는 이미지는 24시간 이내에 만료됩니다.

### Note

Amazon ECR 복제를 사용하여 여러 지역 또는 계정에 걸쳐 리포지토리를 복사하는 경우, 수명 주기 정책은 해당 정책이 생성된 리전의 리포지토리에서만 작업을 수행할 수 있다는 점에 유의하십시오. 따라서 복제가 활성화된 경우 리포지토리를 복제하려는 각 리전 및 계정에 수명 주기 정책을 생성하고자 할 수 있습니다.

## 전제 조건

모범 사례: 수명 주기 정책 미리 보기를 생성하여 수명 주기 정책 규칙에 따라 만료된 이미지가 의도한 대로인지 확인하십시오. 지침은 [Amazon ECR에서 수명 주기 정책 미리 보기 생성](#)을 참조하세요.

## 수명 주기 정책 생성 방법(AWS Management Console)

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/repositories>)을 엽니다.
2. 탐색 모음에서 수명 주기 정책을 생성할 리포지토리가 있는 리전을 선택합니다.
3. 탐색 창의 프라이빗 레지스트리(Private registry)에서 리포지토리(Repositories)를 선택합니다.
4. 프라이빗 리포지토리(Private repositories) 페이지에서 리포지토리를 선택하고 작업 드롭다운을 사용하여 수명 주기 정책(Lifecycle policies)을 선택합니다.
5. 리포지토리에 대한 수명 주기 정책 규칙 페이지에서 규칙 생성(Create rule)을 선택합니다.
6. 수명 주기 정책 규칙에 대한 다음 세부 정보를 입력하세요.
  - a. 규칙 우선순위(Rule priority)에 규칙 우선순위 번호를 입력합니다. 규칙 우선 순위에 따라 수명 주기 정책 규칙이 적용되는 순서가 결정됩니다.
  - b. 규칙 설명(Rule description)에 수명 주기 정책 규칙의 설명을 입력합니다.
  - c. 이미지 상태에 대해 태그 지정됨(와일드카드 일치), 태그 지정됨(접두사 일치), 태그 없음 또는 모두를 선택합니다.
  - d. 이미지 상태에 대해 태그 지정됨(와일드카드 일치)를 선택한 경우 와일드카드 일치에 대한 태그 지정에서 대해 수명 주기 정책에 따라 조치를 취할 와일드카드(\*)를 사용하여 이미지 태그 목록을 지정할 수 있습니다. 예를 들어 prod, prod1, prod2 등으로 이미지가 태그되면

prod\*을(를) 지정하여 모든 이미지에 대한 조치를 취해야 합니다. 여러 개의 태그를 지정하면, 지정된 태그가 있는 모든 이미지들만 선택됩니다.

#### **⚠ Important**

문자열당 와일드카드(\*) 는 최대 4개로 제한됩니다. 예를 들어, ["\*test\*1\*2\*3", "test\*1\*2\*3\*"]은(는) 유효하지만 ["test\*1\*2\*3\*4\*5\*6"]은(는) 유효하지 않습니다.

- e. 이미지 상태에 대해 태그 지정됨(접두사 일치)를 선택한 경우 접두사 일치에 대한 태그 지정에 대해 수명 주기 정책에 따라 조치를 취할 이미지 태그 목록을 지정할 수 있습니다.
  - f. 일치 기준에서 이미지가 푸시된 이후 또는 이미지 개수 초과를 선택한 다음, 값을 지정합니다.
  - g. 저장소를 선택합니다.
7. 5~7단계를 반복하여 수명 주기 정책 규칙을 추가합니다.

## 수명 주기 정책 생성 방법(AWS CLI)

1. 수명 주기 정책을 생성할 리포지토리의 이름을 가져옵니다.

```
aws ecr describe-repositories
```

2. 수명 주기 정책의 내용과 policy.json라는 이름의 로컬 파일을 만듭니다. 수명 주기 정책의 예제는 [Amazon ECR의 수명 주기 정책 예제](#) 단원을 참조하세요.
3. 저장소 이름을 지정하여 수명 주기 정책을 생성하고 생성한 수명 주기 정책 JSON 파일을 참조합니다.

```
aws ecr put-lifecycle-policy \
  --repository-name repository-name \
  --lifecycle-policy-text file:///policy.json
```

## Amazon ECR의 수명 주기 정책 예제

다음은 구문을 보여주는 수명 주기 정책의 예시입니다.

정책 속성에 대한 자세한 내용은 을 참조하십시오 [Amazon ECR의 라이프사이클 정책 속성](#). 를 사용하여 수명 주기 정책을 생성하는 방법에 대한 지침은 AWS CLI을 참조하십시오 [수명 주기 정책 생성 방법 \(AWS CLI\)](#).

## 수명 주기 정책 템플릿

수명 주기 정책의 내용은 저장소에 연결되기 전에 평가됩니다. 수명 주기 정책에 대한 JSON 구문 템플릿은 다음과 같습니다.

```
{
  "rules": [
    {
      "rulePriority": integer,
      "description": "string",
      "selection": {
        "tagStatus": "tagged"|"untagged"|"any",
        "tagPatternList": list<string>,
        "tagPrefixList": list<string>,
        "countType": "imageCountMoreThan"|"sinceImagePushed",
        "countUnit": "string",
        "countNumber": integer
      },
      "action": {
        "type": "expire"
      }
    }
  ]
}
```

## 이미지 수명으로 필터링

다음 예제는 14일보다 오래된 prod\*의 tagPatternList(를) 사용하여 prod(으)로 시작하는 이미지를 만료시키는 정책에 대한 수명 주기 정책 구문을 보여줍니다.

```
{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Expire images older than 14 days",
      "selection": {
        "tagStatus": "tagged",
        "tagPatternList": ["prod*"],
        "countType": "sinceImagePushed",
        "countUnit": "days",
        "countNumber": 14
      },
    },
  ],
}
```

```

        "action": {
            "type": "expire"
        }
    ]
}

```

## 이미지 수로 필터링

다음 예는 태그 없는 이미지 한 개만 유지하고 나머지는 모두 만료시키는 정책에 대한 수명 주기 정책 구문입니다.

```

{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Keep only one untagged image, expire all others",
      "selection": {
        "tagStatus": "untagged",
        "countType": "imageCountMoreThan",
        "countNumber": 1
      },
      "action": {
        "type": "expire"
      }
    }
  ]
}

```

## 복수의 규칙으로 필터링

다음은 수명 주기 정책에 복수의 규칙을 사용한 예입니다. 예로써 리포지토리와 수명 주기 정책이 결과의 설명과 함께 나타나 있습니다.

### 예 A

리포지토리 콘텐츠:

- 이미지 A, Taglist: ["beta-1", "prod-1"], 푸시: 10일 전
- 이미지 B, Taglist: ["beta-2", "prod-2"], 푸시: 9일 전

- 이미지 C, Taglist: ["beta-3"], 푸시: 8일 전

수명 주기 정책 텍스트:

```
{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Rule 1",
      "selection": {
        "tagStatus": "tagged",
        "tagPatternList": ["prod*"],
        "countType": "imageCountMoreThan",
        "countNumber": 1
      },
      "action": {
        "type": "expire"
      }
    },
    {
      "rulePriority": 2,
      "description": "Rule 2",
      "selection": {
        "tagStatus": "tagged",
        "tagPatternList": ["beta*"],
        "countType": "imageCountMoreThan",
        "countNumber": 1
      },
      "action": {
        "type": "expire"
      }
    }
  ]
}
```

이 수명 주기 정책의 논리는 다음과 같습니다.

- 규칙 1은 prod로 시작하는 태그 달린 이미지를 식별합니다. 가장 오래된 것부터 시작해서 일치하는 이미지가 한 개 이하일 때까지 이미지를 표시합니다. 이미지 A를 만료로 표시합니다.
- 규칙 2는 beta로 시작하는 태그 달린 이미지를 식별합니다. 가장 오래된 것부터 시작해서 일치하는 이미지가 한 개 이하일 때까지 이미지를 표시합니다. 이미지 A와 이미지 B를 모두 만료로 표시합니



다. 단, 이미지 A를 이미 규칙 1에서 확인했는데 이미지 B가 만료되면 규칙 1의 위반이 되므로 건너 뛩니다.

- 결과: 이미지 A가 만료되었습니다.

## 예 B

이것은 이전의 예와 동일한 리포지토리이지만, 결과를 설명하기 위해 규칙 우선 순위를 변경했습니다.

리포지토리 콘텐츠:

- 이미지 A, Taglist: ["beta-1", "prod-1"], 푸시: 10일 전
- 이미지 B, Taglist: ["beta-2", "prod-2"], 푸시: 9일 전
- 이미지 C, Taglist: ["beta-3"], 푸시: 8일 전

수명 주기 정책 텍스트:

```
{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Rule 1",
      "selection": {
        "tagStatus": "tagged",
        "tagPatternList": ["beta*"],
        "countType": "imageCountMoreThan",
        "countNumber": 1
      },
      "action": {
        "type": "expire"
      }
    },
    {
      "rulePriority": 2,
      "description": "Rule 2",
      "selection": {
        "tagStatus": "tagged",
        "tagPatternList": ["prod*"],
        "countType": "imageCountMoreThan",
        "countNumber": 1
      },
    },
  ],
}
```

```

        "action": {
            "type": "expire"
        }
    ]
}

```

이 수명 주기 정책의 논리는 다음과 같습니다.

- 규칙 1은 beta로 시작하는 태그 달린 이미지를 식별합니다. 가장 오래된 것부터 시작해서 일치하는 이미지가 한 개 이하일 때까지 이미지를 표시합니다. 세 이미지를 모두 확인하며 이미지 A와 이미지 B를 만료로 표시합니다.
- 규칙 2는 prod로 시작하는 태그 달린 이미지를 식별합니다. 가장 오래된 것부터 시작해서 일치하는 이미지가 한 개 이하일 때까지 이미지를 표시합니다. 가용 이미지는 이미 규칙 1에서 확인했기 때문에 확인할 이미지가 없으며, 따라서 추가 이미지를 표시하지 않습니다.
- 결과: 이미지 A와 이미지 B가 만료되었습니다.

## 단일 규칙에서 복수의 태그로 필터링

다음 예는 단일 규칙에서 복수의 태그 패턴에 대한 수명 주기 정책 구문을 지정합니다. 예로써 리포지토리와 수명 주기 정책이 결과의 설명과 함께 나타나 있습니다.

### 예 A

단일 규칙에 복수의 태그 패턴을 지정할 때 이미지는 목록에 있는 태그 패턴과 모두 일치해야 합니다.

리포지토리 콘텐츠:

- 이미지 A, Taglist: ["alpha-1"], 푸시: 12일 전
- 이미지 B, Taglist: ["beta-1"], 푸시: 11일 전
- 이미지 C, Taglist: ["alpha-2", "beta-2"], 푸시: 10일 전
- 이미지 D, Taglist: ["alpha-3"], 푸시: 4일 전
- 이미지 E, Taglist: ["beta-3"], 푸시: 3일 전
- 이미지 F, Taglist: ["alpha-4", "beta-4"], 푸시: 2일 전

```
{
```

```

"rules": [
  {
    "rulePriority": 1,
    "description": "Rule 1",
    "selection": {
      "tagStatus": "tagged",
      "tagPatternList": ["alpha*", "beta*"],
      "countType": "sinceImagePushed",
      "countNumber": 5,
      "countUnit": "days"
    },
    "action": {
      "type": "expire"
    }
  }
]
}

```

이 수명 주기 정책의 논리는 다음과 같습니다.

- 규칙 1은 alpha 및 beta(으)로 시작하는 태그 달린 이미지를 식별합니다. 이것으로 이미지 C와 F를 확인합니다. 5일보다 오래된 이미지, 즉 이미지 C에 표시해야 합니다.
- 결과: 이미지 C가 만료되었습니다.

## 예 B

다음 예는 함께 사용할 수 없는 태그를 보여 줍니다.

리포지토리 콘텐츠:

- 이미지 A, Taglist: ["alpha-1", "beta-1", "gamma-1"], 푸시: 10일 전
- 이미지 B, Taglist: ["alpha-2", "beta-2"], 푸시: 9일 전
- 이미지 C, Taglist: ["alpha-3", "beta-3", "gamma-2"], 푸시: 8일 전

```

{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Rule 1",
      "selection": {

```

```

        "tagStatus": "tagged",
        "tagPatternList": ["alpha*", "beta*"],
        "countType": "imageCountMoreThan",
        "countNumber": 1
    },
    "action": {
        "type": "expire"
    }
}
]
}

```

이 수명 주기 정책의 논리는 다음과 같습니다.

- 규칙 1은 alpha 및 beta(으)로 시작하는 태그 달린 이미지를 식별합니다. 이것은 모든 이미지를 확인합니다. 가장 오래된 것부터 시작해서 일치하는 이미지가 한 개 이하일 때까지 이미지를 표시합니다. 이미지 A와 B를 만료로 표시합니다.
- 결과: 이미지 A와 이미지 B가 만료되었습니다.

## 모든 이미지 필터링

다음 수명 주기 정책의 예는 여러 필터를 가진 모든 이미지를 지정합니다. 예로써 리포지토리와 수명 주기 정책이 결과의 설명과 함께 나타나 있습니다.

### 예 A

다음은 모든 규칙에 적용되지만 이미지 한 개만 유지하고 나머지는 모두 만료시키는 정책에 대한 수명 주기 정책 구문을 보여 줍니다.

리포지토리 콘텐츠:

- 이미지 A, Taglist: ["alpha-1"], 푸시: 4일 전
- 이미지 B, Taglist: ["beta-1"], 푸시: 3일 전
- 이미지 C, Taglist: [], 푸시: 2일 전
- 이미지 D, Taglist: ["alpha-2"], 푸시: 1일 전

```

{
  "rules": [

```

```

    {
      "rulePriority": 1,
      "description": "Rule 1",
      "selection": {
        "tagStatus": "any",
        "countType": "imageCountMoreThan",
        "countNumber": 1
      },
      "action": {
        "type": "expire"
      }
    }
  ]
}

```

이 수명 주기 정책의 논리는 다음과 같습니다.

- 규칙 1은 모든 이미지를 식별합니다. 이미지 A, B, C, D를 확인합니다. 가장 최신 것이 아닌 모든 이미지를 만료시켜야 합니다. 이미지 A, B, C를 만료로 표시합니다.
- 결과: 이미지 A, B, C가 만료되었습니다.

## 예 B

다음 예는 모든 규칙 유형을 하나의 단일 정책에 통합하는 수명 주기 정책을 보여 줍니다.

리포지토리 콘텐츠:

- 이미지 A, Taglist: ["alpha-", "beta-1", "-1"], 푸시: 4일 전
- Image B, Taglist: [], Pushed: 3일 전
- 이미지 C, Taglist: ["alpha-2"], 푸시: 2일 전
- Image D, Taglist: ["git hash"], Pushed: 1일 전
- 이미지 E, Taglist: [], 푸시: 1일 전

```

{
  "rules": [
    {
      "rulePriority": 1,
      "description": "Rule 1",
      "selection": {

```

```

        "tagStatus": "tagged",
        "tagPatternList": ["alpha"],
        "countType": "imageCountMoreThan",
        "countNumber": 1
    },
    "action": {
        "type": "expire"
    }
},
{
    "rulePriority": 2,
    "description": "Rule 2",
    "selection": {
        "tagStatus": "untagged",
        "countType": "sinceImagePushed",
        "countUnit": "days",
        "countNumber": 1
    },
    "action": {
        "type": "expire"
    }
},
{
    "rulePriority": 3,
    "description": "Rule 3",
    "selection": {
        "tagStatus": "any",
        "countType": "imageCountMoreThan",
        "countNumber": 1
    },
    "action": {
        "type": "expire"
    }
}
]
}

```

이 수명 주기 정책의 논리는 다음과 같습니다.

- 규칙 1은 alpha로 시작하는 태그 달린 이미지를 식별합니다. 이미지 A와 C를 식별합니다. 가장 최신 이미지를 보존하고 나머지는 만료로 표시해야 합니다. 이미지 A를 만료로 표시합니다.
- 규칙 2는 태그가 지정되지 않은 이미지를 식별합니다. 이미지 B와 E를 식별합니다. 1일이 넘은 모든 이미지를 만료로 표시해야 합니다. 이미지 B를 만료로 표시합니다.

- 규칙 3은 모든 이미지를 식별합니다. 이미지 A, B, C, D를 식별합니다. 가장 최신 이미지를 보존하고 나머지는 만료로 표시해야 합니다. 하지만 이미지 A, B, C 또는 E에는 표시할 수 없습니다. 해당 이미지들은 더 높은 우선 순위 규칙으로 식별되었기 때문입니다. 이미지D를 만료로 표시합니다.
- 결과: 이미지 A, B, C, D가 만료되었습니다.

## Amazon ECR의 라이프사이클 정책 속성

수명 주기 정책에는 다음과 같은 속성이 있습니다.

수명 주기 정책의 예를 보려면 을 참조하십시오 [Amazon ECR의 수명 주기 정책 예제](#). 를 사용하여 수명 주기 정책을 생성하는 방법에 대한 지침은 AWS CLI을 참조하십시오 [수명 주기 정책 생성 방법\(AWS CLI\)](#).

### 규칙 우선 순위

`rulePriority`

유형: 정수

필수 항목 여부: 예

규칙을 적용하는 순서를 가장 낮은 값에서 가장 높은 값까지 설정합니다. 우선 순위가 인 수명 주기 정책 규칙이 먼저 1 적용되고 우선 순위가 인 규칙이 다음에 적용되는 2 식입니다. 수명 주기 정책에 규칙을 추가할 때 각 규칙에 고유한 `rulePriority` 값을 부여해야 합니다. 정책의 여러 규칙에서 값이 순차적일 필요는 없습니다. `tagStatus` 값이 `any`인 규칙은 `rulePriority`에서 가장 높은 값을 가지며 마지막으로 평가됩니다.

### 설명

`description`

유형: string

필수 항목 여부: 아니요

(선택 사항) 수명 주기 정책에서 규칙의 목적을 설명합니다.

## 태그 상태

### tagStatus

유형: string

필수 항목 여부: 예

추가하는 수명 주기 정책의 규칙이 이미지에 대한 태그를 지정할지를 정의합니다. tagged, untagged, any 옵션을 사용할 수 있습니다. any를 지정하면 모든 이미지는 평가 규칙을 갖습니다. tagged를 지정하면 tagPrefixList 값도 지정해야 합니다. untagged를 지정하면 tagPrefixList를 생략해야 합니다.

## 태그 패턴 목록

### tagPatternList

유형: 목록[문자열]

필수 항목 여부: tagStatus이(가) 태그로 지정되어 있고 tagPrefixList은(는) 지정되지 않은 경우 예

태그가 지정된 이미지에 대한 수명 주기 정책을 생성할 때는 tagPatternList을(를) 사용하여 만료될 태그를 지정하는 것이 좋습니다. 수명 주기 방식으로 시행하려면 와일드카드(\*)를 포함할 수 있는 심표로 구분되는 이미지 태그 패턴 목록을 지정해야 합니다. 예를 들어 prod, prod1, prod2 등으로 이미지가 태그되면 태그 패턴 목록 prod\*를 써서 모든 이미지를 지정해야 합니다. 여러 개의 태그를 지정하면, 지정된 태그가 있는 모든 이미지들만 선택됩니다.

#### Important

문자열당 와일드카드(\*)는 최대 4개로 제한됩니다. 예를 들어, ["\*test\*1\*2\*3", "test\*1\*2\*3\*"]은(는) 유효하지만 ["test\*1\*2\*3\*4\*5\*6"]은(는) 유효하지 않습니다.

## 태그 접두사 목록

### tagPrefixList

유형: 목록[문자열]



필수 항목 여부: `tagStatus`이(가) 태그로 지정되어 있고 `tagPatternList`은(는) 지정되지 않은 경우 예

`"tagStatus": "tagged"`을(를) 지정하고 `tagPatternList`을(를) 지정하지 않는 경우에만 사용됩니다. 수명 주기 방식으로 시행하려면 쉼표로 구분되는 이미지 태그 접두사 목록을 지정해야 합니다. 예를 들어 `prod`, `prod1`, `prod2` 등으로 이미지가 태그되면 태그 접두사 `prod`를 써서 모든 이미지를 지정해야 합니다. 여러 개의 태그를 지정하면, 지정된 태그가 있는 모든 이미지들만 선택됩니다.

## 카운트 유형

### `countType`

유형: `string`

필수 항목 여부: 예

이미지에 적용할 카운트 유형을 지정합니다.

`countType`이 `imageCountMoreThan`으로 설정되면 `countNumber`도 지정하여 리포지토리에 존재하는 이미지 수에 제한을 정하는 규칙을 만듭니다. `countType`이 `sinceImagePushed`로 설정되면 `countUnit` 및 `countNumber`도 지정하여 리포지토리에 존재하는 이미지에 시간 제한을 지정합니다.

## 카운트 단위

### `countUnit`

유형: `string`

필수: `countType`이 `sinceImagePushed`로 설정된 경우에만 그렇습니다

시간 단위를 나타내는 `days`의 카운트 단위를 지정하고 날짜 수인 `countNumber`를 지정합니다.

이는 `countType`이 `sinceImagePushed`일 때만 지정해야 하며 `countType`이 다른 값일 때 카운트 단위를 지정하면 오류가 발생합니다.

## 카운트 수

### countNumber

유형: 정수

필수 항목 여부: 예

카운트 번호를 지정합니다. 허용되는 값은 양의 정수입니다(0은 허용되는 값이 아님).

사용한 countType이 imageCountMoreThan이라면, 값은 리포지토리에 보유하고 싶은 이미지의 최대수입니다. 사용한 countType이 sinceImagePushed라면, 값은 이미지에 대한 최대 수명 한도입니다.

## 작업

### type

유형: string

필수 항목 여부: 예

동작 유형을 지정합니다. 지원되는 값은 expire입니다.

# Amazon Elastic Container Registry의 보안

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 기업 간의 AWS 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사원은 정기적으로 [AWS 규정 준수 프로그램](#)의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon ECR에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 [AWS 규정 준수 프로그램별 범위 내 서비스](#)를 참조하세요.
- 클라우드에서의 보안 — 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀하의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Amazon ECR을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 Amazon ECR을 구성하는 방법을 보여줍니다. 또한 Amazon ECR 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

## 주제

- [Amazon Elastic Container Registry용 Identity and Access Management](#)
- [Amazon ECR에서의 데이터 보호](#)
- [Amazon Elastic 컨테이너 레지스트리에 대한 규정 준수 확인](#)
- [Amazon Elastic Container Registry의 인프라 보안](#)
- [교차 서비스 혼동된 대리인 방지](#)

## Amazon Elastic Container Registry용 Identity and Access Management

AWS Identity and Access Management (IAM)은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 있도록 AWS 서비스 있도록 도와줍니다. IAM 관리자는 어떤 사용자가 Amazon ECR 리소스를 사용할

수 있는 인증(로그인) 및 권한(권한 보유)을 받을 수 있는지를 제어합니다. IAM은 추가 AWS 서비스 비용 없이 사용할 수 있습니다.

## 주제

- [고객](#)
- [ID를 통한 인증](#)
- [정책을 사용한 액세스 관리](#)
- [Amazon Elastic Container Registry가 IAM과 작동하는 방식](#)
- [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#)
- [태그 기반 액세스 제어 사용](#)
- [AWS Amazon Elastic 컨테이너 레지스트리에 대한 관리형 정책](#)
- [Amazon ECR에 대한 서비스 연결 역할 사용](#)
- [Amazon Elastic Container Registry Identity and Access 문제 해결](#)

## 고객

Amazon ECR에서 수행하는 작업에 따라 사용 방법 AWS Identity and Access Management (IAM) 이 다릅니다.

서비스 사용자 - Amazon ECR 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한은 관리자가 제공합니다. 더 많은 Amazon ECR 기능을 사용하여 작업을 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon ECR의 기능에 액세스할 수 없다면 [Amazon Elastic Container Registry Identity and Access 문제 해결](#)(을) 참조하세요.

서비스 관리자 - 회사에서 Amazon ECR 리소스를 책임지고 있다면 Amazon ECR에 대한 모든 액세스 권한이 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon ECR 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 Amazon ECR에서 IAM을 사용할 수 있는 방법에 대해 자세히 알아보려면 [Amazon Elastic Container Registry가 IAM과 작동하는 방식](#)(을) 참조하세요.

IAM 관리자 - IAM 관리자라면 Amazon ECR에 대한 액세스 권한 관리를 위한 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 Amazon ECR 자격 증명 기반 정책 예제를 보려면 [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#) 섹션을 참조하세요.

## ID를 통한 인증

인증은 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 연동 자격 증명으로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 [내 로그인 방법을](#) 참조하십시오. AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK)와 명령줄 인터페이스 (CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS [API 요청 서명을](#) 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA)을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [다중 인증](#) 및 IAM 사용 설명서의 [AWS에서 다중 인증\(MFA\) 사용](#)을 참조하세요.

### AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 [루트 사용자 보안 인증이 필요한 태스크](#)를 참조하세요.

### IAM 사용자 및 그룹

[IAM 사용자는 단일 사용자](#) 또는 애플리케이션에 대한 특정 권한을 가진 사용자 내의 자격 증명입니다. AWS 계정 가능하면 암호 및 액세스 키와 같은 장기 자격 증명에 있는 IAM 사용자를 생성하는 대신 임시 자격 증명에 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명에 필요한 특정 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 [장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체](#)를 참조하세요.

**IAM 그룹**은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 [IAM 사용자를 만들어야 하는 경우\(역할이 아님\)](#)를 참조하세요.

## IAM 역할

**IAM 역할**은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console [전환하여](#) 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 [IAM 역할 사용](#)을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 연동 자격 증명에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 [타사 자격 증명 공급자의 역할 만들기](#)를 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관 짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [권한 세트](#)를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스 경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 [IAM 역할과 리소스 기반 정책의 차이](#)를 참조하세요.
- 서비스 간 액세스 — 일부는 다른 AWS 서비스서비스의 기능을 AWS 서비스 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.

- 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 [전달 액세스 세션](#)을 참조하세요.
- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 [IAM 역할](#)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [AWS 서비스에 대한 권한을 위임할 역할 생성](#)을 참조하세요.
- 서비스 연결 역할 — 서비스 연결 역할은 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수입할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여](#)를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 [IAM 사용 설명서](#)의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

## 정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 [JSON 정책 개요](#)를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수입할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, iam:GetRole태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

## ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [IAM 정책 생성](#)을 참조하세요.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 [관리형 정책과 인라인 정책의 선택](#)을 참조하세요.

## 리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 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 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 [보안 주체를 지정](#)해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

## 기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 보안 인증 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결



과로 얻는 권한은 엔터티의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. Principal 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 [IAM 엔터티에 대한 권한 경계](#)를 참조하세요.

- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU)에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책 (SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함)에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 [SCP 작동 방식](#)을 참조하세요.
- 세션 정책 - 세션 정책은 역할 또는 페더레이션 사용자에게 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할 자격 증명 기반 정책의 교차 및 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 [세션 정책](#)을 참조하세요.

## 여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련된 경우 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 [정책 평가 로직](#)을 참조하십시오.

## Amazon Elastic Container Registry가 IAM과 작동하는 방식

IAM을 사용하여 Amazon ECR에 대한 액세스를 관리하기 전에 Amazon ECR에서 사용할 수 있는 IAM 기능을 이해해야 합니다. Amazon ECR 및 기타 AWS 서비스가 IAM과 어떻게 연동되는지 자세히 알아보려면 IAM 사용 설명서의 [IAM과 연동되는AWS 서비스를](#) 참조하십시오.

### 주제

- [Amazon ECR 자격 증명 기반 정책](#)
- [Amazon ECR 리소스 기반 정책](#)
- [Amazon ECR 태그 기반 권한 부여](#)
- [Amazon ECR IAM 역할](#)

## Amazon ECR 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. Amazon ECR은 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 [IAM JSON 정책 요소 참조](#)를 참조하세요.

### 작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 Action요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 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 포함하십시오.

Amazon ECR의 정책 작업은 작업 앞에 다음 접두사를 사용합니다 `ecr:`. 예를 들어 Amazon ECR `CreateRepository` API 작업으로 Amazon ECR 리포지토리를 생성할 수 있는 권한을 누군가에게 부여하려면 해당 정책에 `ecr:CreateRepository` 작업을 포함합니다. 정책 문에는 Action 또는 NotAction 요소가 포함되어야 합니다. Amazon ECR은 이 서비스로 수행할 수 있는 태스크를 설명하는 고유한 작업 집합을 정의합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
  "ecr:action1",
  "ecr:action2"
```

와일드카드(\*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "ecr:Describe*"
```

Amazon ECR 작업 목록을 보려면 IAM 사용 설명서의 [Amazon Elastic Container Registry에 사용되는 작업, 리소스 및 조건 키](#)를 참조하십시오.

## 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 있는지 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 Resource 또는 NotResource 요소가 반드시 추가되어야 합니다. 모범 사례에 따라 [Amazon 리소스 이름\(ARN\)](#)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(\*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"

```

Amazon ECR 리포지토리 리소스에는 다음의 ARN이 있습니다.

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-name}

```

ARN 형식에 대한 자세한 내용은 [Amazon 리소스 이름 \(ARN\) 및 AWS 서비스 네임스페이스](#)를 참조하십시오.

예를 들어 문에서 us-east-1 리전의 my-repo 리포지토리를 지정하려면 다음 ARN을 사용하십시오.

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"

```

특정 계정에 속한 모든 리포지토리를 지정하려면 와일드카드(\*)를 사용합니다.

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"

```

단일 문에서 여러 리소스를 지정하려면 ARN을 쉼표로 구분합니다.

```
"Resource": [
  "resource1",
  "resource2"
]

```

Amazon ECR 리소스 유형 및 해당 ARN의 목록을 보려면 IAM 사용 설명서의 [Amazon Elastic Container Registry에서 정의한 리소스](#)를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 [Amazon Elastic Container Registry에서 정의한 작업](#)을 참조하세요.

## 조건 키

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 옵션입니다. 같거나 작음과 같은 [조건 연산자](#)를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 Condition요소를 지정하거나 단일 Condition요소에서 여러 키를 지정하는 경우 AWS 는 논리적 AND태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 OR 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [IAM 정책 요소: 변수 및 태그](#)를 참조하세요.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 [AWS 설명서의 글로벌 조건 컨텍스트 키](#)를 참조하십시오.

Amazon ECR은 자체 조건 키 집합을 정의하며 일부 전역 조건 키 사용도 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 설명서의 [AWS 글로벌 조건 컨텍스트 키](#)를 참조하십시오.

대부분의 Amazon ECR 작업에서는 `aws:ResourceTag` 및 `ecr:ResourceTag` 조건 키를 지원합니다. 자세한 정보는 [태그 기반 액세스 제어 사용](#)을 참조하세요.

Amazon ECR 조건 키 목록은 IAM 사용 설명서의 [Amazon Elastic Container Registry에서 정의한 조건 키](#)를 참조하십시오. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 [Amazon Elastic Container Registry에서 정의한 작업](#)을 참조하세요.

## 예제

Amazon ECR 자격 증명 기반 정책 예제를 보려면 [Amazon Elastic Container Registry 자격 증명 기반 정책 예제](#)(를) 참조하세요.

## Amazon ECR 리소스 기반 정책

리소스 기반 정책은 지정된 주체가 Amazon ECR 리소스에 대해 수행할 수 있는 작업 및 관련 조건을 지정하는 JSON 정책 문서입니다. Amazon ECR은 Amazon ECR 리포지토리에 대한 리소스 기반 권한

정책을 지원합니다. 리소스 기반 정책을 사용하여 리소스별로 다른 계정에 사용 권한을 부여할 수 있습니다. 또한 리소스 기반 정책을 사용하여 AWS 서비스가 Amazon ECR 리포지토리에 액세스하도록 허용할 수 있습니다.

크로스 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 엔티티를 [리소스 기반 정책의 보안 주체](#)로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 서로 다른 AWS 계정에 있는 경우 보안 주체 엔티티에 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔티티에 보안 인증 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용자 설명서의 [IAM 역할과 리소스 기반 정책의 차이](#)를 참조합니다.

Amazon ECR 서비스는 리소스 기반 정책 중 한 가지 유형만 지원하는데, 이 정책 유형은 리포지토리 정책이라고 하며 리포지토리에 연결되어 있습니다. 이 정책은 리포지토리에서 작업을 수행할 수 있는 주체 엔티티(계정, 사용자, 역할 및 페더레이션 사용자)를 정의합니다. 리포지토리에 리소스 기반 정책을 연결하는 방법은 [Amazon ECR의 프라이빗 리포지토리 정책](#) 단원을 참조하십시오.

#### Note

Amazon ECR 리포지토리 정책에서 정책 요소 Sid는 IAM 정책에서 지원되지 않는 추가 문자 및 공백을 지원합니다.

## 예제

Amazon ECR 리소스 기반 정책의 예를 보려면 [Amazon ECR의 프라이빗 리포지토리 정책 예제](#)(를) 참조하십시오,

## Amazon ECR 태그 기반 권한 부여

Amazon ECR 리소스에 태그를 연결하거나 Amazon ECR에 대한 요청에서 태그를 전달할 수 있습니다. 태그를 기반으로 액세스를 제어하려면 `ecr:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 [조건 요소](#)에 태그 정보를 제공합니다. Amazon ECR 리소스 태깅에 대한 자세한 내용은 [Amazon ECR의 프라이빗 리포지토리에 태그 지정](#)(를) 참조하세요.

리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예제는 [태그 기반 액세스 제어 사용](#) 섹션에서 확인할 수 있습니다.

## Amazon ECR IAM 역할

[IAM 역할](#)은 AWS 계정 내에서 특정 권한을 가진 엔티티입니다.

### Amazon ECR에서 임시 자격 증명 사용

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. [AssumeRole](#) 또는 [GetFederation토큰과](#) 같은 AWS STS API 작업을 호출하여 임시 보안 자격 증명을 얻습니다.

Amazon ECR은 임시 자격 증명 사용을 지원합니다.

### 서비스 연결 역할

[서비스 연결 역할](#)을 사용하면 AWS 서비스가 다른 서비스의 리소스에 액세스하여 사용자를 대신하여 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수 없습니다.

Amazon ECR은 서비스 연결 역할을 지원합니다. 자세한 정보는 [Amazon ECR에 대한 서비스 연결 역할 사용](#)을 참조하세요.

## Amazon Elastic Container Registry 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할은 Amazon ECR 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [IAM 정책 생성](#)을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 비롯하여 Amazon ECR에서 정의되는 작업 및 리소스 유형에 대한 자세한 내용은 [서비스 승인 참조](#)의 Amazon Elastic Container Registry에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [JSON 탭에서 정책 생성](#)을 참조하세요.

### 주제

- [정책 모범 사례](#)

- [Amazon ECR 콘솔 사용](#)
- [사용자가 자신이 권한을 볼 수 있도록 허용](#)
- [하나의 Amazon ECR 리포지토리에 액세스](#)

## 정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon ECR 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르십시오.

- AWS 관리형 정책으로 시작하고 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하세요. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 [AWS 관리형 정책](#) 또는 [AWS 직무에 대한 관리형 정책](#)을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 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 권한](#)을 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. 예를 들어 AWS 서비스들에서 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용 설명서의 [IAM JSON 정책 요소: 조건](#)을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 [IAM Access Analyzer 정책 검증](#)을 참조하세요.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접 호출할 때 MFA가 필요하다면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 [MFA 보호 API 액세스 구성](#)을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 [IAM의 보안 모범 사례](#)를 참조하세요.

## Amazon ECR 콘솔 사용

Amazon Elastic Container Registry에 액세스하려면 최소한의 권한 집합이 있어야 합니다. 이러한 권한을 통해 AWS 계정의 Amazon ECR 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다. 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

이러한 엔티티가 Amazon ECR 콘솔을 계속 사용할 수 있도록 하려면 엔티티에 AmazonEC2ContainerRegistryReadOnly AWS 관리형 정책을 추가하십시오. 자세한 내용은 IAM 사용 설명서의 [사용자에게 권한 추가](#)를 참조하십시오.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "ecr:DescribeImages",
        "ecr:BatchGetImage",
        "ecr:GetLifecyclePolicy",
        "ecr:GetLifecyclePolicyPreview",
        "ecr:ListTagsForResource",
        "ecr:DescribeImageScanFindings"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.



## 사용자가 자신이 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

## 하나의 Amazon ECR 리포지토리에 액세스

이 예시에서는 AWS 계정의 사용자에게 Amazon ECR 리포지토리 중 하나에 대한 액세스 권한을 부여하려고 합니다. `my-repo` 또한 사용자가 이미지를 푸시 및 풀하고 나열할 수 있게 하려고 합니다.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"ListImagesInRepository",
      "Effect":"Allow",
      "Action":[
        "ecr:ListImages"
      ],
      "Resource":"arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
    },
    {
      "Sid":"GetAuthorizationToken",
      "Effect":"Allow",
      "Action":[
        "ecr:GetAuthorizationToken"
      ],
      "Resource":""
    },
    {
      "Sid":"ManageRepositoryContents",
      "Effect":"Allow",
      "Action":[
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "ecr:DescribeImages",
        "ecr:BatchGetImage",
        "ecr:InitiateLayerUpload",
        "ecr:UploadLayerPart",
        "ecr:CompleteLayerUpload",
        "ecr:PutImage"
      ],
      "Resource":"arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
    }
  ]
}
```

}

## 태그 기반 액세스 제어 사용

Amazon ECR CreateRepository API 작업을 사용하면 리포지토리를 생성할 때 태그를 지정할 수 있습니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리에 태그 지정](#)을 참조하세요.

사용자가 생성 시 리포지토리에 태그를 지정할 수 있으려면 리소스를 생성하는 작업을 사용할 권한이 있어야 합니다(예: ecr:CreateRepository). 리소스 생성 작업에서 태그가 지정되면 Amazon은 ecr:CreateRepository 작업에서 추가 권한 부여를 수행해 사용자에게 태그를 생성할 권한이 있는지 확인합니다.

IAM 정책을 통해 태그 기반 액세스 제어를 사용할 수 있습니다. 예를 들면 다음과 같습니다.

다음 정책에서는 사용자가 리포지토리를 생성하거나 key=environment, value=dev로 태그를 지정하도록 허용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateTaggedRepository",
      "Effect": "Allow",
      "Action": [
        "ecr:CreateRepository"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/environment": "dev"
        }
      }
    },
    {
      "Sid": "AllowTagRepository",
      "Effect": "Allow",
      "Action": [
        "ecr:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/environment": "dev"
        }
      }
    }
  ]
}
```

```

    }
  }
}
]
}

```

다음 정책에서는 사용자가 `key=environment,value=prod`로 태그를 지정하지 않은 모든 리포지토리에 액세스하도록 허용합니다.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ecr:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "ecr:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ecr:ResourceTag/environment": "prod"
        }
      }
    }
  ]
}

```

## AWS Amazon Elastic 컨테이너 레지스트리에 대한 관리형 정책

AWS 관리형 정책은 에서 생성하고 관리하는 독립 실행형 정책입니다. AWS AWS 관리형 정책은 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었으므로 사용자, 그룹 및 역할에 권한을 할당하기 시작할 수 있습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한 권한을 부여하지 않을 수도 있다는 점에 유의하세요. 사용 사례에 고유한 [고객 관리형 정책](#)을 정의하여 권한을 줄이는 것이 좋습니다.

관리형 정책에 정의된 권한은 변경할 수 없습니다. AWS AWS 관리형 정책에 정의된 권한을 업데이트 하는 경우 AWS 해당 업데이트는 정책이 연결된 모든 주체 ID (사용자, 그룹, 역할) 에 영향을 미칩니다.

AWS 새 API 작업이 시작되거나 기존 서비스에 새 AWS 서비스 API 작업을 사용할 수 있게 되면 AWS 관리형 정책을 업데이트할 가능성이 가장 높습니다.

자세한 내용은 IAM 사용자 설명서의 [AWS 관리형 정책](#)을 참조하세요.

Amazon ECR은 IAM ID 또는 Amazon EC2 인스턴스에 연결할 수 있는 여러 관리형 정책을 제공합니다. 이러한 관리형 정책을 통해 Amazon ECR 리소스 및 API 작업에 대한 액세스를 다양한 수준으로 제어할 수 있습니다. 이러한 정책에 언급되는 각 API 작업에 대한 자세한 내용은 Amazon Elastic Container Registry API 참조의 [작업](#) 단원을 참조하십시오.

## 주제

- [AmazonEC2ContainerRegistryFullAccess](#)
- [AmazonEC2ContainerRegistryPowerUser](#)
- [AmazonEC2ContainerRegistryReadOnly](#)
- [AWSECRPullThroughCache\\_ServiceRolePolicy](#)
- [ECRReplicationServiceRolePolicy](#)
- [AWS 관리형 정책에 대한 Amazon ECR 업데이트](#)

## AmazonEC2ContainerRegistryFullAccess

AmazonEC2ContainerRegistryFullAccess 정책을 IAM 보안 인증에 연결할 수 있습니다.

이 관리형 정책을 시작점으로 사용하여 특정 요구 사항에 따라 고유한 IAM 정책을 생성할 수 있습니다. 예를 들어 사용자나 역할에 Amazon ECR 사용을 관리할 전체 관리자 액세스 권한을 제공하는 정책을 생성할 수 있습니다. [Amazon ECR 수명 주기 정책](#) 기능을 사용하여 리포지토리에서 이미지의 수명 주기 관리를 지정할 수 있습니다. 수명 주기 정책 이벤트는 이벤트로 CloudTrail 보고됩니다. Amazon ECR은 과 통합되어 AWS CloudTrail 있어 Amazon ECR 콘솔에 수명 주기 정책 이벤트를 직접 표시할 수 있습니다. AmazonEC2ContainerRegistryFullAccess 관리형 IAM 정책에는 이 동작을 촉진할 수 있는 `cloudtrail:LookupEvents` 권한이 포함되어 있습니다.

## 권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `ecr` – 보안 주체가 모든 Amazon ECR API에 대한 모든 권한을 허용합니다.
- `cloudtrail`— 주체가 에서 캡처한 관리 이벤트 또는 AWS CloudTrail Insights 이벤트를 조회할 수 있습니다. CloudTrail

이 AmazonEC2ContainerRegistryFullAccess 정책은 다음과 같습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:*",
        "cloudtrail:LookupEvents"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": [
            "replication.ecr.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

## AmazonEC2ContainerRegistryPowerUser

AmazonEC2ContainerRegistryPowerUser 정책을 IAM 보안 인증에 연결할 수 있습니다.

이 정책은 IAM 사용자가 리포지토리에 대한 읽기 및 쓰기를 허용하는 관리 권한을 부여하지만, 리포지토리를 삭제하거나 리포지토리에 적용된 정책 설명을 변경할 수는 없습니다.

권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `ecr` – 보안 주체가 리포지토리에 읽기 및 쓰기 그리고 수명 주기 정책 읽기를 허용합니다. 보안 주체에게는 리포지토리를 삭제하거나 저장소에 적용되는 수명 주기 정책을 변경할 수 있는 권한이 부여되지 않습니다.

이 `AmazonEC2ContainerRegistryPowerUser` 정책은 다음과 같습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "ecr:DescribeImages",
        "ecr:BatchGetImage",
        "ecr:GetLifecyclePolicy",
        "ecr:GetLifecyclePolicyPreview",
        "ecr:ListTagsForResource",
        "ecr:DescribeImageScanFindings",
        "ecr:InitiateLayerUpload",
        "ecr:UploadLayerPart",
        "ecr:CompleteLayerUpload",
        "ecr:PutImage"
      ],
      "Resource": "*"
    }
  ]
}
```

## AmazonEC2ContainerRegistryReadOnly

`AmazonEC2ContainerRegistryReadOnly` 정책을 IAM 보안 인증에 연결할 수 있습니다.

이 정책은 Amazon ECR에 대한 읽기 전용 권한을 부여합니다. 여기에는 리포지토리와 리포지토리 내의 이미지를 나열하는 기능이 포함됩니다. 또한 Docker CLI를 사용하여 Amazon ECR에서 이미지를 가져올 수 있는 기능도 포함되어 있습니다.

## 권한 세부 정보

이 정책에는 다음 권한이 포함되어 있습니다.

- `ecr` – 보안 주체가 리포지토리 및 해당 수명 주기 정책을 읽을 수 있도록 합니다.

이 `AmazonEC2ContainerRegistryReadOnly` 정책은 다음과 같습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "ecr:DescribeImages",
        "ecr:BatchGetImage",
        "ecr:GetLifecyclePolicy",
        "ecr:GetLifecyclePolicyPreview",
        "ecr:ListTagsForResource",
        "ecr:DescribeImageScanFindings"
      ],
      "Resource": "*"
    }
  ]
}
```

## AWSECRPullThroughCache\_ServiceRolePolicy

`AWSECRPullThroughCache_ServiceRolePolicy` 관리형 IAM 정책을 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Amazon ECR이 풀스루 캐시 워크플로를 통해 이미지를 리포지토리에 푸시할 수 있도록 하는 서비스 연결 역할에 연결됩니다. 자세한 정보는 [풀스루 캐시에 대한 Amazon ECR 서비스 연결 역할](#)을 참조하세요.



## ECRReplicationServiceRolePolicy

ECRReplicationServiceRolePolicy 관리형 IAM 정책을 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Amazon ECR에 사용자를 대신하여 작업을 수행할 수 있도록 하는 서비스 연결 역할에 연결됩니다. 자세한 정보는 [Amazon ECR에 대한 서비스 연결 역할 사용](#)을 참조하세요.

### AWS 관리형 정책에 대한 Amazon ECR 업데이트

이 서비스가 변경 사항을 추적하기 시작한 이후 Amazon ECR의 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인하십시오. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon ECR 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
<a href="#">AWSECRPullThroughCache_ServiceRolePolicy</a> - 기존 정책에 대한 업데이트	Amazon ECR에서 AWSECRPullThroughCache_ServiceRolePolicy 정책에 대한 새로운 권한을 추가했습니다. 이러한 권한을 통해 Amazon ECR은 Secrets Manager 보안 암호의 암호화된 콘텐츠를 검색할 수 있습니다. 이는 폴스루 캐시 규칙을 사용하여 인증이 필요한 업스트림 레지스트리에서 이미지를 캐시할 때 필요합니다.	2023년 11월 15일
<a href="#">AWSECRPullThroughCache_ServiceRolePolicy</a> - 새 정책	Amazon ECR에서 새 정책을 추가했습니다. 이 정책은 폴스루 캐시 기능에 대한 AWSServiceRoleForECRPullThroughCache 서비스 연결 역할과 연결됩니다.	2021년 11월 29일
<a href="#">ECR ReplicationServiceRolePolicy</a> — 새 정책	Amazon ECR에서 새 정책을 추가했습니다. 이 정책은 복	2020년 12월 4일

변경 사항	설명	날짜
	제 기능에 대한 AWSServiceRoleForECRReplication 서비스 연결 역할과 연결됩니다.	
<a href="#">AmazonEC2 Container Registry FullAccess</a> — 기존 정책에 대한 업데이트	Amazon ECR에서 AmazonEC2ContainerRegistryFullAccess 정책에 대한 새로운 권한을 추가했습니다. 이러한 권한을 사용하여 보안 주체는 Amazon ECR 서비스 연결 역할을 생성할 수 있습니다.	2020년 12월 4일
<a href="#">AmazonEC2 Container Registry ReadOnly</a> — 기존 정책에 대한 업데이트	Amazon ECR에서 보안 주체가 수명 주기 정책을 읽고, 태그를 나열하고, 이미지에 대한 검색 결과를 설명할 수 있는 새로운 권한을 AmazonEC2ContainerRegistryReadOnly 정책에 추가했습니다.	2019년 12월 10일
<a href="#">AmazonEC2 Container Registry PowerUser</a> — 기존 정책에 대한 업데이트	Amazon ECR에서 AmazonEC2ContainerRegistryPowerUser 정책에 대한 새로운 권한을 추가했습니다. 이는 보안 주체가 수명 주기 정책을 읽고, 태그를 나열하고, 이미지에 대한 검색 결과를 설명할 수 있도록 허용합니다.	2019년 12월 10일

변경 사항	설명	날짜
<a href="#">AmazonEC2 Container Registry FullAccess</a> — 기존 정책에 대한 업데이트	Amazon ECR에서 AmazonEC2ContainerRegistryFullAccess 정책에 대한 새로운 권한을 추가했습니다. 이를 통해 보안 주체는 에서 캡처한 관리 이벤트 또는 AWS CloudTrail Insights 이벤트를 조회할 수 있습니다. CloudTrail	2017년 11월 10일
<a href="#">AmazonEC2 Container Registry ReadOnly</a> — 기존 정책에 대한 업데이트	Amazon ECR에서 AmazonEC2ContainerRegistryReadOnly 정책에 대한 새로운 권한을 추가했습니다. 이는 보안 주체가 Amazon ECR 이미지를 설명할 수 있도록 허용합니다.	2016년 10월 11일
<a href="#">AmazonEC2 Container Registry PowerUser</a> — 기존 정책에 대한 업데이트	Amazon ECR에서 AmazonEC2ContainerRegistryPowerUser 정책에 대한 새로운 권한을 추가했습니다. 이는 보안 주체가 Amazon ECR 이미지를 설명할 수 있도록 허용합니다.	2016년 10월 11일

변경 사항	설명	날짜
<a href="#">아마존 EC2 — 새 정책 ContainerRegistry ReadOnly</a>	Amazon ECR은 Amazon ECR에 읽기 전용 권한을 부여하는 새로운 정책을 추가했습니다. 이러한 권한에는 리포지토리 및 리포지토리 내의 이미지를 나열하는 기능이 포함됩니다. 또한 Docker CLI를 사용하여 Amazon ECR에서 이미지를 가져올 수 있는 기능도 포함되어 있습니다.	2015년 12월 21일
<a href="#">아마존 EC2 — 새 정책 ContainerRegistry PowerUser</a>	Amazon ECR에서 사용자가 리포지토리에 대한 읽기 및 쓰기는 허용하지만, 리포지토리를 삭제하거나 리포지토리에 적용된 정책 문서를 변경할 수는 없는 관리 권한을 부여하는 새 정책을 추가했습니다.	2015년 12월 21일
<a href="#">아마존 EC2 — 새 정책 ContainerRegistry FullAccess</a>	Amazon ECR에서 새 정책을 추가했습니다. 이 정책은 Amazon ECR에 대한 모든 액세스 권한을 부여합니다.	2015년 12월 21일
Amazon ECR, 변경 사항 추적 시작	Amazon ECR은 AWS 관리형 정책의 변경 사항을 추적하기 시작했습니다.	2021년 6월 24일

## Amazon ECR에 대한 서비스 연결 역할 사용

Amazon Elastic Container Registry (Amazon ECR) AWS Identity and Access Management 는 (IAM) [서비스 연결 역할을 사용하여 복제 및 풀 스루](#) 캐시 기능을 사용하는 데 필요한 권한을 제공합니다. 서비스 연결 역할은 Amazon ECR에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon ECR에서 미리 정의합니다. 여기에는 프라이빗 레지스트리에 대한 복제 및 풀 스루 캐시 기능을 지원하기 위해 서비스에서 필요로 하는 모든 권한이 포함됩니다. 레지스트리에 대한 복제 또는 풀 스루

루 캐시를 구성하고 나면 서비스 연결 역할이 사용자를 대신하여 자동으로 만들어집니다. 자세한 정보는 [Amazon ECR의 프라이빗 레지스트리 설정](#)을 참조하세요.

서비스 연결 역할을 사용하면 Amazon ECR을 통한 복제 및 풀스루 캐시를 더 쉽게 설정할 수 있습니다. 이를 사용하면 필요한 권한을 모두 수동으로 추가할 필요가 없기 때문입니다. Amazon ECR에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon ECR만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함됩니다. 권한 정책은 다른 어떤 IAM 엔터티에도 연결할 수 없습니다.

레지스트리에서 풀스루 캐시 또는 복제를 사용 중지한 후에만 해당 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 Amazon ECR에서 이러한 기능에 필요한 권한을 실수로 제거하지 않도록 할 수 있습니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 자세한 내용은 [IAM으로 작업하는AWS 서비스](#)를 참조하세요. 이 연결 페이지에서 서비스 연결 역할 옆에 예라고 표시된 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

## 주제

- [Amazon ECR 서비스 연결 역할이 지원되는 리전](#)
- [복제에 대한 Amazon ECR 서비스 연결 역할](#)
- [풀스루 캐시에 대한 Amazon ECR 서비스 연결 역할](#)

## Amazon ECR 서비스 연결 역할이 지원되는 리전

Amazon ECR은 Amazon ECR 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. Amazon ECR 리전 가용성에 대한 자세한 내용은 [AWS 리전 및 엔드포인트](#)를 참조하세요.

## 복제에 대한 Amazon ECR 서비스 연결 역할

Amazon ECR은 Amazon ECR이 여러 계정 간에 이미지를 AWSServiceRoleForECRReplication복제할 수 있도록 하는 이름이 지정된 서비스 연결 역할을 사용합니다.

## Amazon ECR에 대한 서비스 연결 역할 권한

AWSServiceRoleForECRReplication 서비스 연결 역할은 다음 서비스가 역할을 맡을 것으로 신뢰합니다.

- replication.ecr.amazonaws.com

다음 ECRReplicationServiceRolePolicy 역할 권한 정책은 Amazon ECR가 리소스에서 다음 작업을 사용하도록 허용합니다.

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

#### Note

이 ReplicateImage는 Amazon ECR이 복제에 사용하는 내부 API이며 직접 호출할 수 없습니다.

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성해야 합니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 권한](#) 섹션을 참조하세요.

#### Amazon ECR에 대한 서비스 연결 역할 생성

Amazon ECR 서비스 연결 역할을 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI, 또는 AWS API에서 레지스트리의 복제 설정을 구성하면 Amazon ECR에서 서비스 연결 역할을 자동으로 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 레지스트리에 대한 복제 설정을 구성하는 경우, Amazon ECR에서 서비스 연결 역할을 다시 생성합니다.

#### Amazon ECR에 대한 서비스 연결 역할 편집

Amazon ECR에서는 AWSServiceRoleForECRReplication 서비스 연결 역할을 수동으로 편집할 수 없습니다. 서비스 연결 역할을 생성한 후에는 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름

을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 편집](#)을 참조하세요.

## Amazon ECR에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 이렇게 하면 활성적으로 모니터링하거나 유지 관리하지 않는 미사용 개체가 없게 됩니다. 그러나 서비스 연결 역할을 수동으로 삭제하려면 먼저 모든 리전에서 레지스트리에 대한 복제 구성을 제거해야 합니다.

### Note

Amazon ECR 서비스에서 계속 역할을 사용하고 있는 동안 리소스를 삭제하려고 하면 삭제 작업이 실패할 수 있습니다. 그 경우 몇 분 동안 기다렸다가 다시 시도하십시오.

에서 사용하는 Amazon ECR 리소스를 삭제하려면 `AWSServiceRoleForECRReplication`

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 복제 구성이 설정된 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리를 선택합니다.
4. 프라이빗 레지스트리(Private registry) 페이지의 복제 구성(Replication configuration) 섹션에서 편집(Edit)을 선택합니다.
5. 모든 복제 규칙을 삭제하려면 모두 삭제>Delete all)를 선택합니다. 이 단계를 사용하려면 확인이 필요합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 `AWSServiceRoleForECRReplication` 서비스 연결 역할을 삭제합니다. 자세한 내용은 [IAM 사용 설명서](#)의 서비스 연결 역할 삭제를 참조하세요.

## 풀스루 캐시에 대한 Amazon ECR 서비스 연결 역할

Amazon ECR은 이름이 지정된 서비스 연결 역할을 사용하여 Amazon ECR이 사용자 대신 작업을 수행하여 풀 스루 캐시 작업을 완료할 수 `AWSServiceRoleForECRPullThroughCache` 있는 권한을 부여합니다. 풀 스루 캐시에 대한 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화](#)를 참조하세요.

## Amazon ECR에 대한 서비스 연결 역할 권한

AWSServiceRoleForECRPullThroughCache 서비스 연결 역할은 다음 서비스가 역할을 맡을 것으로 신뢰합니다.

- `pullthroughcache.ecr.amazonaws.com`

### 권한 세부 정보

이 `AWSECRPullThroughCache_ServiceRolePolicy` 권한 정책은 서비스 역할에 연결됩니다. 이 관리형 정책은 Amazon ECR에 다음 작업을 수행할 수 있는 권한을 부여합니다. 자세한 정보는 [AWSECRPullThroughCache\\_ServiceRolePolicy](#)을 참조하세요.

- `ecr` - Amazon ECR 서비스에서 이미지를 프라이빗 리포지토리로 푸시할 수 있도록 허용합니다.
- `secretsmanager:GetSecretValue`— Amazon ECR 서비스가 암호화된 AWS Secrets Manager 보안 콘텐츠를 검색할 수 있도록 허용합니다. 이는 풀스루 캐시 규칙을 사용하여 프라이빗 레지스트리의 인증이 필요한 업스트림 레지스트리에서 이미지를 캐시하는 경우 필요합니다. 이 권한은 `ecr-pullthroughcache/` 이름 접두사가 있는 보안 암호에만 적용됩니다.

`AWSECRPullThroughCache_ServiceRolePolicy` 정책에는 다음 JSON이 포함됩니다.

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



```

        "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:aws:secretsmanager:*:*:secret:ecr-pullthroughcache/*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
    }
}
]
}

```

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 연결 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성해야 합니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 권한](#)을 참조하세요.

### Amazon ECR에 대한 서비스 연결 역할 생성

풀스루 캐시에 대한 Amazon ECR 서비스 연결 역할을 수동으로 생성할 필요가 없습니다. AWS Management Console AWS CLI, 또는 AWS API에서 프라이빗 레지스트리에 대한 풀스루 캐시 규칙을 생성하면 Amazon ECR이 서비스 연결 역할을 자동으로 생성합니다.

이 서비스 연결 역할을 삭제한 다음 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 프라이빗 레지스트리에 대한 풀스루 캐시 규칙을 생성하면, Amazon ECR은 서비스 연결 역할이 아직 없는 경우 사용자를 위해 이를 다시 생성합니다.

### Amazon ECR에 대한 서비스 연결 역할 편집

Amazon ECR에서는 AWSServiceRoleForECRPullThroughCache 서비스 연결 역할을 수동으로 편집할 수 없습니다. 서비스 연결 역할이 생성된 후에는 여러 엔터티가 역할을 참조할 수 있으므로, 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 편집](#)을 참조하세요.

### Amazon ECR에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제하는 것이 좋습니다. 이렇게 하면 활성적으로 모니터링하거나 유지 관리하지 않는 미사용 개체가 없게 됩니다. 그러나 서비스 연결 역할을 수동으로 삭제하려면 먼저 모든 리전에서 레지스트리에 대한 풀스루 캐시 규칙을 삭제해야 합니다.

**Note**

Amazon ECR 서비스에서 계속 역할을 사용하고 있는 중에 리소스를 삭제하려고 하면 삭제 작업이 실패할 수 있습니다. 그 경우 몇 분 동안 기다렸다가 다시 시도하십시오.

AWSServiceRoleForECRPullThroughCache 서비스 연결 역할에서 사용되는 Amazon ECR 리소스를 삭제하는 방법

1. Amazon ECR 콘솔(<https://console.aws.amazon.com/ecr/>)을 엽니다.
2. 탐색 모음에서 풀스루 캐시 규칙이 생성된 리전을 선택합니다.
3. 탐색 창에서 프라이빗 레지스트리를 선택합니다.
4. 프라이빗 레지스트리(Private registry) 페이지의 풀스루 캐시 구성(Pull through cache configuration) 섹션에서 편집(Edit)을 선택합니다.
5. 생성한 각각의 풀스루 캐시 규칙에 대해 규칙을 선택한 후 규칙 삭제>Delete rule)를 선택합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 서비스 연결 역할을 삭제하십시오.

AWSServiceRoleForECRPullThroughCache 자세한 내용은 [IAM 사용 설명서](#)의 서비스 연결 역할 삭제를 참조하세요.

## Amazon Elastic Container Registry Identity and Access 문제 해결

다음 정보를 사용하여 Amazon ECR 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

### 주제

- [Amazon ECR에서 작업을 수행할 권한이 없음](#)
- [IAM을 수행할 권한이 없습니다. PassRole](#)
- [외부 사용자가 내 Amazon ECR 리소스에 액세스할 AWS 계정 수 있도록 허용하고 싶습니다.](#)

### Amazon ECR에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojacksonIAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 *ecr:GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
ecr:GetWidget on resource: my-example-widget
```

이 경우 *ecr:GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

## IAM을 수행할 권한이 없습니다. PassRole

*iam:PassRole* 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon ECR에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 AWS 서비스 수 있는 기능도 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 Amazon ECR에서 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 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

외부 사용자가 내 Amazon ECR 리소스에 액세스할 AWS 계정 수 있도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- Amazon ECR에서 이러한 기능을 지원하는지 여부를 알아보려면 [Amazon Elastic Container Registry가 IAM과 작동하는 방식](#)을 참조하십시오.
- 소유하고 AWS 계정 있는 모든 리소스에 대한 액세스를 [제공하는 방법을 알아보려면 IAM 사용 설명서의 다른 AWS 계정 IAM 사용자에게 액세스 권한 제공](#)을 참조하십시오.
- [제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 AWS 계정 소유에 대한 액세스 제공](#)을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [외부에서 인증된 사용자에게 액세스 권한 제공\(ID 페더레이션\)](#)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 [IAM 역할과 리소스 기반 정책의 차이](#)를 참조하세요.

## Amazon ECR에서의 데이터 보호

AWS [공동 책임 모델](#) [공동 책임 모델](#) 이 모델에 설명된 대로 AWS 은 (는) 모두를 실행하는 글로벌 인프라를 보호할 책임이 AWS 클라우드에 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 [데이터 프라이버시 FAQ](#)를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 [AWS 공동 책임 모델 및 GDPR](#) 블로그 게시물을 참조하세요.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 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 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 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 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 [FIPS\(Federal Information Processing Standard\) 140-2](#)를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 Amazon ECS 또는 기타 콘솔 AWS CLI, API 또는 AWS 서비스 AWS SDK를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

주제

- [저장된 데이터 암호화](#)

## 저장된 데이터 암호화

Amazon ECR은 Amazon ECR이 관리하는 Amazon S3 버킷에 이미지를 저장합니다. 기본적으로 Amazon ECR은 Amazon S3 관리형 암호화 키를 사용하여 서버 측 암호화를 사용합니다. 이 암호화 키는 AES-256 암호화 알고리즘을 사용하여 유휴 데이터를 암호화합니다. 이 작업에 대한 조치는 필요하지 않으며 추가 비용 없이 제공됩니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3 관리형 암호화 키\(SSE-S3\)로 서버 측 암호화를 사용하여 데이터 보호](#)를 참조하세요.

Amazon ECR 리포지토리의 암호화를 더 잘 제어하려면 () 에 저장된 KMS 키를 사용한 서버 측 암호화를 사용할 수 있습니다. AWS Key Management Service AWS KMS를 사용하여 데이터를 AWS KMS 암호화하는 경우 Amazon ECR에서 관리하는 기본값을 AWS 관리형 키사용하거나 자체 KMS 키 (고객 관리 키라고 함) 를 지정할 수 있습니다. 자세한 내용은 Amazon 심플 스토리지 서비스 사용 설명서의 [KMS 키가 저장된 서버 측 암호화 AWS KMS \(SSE-KMS\) 를 사용한 데이터 보호](#)를 참조하십시오.

각 Amazon ECR 리포지토리에는 리포지토리가 생성될 때 설정되는 암호화 구성이 있습니다. 각 리포지토리에 서로 다른 암호화 구성을 사용할 수 있습니다. 자세한 정보는 [이미지를 저장하기 위한 Amazon ECR 사실 리포지토리 생성](#)을 참조하세요.

AWS KMS 암호화가 활성화된 리포지토리를 생성하면 KMS 키를 사용하여 리포지토리의 콘텐츠를 암호화합니다. 또한 Amazon ECR은 Amazon ECR 리포지토리를 수혜자 주체로 하여 KMS 키에 AWS KMS 권한 부여를 추가합니다.

다음은 Amazon ECR이 AWS KMS 와 통합되어 리포지토리를 암호화 및 해독하는 방식의 높은 수준의 이해를 제공합니다.

1. 리포지토리를 생성할 때 Amazon ECR은 [DescribeKey](#)호출을 AWS KMS 전송하여 암호화 구성에 지정된 KMS 키의 Amazon 리소스 이름 (ARN) 을 확인하고 검색합니다.
2. Amazon ECR은 Amazon ECR이 데이터 키를 AWS KMS 사용하여 데이터를 암호화하고 복호화할 수 있도록 KMS 키에 권한을 부여해 달라는 두 가지 [CreateGrant](#)요청을 보냅니다.

3. 이미지를 푸시할 때 이미지 레이어 및 [GenerateData매니페스트](#)를 AWS KMS 암호화하는 데 사용할 KMS 키를 지정하는 키 요청이 이루어집니다.
4. AWS KMS 새 데이터 키를 생성하고 지정된 KMS 키로 암호화한 다음 암호화된 데이터 키를 전송하여 이미지 레이어 메타데이터 및 이미지 매니페스트와 함께 저장합니다.
5. 이미지를 가져오면 [암호화된 데이터 키를 지정하는 암호 해독](#) 요청이 전송됩니다 AWS KMS.
6. AWS KMS 암호화된 데이터 키를 복호화하고 해독된 데이터 키를 Amazon S3로 보냅니다.
7. 데이터 키는 이미지 레이어를 가져오기 전에 이미지 레이어를 복호화하는 데 사용됩니다.
8. 리포지토리가 삭제되면 Amazon ECR은 리포지토리에 대해 생성된 AWS KMS 권한을 폐기해 달라는 [RetireGrant](#)요청을 두 번 보냅니다.

## 고려 사항

Amazon ECR에서 AWS KMS 암호화를 사용할 때는 다음 사항을 고려해야 합니다.

- KMS 암호화를 사용하여 Amazon ECR 리포지토리를 생성하고 KMS 키를 지정하지 않는 경우 Amazon ECR은 기본적으로 별칭이 AWS 관리형 키 있는 a를 사용합니다. `aws/ecr` 이 KMS 키는 KMS 암호화가 활성화된 리포지토리를 처음 생성할 때 계정에 생성됩니다.
- 고유의 KMS 키로 KMS 암호화를 사용하는 경우, 해당 키는 리포지토리와 동일한 리전에 있어야 합니다.
- Amazon ECR이 사용자를 대신하여 생성하는 부여는 취소되지 않아야 합니다. Amazon ECR에 계정의 AWS KMS 키를 사용할 권한을 부여하는 권한을 취소하면 Amazon ECR은 이 데이터에 액세스하거나, 리포지토리로 푸시된 새 이미지를 암호화하거나, 이미지를 가져왔을 때 복호화할 수 없습니다. Amazon ECR에 대한 권한 부여를 취소하면 변경 사항이 즉시 발생합니다. 액세스 권한을 취소하려면 권한 부여를 취소하는 대신 리포지토리를 삭제합니다. 리포지토리가 삭제되면 Amazon ECR은 사용자를 대신하여 부여 권한의 사용을 중지시킵니다.
- 키 사용에는 비용이 발생합니다. AWS KMS 자세한 내용은 [AWS Key Management Service 요금](#)을 참조하세요.

## 필수 IAM 권한

AWS KMS를 사용하여 서버 측 암호화로 Amazon ECR 리포지토리를 생성하거나 삭제하는 경우, 필요한 사용 권한은 사용 중인 특정 KMS 키에 따라 다릅니다.

## Amazon AWS 관리형 키 ECR용 을 사용할 때 필요한 IAM 권한

기본적으로 Amazon ECR 리포지토리에 AWS KMS 암호화가 활성화되어 있지만 KMS 키가 지정되지 않은 경우 AWS 관리형 키 Amazon ECR용 암호화가 사용됩니다. Amazon AWS ECR용 -managed KMS 키를 사용하여 리포지토리를 암호화하는 경우 리포지토리를 생성할 권한이 있는 모든 보안 주체는 리포지토리에서 AWS KMS 암호화를 활성화할 수도 있습니다. 그러나 리포지토리를 삭제하는 IAM 보안 주체는 kms:RetireGrant 권한이 있어야 합니다. 이렇게 하면 리포지토리가 생성될 때 AWS KMS 키에 추가된 권한 부여를 폐기할 수 있습니다.

다음의 예제 IAM 정책은 암호화가 활성화된 리포지토리를 삭제하는 데 필요한 최소 권한을 갖도록 사용자에게 인라인 정책으로 추가될 수 있습니다. 리포지토리를 암호화하는 데 사용되는 KMS 키는 리소스 파라미터를 사용하여 지정할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Id": "ecr-kms-permissions",
  "Statement": [
    {
      "Sid": "AllowAccessToRetireTheGrantsAssociatedWithTheKey",
      "Effect": "Allow",
      "Action": [
        "kms:RetireGrant"
      ],
      "Resource": "arn:aws:kms:us-  
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
    }
  ]
}
```

## 고객 관리형 키를 사용할 때 필요한 IAM 권한

고객 관리 키를 사용하여 AWS KMS 암호화가 활성화된 리포지토리를 생성하는 경우 리포지토리를 생성하는 사용자 또는 역할의 KMS 키 정책과 IAM 정책 모두에 필요한 권한이 있어야 합니다.

고유한 KMS 키를 만들 때 AWS KMS 에서 생성하는 기본 키를 사용하거나 직접 지정할 수도 있습니다. 계정 소유자가 고객 관리 키를 관리할 수 있도록 하려면 KMS 키의 키 정책에서 계정의 루트 사용자에 대한 모든 AWS KMS 작업을 허용해야 합니다. 키 정책에 범위가 지정된 권한을 추가할 수 있지만 최소한 루트 사용자에게 KMS 키를 관리할 수 있는 권한이 부여되어야 합니다. Amazon ECR에서 시작된 요청에만 KMS 키를 사용할 수 있도록 하려면 [kms: ViaService 조건 키](#)를 값과 함께 사용하면 됩니다. ecr.<region>.amazonaws.com

다음 예제 키 정책은 KMS 키를 소유한 AWS 계정 (루트 사용자) 에게 KMS 키에 대한 전체 액세스 권한을 부여합니다. 이 예제 키 정책에 대한 자세한 내용은 개발자 [안내서의 AWS 계정 액세스 허용 및 IAM 정책 활성화](#)를 참조하십시오. [AWS Key Management Service](#)

```
{
  "Version": "2012-10-17",
  "Id": "ecr-key-policy",
  "Statement": [
    {
      "Sid": "EnableIAMUserPermissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:*",
      "Resource": "*"
    }
  ]
}
```

리포지토리를 생성하는 IAM 사용자, IAM 역할 또는 AWS 계정에는 필요한 Amazon ECR `kms:DescribeKey` 권한 외에 `kms:CreateGrant`, `kms:RetireGrant`, 및 권한이 있어야 합니다.

#### Note

이 `kms:RetireGrant` 권한은 리포지토리를 생성하는 사용자 또는 역할의 IAM 정책에 추가되어야 합니다. 이 `kms:CreateGrant` 및 `kms:DescribeKey` 권한은 KMS 키의 키 정책이나 리포지토리를 생성하는 사용자 또는 역할의 IAM 정책에 추가할 수 있습니다. AWS KMS 권한 작동 방식에 대한 자세한 내용은 개발자 안내서의 [AWS KMS API 권한: 작업 및 리소스 참조](#)를 참조하십시오. [AWS Key Management Service](#)

다음의 예제 IAM 정책은 암호화가 활성화된 리포지토리를 생성하고 리포지토리가 완료되면 삭제하는데 필요한 최소 권한을 갖도록 사용자에게 인라인 정책으로 추가할 수 있습니다. 리포지토리를 암호화하는 데 사용되는 AWS KMS key 는 리소스 파라미터를 사용하여 지정할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Id": "ecr-kms-permissions",
  "Statement": [
```



```

    {
      "Sid":
"AllowAccessToCreateAndRetireTheGrantsAssociatedWithTheKeyAsWellAsDescribeTheKey",
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:RetireGrant",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:us-
west-2:111122223333:key/b8d9ae76-080c-4043-92EXAMPLE"
    }
  ]
}

```

리포지토리를 생성할 때 사용자가 콘솔에서 KMS 키를 나열하도록 허용

Amazon ECR 콘솔을 사용하여 리포지토리를 생성할 때 리포지토리에 대한 암호화를 활성화할 때 사용자가 리포지토리에 암호화를 사용 시 리전에서 고객 관리형 KMS 키를 나열할 수 있는 권한을 부여할 수 있습니다. 다음 IAM 정책 예제는 콘솔을 사용할 때 KMS 키와 별칭을 나열하는 데 필요한 권한을 보여줍니다.

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListAliases",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
}

```

## AWS KMS를 통한 Amazon ECR 상호 작용 모니터링

Amazon ECR이 사용자를 대신하여 보내는 요청을 추적하는 데 AWS KMS 사용할 AWS CloudTrail 수 있습니다. 로그의 로그 항목에는 더 쉽게 식별할 수 있도록 암호화 컨텍스트 키가 포함되어 있습니다. CloudTrail

## Amazon ECR 암호화 컨텍스트

암호화 컨텍스트는 보안되지 않은 임의의 데이터를 포함하는 키-값 페어 세트입니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하면 암호화 컨텍스트가 암호화된 데이터에 AWS KMS 암호적으로 바인딩됩니다. 따라서 동일한 암호화 컨텍스트로 전달해야 이 데이터를 해독할 수 있습니다.

Amazon ECR은 [GenerateData키](#) 및 [암호 해독](#) 요청에 대해 리포지토리와 사용 중인 AWS KMS Amazon S3 버킷을 식별하는 두 개의 이름-값 쌍이 포함된 암호화 컨텍스트를 사용합니다. 방법은 다음 예제와 같습니다. 이름은 다르지 않지만 결합된 암호화 컨텍스트 값은 각 값마다 다릅니다.

```
"encryptionContext": {
  "aws:s3:arn": "arn:aws:s3::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df",
  "aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
}
```

암호화 컨텍스트를 사용하여 감사 기록 및 로그 (예: Amazon CloudWatch Logs) 에서 이러한 암호화 작업을 식별하고 정책 및 권한 부여의 승인 조건으로 사용할 수 있습니다. [AWS CloudTrail](#)

Amazon ECR 암호화 컨텍스트는 두 개의 이름-값 페어로 구성됩니다.

- `aws:s3:arn`— 첫 번째 이름-값 페어는 버킷을 식별합니다. 키는 `aws:s3:arn`입니다. 이 값은 Amazon S3 버킷의 Amazon 리소스 이름(ARN)입니다.

```
"aws:s3:arn": "ARN of an Amazon S3 bucket"
```

예를 들어, 버킷의 ARN이 `arn:aws:s3::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df`이면 암호화 컨텍스트는 다음 페어를 포함합니다.

```
"arn:aws:s3::us-west-2-starport-manifest-bucket/EXAMPLE1-90ab-cdef-fedc-ba987BUCKET1/sha256:a7766145a775d39e53a713c75b6fd6d318740e70327aaa3ed5d09e0ef33fc3df"
```

- `aws:ecr:arn`— 두 번째 이름-값 페어는 리포지토리의 Amazon 리소스 이름(ARN)을 식별합니다. 키는 `aws:ecr:arn`입니다. 이 값은 리포지토리의 ARN입니다.

```
"aws:ecr:arn": "ARN of an Amazon ECR repository"
```

예를 들어, 리포지토리의 ARN이 `arn:aws:ecr:us-west-2:111122223333:repository/repository-name`이면 암호화 컨텍스트는 다음 페어를 포함합니다.

```
"aws:ecr:arn": "arn:aws:ecr:us-west-2:111122223333:repository/repository-name"
```

## 문제 해결

콘솔로 Amazon ECR 리포지토리를 삭제할 때 리포지토리가 성공적으로 삭제되었지만 Amazon ECR 이 리포지토리의 KMS 키에 추가된 부여 권한의 사용을 중지시킬 수 없는 경우 다음과 같은 오류가 발생합니다.

```
The repository [{repository-name}] has been deleted successfully but the grants created by the kmsKey [{kms_key}] failed to be retired
```

이 경우 리포지토리에 대한 AWS KMS 허가를 직접 철회할 수 있습니다.

리포지토리에 대한 AWS KMS 지원금을 수동으로 사용 중지하려면

- 리포지토리에 사용된 AWS KMS 키에 대한 권한 부여를 나열하십시오. 이 key-id 값은 콘솔에서 받은 오류에 포함됩니다. 또한 `list-keys` 명령을 사용하여 계정의 특정 지역에 있는 KMS AWS 관리형 키 키와 고객 관리 KMS 키를 모두 나열할 수 있습니다.

```
aws kms list-grants \
  --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc
  --region us-west-2
```

출력에는 리포지토리의 Amazon 리소스 이름(ARN)과 EncryptionContextSubset가 포함됩니다. 이는 키에 추가된 부여 권한 중 사용을 중지시키려는 권한을 결정하는 데 사용할 수 있습니다. 이 GrantId 값은 다음 단계에서 부여 권한을 중지할 때 사용됩니다.

- 리포지토리에 추가된 AWS KMS 키에 대한 각 부여를 폐기하십시오. 이 값을 이전 단계 출력의 권한 `GrantId`로 대체하십시오.

```
aws kms retire-grant \
  --key-id b8d9ae76-080c-4043-9237-c815bfc21dfc \
  --grant-id GrantId \
  --region us-west-2
```

## Amazon Elastic 컨테이너 레지스트리에 대한 규정 준수 확인

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면 AWS 서비스 규정 준수 [프로그램의 AWS 서비스 범위별, 규정](#) 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 [AWS 규정 준수 프로그램 AWS 보증 프로그램 규정 AWS](#) 참조하십시오.

를 사용하여 AWS Artifact 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 <https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html> 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- [보안 및 규정 준수 킷 스타트 가이드](#) - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- [Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.](#)

### Note

모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 [HIPAA 적격 서비스 참조](#)를 참조하십시오.

- [AWS 규정 준수 리소스 AWS](#) — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- [AWS 고객 규정 준수 가이드](#) — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 [규칙을 사용하여 리소스 평가](#) — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- [AWS Security Hub](#) — 이를 AWS 서비스 통해 내부 AWS 보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 [Security Hub 제어 참조](#)를 참조하십시오.
- [Amazon GuardDuty](#) — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정 모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수

프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.

- [AWS Audit Manager](#)— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

## Amazon Elastic Container Registry의 인프라 보안

Amazon Elastic 컨테이너 레지스트리는 관리형 서비스로서 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 [AWS 클라우드 보안을](#) 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 [인프라 보호](#)를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 Amazon ECR에 액세스합니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 [AWS Security Token Service](#)(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

이러한 API 작업은 어떤 네트워크 위치에서든 호출할 수 있지만, Amazon ECR은 소스 IP 주소에 따른 제한 사항을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. Amazon ECR 정책을 사용하여 특정 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트 또는 특정 VPC에서 액세스를 제어할 수도 있습니다. 이를 통해 네트워크 내의 특정 VPC로부터 특정 Amazon ECR 리소스에 대한 네트워크 액세스를 효과적으로 분리할 수 있습니다. AWS 자세한 정보는 [Amazon ECR 인터페이스 VPC 엔드포인트 \(\)AWS PrivateLink](#)을 참조하세요.

## Amazon ECR 인터페이스 VPC 엔드포인트 ()AWS PrivateLink

인터페이스 VPC 엔드포인트를 사용하도록 Amazon ECR을 구성하여 VPC의 보안 상태를 향상시킬 수 있습니다. VPC 엔드포인트는 프라이빗 IP 주소를 통해 Amazon ECR API에 비공개로 액세스할 수 있는 기술인 에 의해 AWS PrivateLink 구동됩니다. AWS PrivateLink VPC와 Amazon ECR 간의 모든 네

트위크 트래픽을 Amazon 네트워크로 제한합니다. 인터넷 게이트웨이, NAT 디바이스 또는 가상 프라이빗 게이트웨이가 필요 없습니다.

AWS PrivateLink 및 VPC 엔드포인트에 대한 자세한 내용은 Amazon VPC 사용 설명서의 [VPC 엔드포인트](#)를 참조하십시오.

## Amazon ECR VPC 엔드포인트에 대한 고려 사항

Amazon ECR에 대해 VPC 엔드포인트를 구성하기 전에 다음 고려 사항에 유의하십시오.

- Amazon EC2 인스턴스에서 호스팅되는 Amazon ECS 작업이 Amazon ECR에서 프라이빗 이미지를 가져올 수 있도록 하려면 Amazon ECS용 인터페이스 VPC 엔드포인트도 생성해야 합니다. 자세한 내용은 Amazon Elastic 컨테이너 서비스 개발자 안내서의 [인터페이스 VPC 엔드포인트 \(AWS PrivateLink\)](#)를 참조하십시오.

### Important

Fargate 에서 호스팅되는 Amazon ECS 작업에는 Amazon ECS 인터페이스 VPC 엔드포인트가 필요하지 않습니다.

- Linux 플랫폼 버전 1.3.0 이하를 사용하여 Fargate에서 호스팅되는 Amazon ECS 작업은 `com.amazonaws.region.ecr.dkr` Amazon ECR VPC 엔드포인트 및 Amazon S3 게이트웨이 엔드포인트만을 사용하여 이 기능을 활용할 수 있습니다.
- Linux 플랫폼 버전 1.4.0 이상을 사용하여 Fargate에서 호스팅되는 Amazon ECS 태스크를 수행하려는 경우, `com.amazonaws.region.ecr.dkr` 및 `com.amazonaws.region.ecr.api` Amazon ECR VPC 엔드포인트뿐만 아니라 Amazon S3 게이트웨이 엔드포인트까지 사용해야 이 기능을 활용할 수 있습니다.
- Windows 플랫폼 버전 1.0.0 이상을 사용하여 Fargate에서 호스팅되는 Amazon ECS 태스크를 수행하려는 경우, `com.amazonaws.region.ecr.dkr` 및 `com.amazonaws.region.ecr.api` Amazon ECR VPC 엔드포인트뿐만 아니라 Amazon S3 게이트웨이 엔드포인트까지 사용해야 이 기능을 활용할 수 있습니다.
- Amazon ECR에서 컨테이너 이미지를 가져오는 Fargate에서 호스팅되는 Amazon ECS 작업은 작업에 대한 작업 실행 IAM 역할에 조건 키를 추가하여 해당 작업에 사용되는 특정 VPC 및 해당 서비스에 사용되는 VPC 엔드포인트에 대한 액세스를 제한할 수 있습니다. 자세한 내용은 Amazon Elastic Container Service 개발자 가이드의 [인터페이스 엔드포인트를 통해 Amazon ECR 이미지를 가져오는 Fargate 작업에 대한 IAM 권한\(선택사항\)](#)을 참조하십시오.

- 로그 드라이버를 사용하여 로그 정보를 Logs로 전송하는 Amazon ECR에서 컨테이너 이미지를 가져오는 Fargate에서 호스팅되는 Amazon ECS `awslogs` 작업에는 Logs VPC CloudWatch 엔드포인트가 필요합니다 CloudWatch . 자세한 정보는 [로그 엔드포인트 생성 CloudWatch](#) 을 참조하세요.
- VPC 엔드포인트에 연결된 보안 그룹은 VPC의 프라이빗 서브넷에서 443 포트로 들어오는 연결을 허용해야 합니다.
- VPC 엔드포인트는 교차 리전 요청을 현재 지원하지 않습니다. API 호출을 Amazon ECR로 발행할 계획인 동일 리전에 VPC 엔드포인트를 생성해야 합니다.
- VPC 엔드포인트는 현재 Amazon ECR 퍼블릭 리포지토리를 지원하지 않습니다. 풀스루 캐시 규칙을 사용하여 VPC 엔드포인트와 동일한 리전의 프라이빗 리포지토리에서 퍼블릭 이미지를 호스팅하는 것을 고려해 보세요. 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화](#)을 참조하세요.
- VPC 엔드포인트는 Amazon Route 53을 통해 AWS 제공된 DNS만 지원합니다. 자신의 DNS를 사용하는 경우에는 조건적인 DNS 전송을 사용할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 [DHCP 옵션 세트](#)를 참조하세요.
- 컨테이너에 Amazon S3에 대한 기존 연결이 있는 경우, Amazon S3 게이트웨이 엔드포인트를 추가할 때 그 연결이 잠시 중단될 수 있습니다. 중단을 받지 않으려면 Amazon S3 게이트웨이 엔드포인트를 사용하는 새 VPC를 생성한 후 Amazon ECS 클러스터와 그 컨테이너를 새 VPC에 마이그레이션합니다.
- 풀스루 캐시 규칙을 사용하여 이미지를 처음 가져올 때 AWS PrivateLink 를 사용하여 인터페이스 VPC 엔드포인트를 사용하도록 Amazon ECR을 구성한 경우, NAT 게이트웨이를 사용하여 동일한 VPC에 퍼블릭 서브넷을 생성해야 합니다. 그런 다음 해당 프라이빗 서브넷에서 NAT 게이트웨이로, 모든 아웃바운드 트래픽을 인터넷으로 라우팅해야 풀 작업을 수행할 수 있습니다. 후속 이미지 풀에는 이 작업이 필요하지 않습니다. 자세한 내용은 Amazon Virtual Private Cloud 사용 설명서의 [시나리오: 프라이빗 서브넷에서 인터넷 액세스](#)를 참조하세요.

## Windows 이미지에 대한 고려 사항

Windows 운영 체제 기반 이미지는 라이선스로 인해 배포가 제한되는 아티팩트가 포함됩니다. 기본적으로 Windows 이미지를 Amazon ECR 리포지토리로 푸시하는 경우, 이러한 아티팩트가 포함된 계층은 외래 계층으로 간주되어 푸시되지 않습니다. Microsoft가 아티팩트를 제공하는 경우, 외래 계층은 Microsoft Azure 인프라에서 검색됩니다. 이러한 이유로 컨테이너가 Azure에서 이러한 외래 계층을 가져올 수 있도록 하려면 VPC 엔드포인트를 만드는 것 이상의 추가 단계가 필요합니다.

Docker 데몬에서 `--allow-nondistributable-artifacts` 플래그를 사용하여 Amazon ECR로 Windows 이미지를 푸시할 때 이 동작을 무시할 수 있습니다. 활성화되면 이 플래그는 라이선스가 부여

된 계층을 Amazon ECR로 푸시합니다. 그러면 Azure에 대한 추가 액세스 없이 VPC 엔드포인트를 통해 Amazon ECR에서 이러한 이미지를 가져올 수 있습니다.

### ⚠ Important

--allow-nondistributable-artifacts 플래그의 사용은 Windows 컨테이너 기본 이미지 라이선스의 조건을 준수해야 할 의무를 배제하지 않으며 공개 또는 타사 재배포용으로 Windows 콘텐츠를 게시할 수 없습니다. 사용자 환경 내에서의 사용은 허용됩니다.

Docker 설치에 이 플래그를 사용하려면 Docker 설치에 따라 Docker 데몬 구성 파일을 수정해야 합니다. 이는 일반적으로 Docker 엔진섹션의 설정 또는 환경 설정 메뉴에서 구성하거나 C:\ProgramData\docker\config\daemon.json 파일을 직접 수정하여 구성할 수 있습니다.

다음은 필요한 구성의 예입니다. 값을 이미지를 푸시하는 리포지토리 URI로 바꿉니다.

```
{
  "allow-nondistributable-artifacts": [
    "111122223333.dkr.ecr.us-west-2.amazonaws.com"
  ]
}
```

Docker 데몬 구성 파일을 수정한 후에는 이미지를 푸시하기 전에 Docker 데몬을 다시 시작해야 합니다. 기본 계층이 리포지토리에 푸시되었는지 확인하여 푸시가 작동했는지 확인합니다.

### ℹ Note

Windows 이미지의 기본 계층은 크기가 큽니다. 큰 크기의 계층은 푸시하는 데 시간이 길어지고 Amazon ECR에서 이러한 이미지에 대한 추가 스토리지 비용을 발생시킵니다. 이러한 이유로 빌드 시간과 지속적인 저장소 비용을 줄이는 데 꼭 필요한 경우에만 이 옵션을 사용하는 것이 좋습니다. 예를 들어 mcr.microsoft.com/windows/servercore 이미지의 크기는 Amazon ECR에서 압축할 때 약 1.7GiB입니다.

## Amazon ECR용 VPC 엔드포인트 생성

Amazon ECR 서비스에 대한 VPC 엔드포인트를 생성하려면 Amazon VPC 사용 설명서의 [인터페이스 엔드포인트 생성](#) 절차를 사용합니다.



Amazon EC2 인스턴스에서 호스팅되는 Amazon ECS 작업에는 Amazon ECR 엔드포인트와 Amazon S3 게이트웨이 엔드포인트가 모두 필요합니다.

플랫폼 버전 1.4.0 이상을 사용하여 Fargate에서 호스팅되는 Amazon ECS 작업은 Amazon ECR VPC 엔드포인트와 Amazon S3 게이트웨이 엔드포인트가 모두 필요합니다.

플랫폼 버전 1.3.0 이하를 사용하여 Fargate에서 호스팅되는 Amazon ECS 작업은 `com.amazonaws.region.ecr.dkr` Amazon ECR VPC 엔드포인트 및 Amazon S3 게이트웨이 엔드포인트만을 필요로 합니다.

#### Note

엔드포인트가 생성되는 순서는 중요하지 않습니다.

`com.amazonaws.region.ecr.dkr`

이 엔드포인트는 Docker Registry API에 사용됩니다. `push` 및 `pull`와 같은 Docker 클라이언트 명령은 이 엔드포인트를 사용합니다.

이 엔드포인트를 생성할 때 프라이빗 DNS 호스트 이름을 활성화해야 합니다. 그렇게 하려면 VPC 엔드포인트를 생성할 때 Amazon VPC 콘솔에서 프라이빗 DNS 이름 활성화 옵션을 반드시 선택하십시오.

`com.amazonaws.region.ecr.api`

#### Note

지정된 `###` 미국 동부 (오하이오) AWS 지역과 같이 `us-east-2` Amazon ECR에서 지원하는 지역의 지역 식별자를 나타냅니다.

이 엔드포인트는 Amazon ECR API에 대한 호출에 사용됩니다. `DescribeImages` 및 `CreateRepository`와 같은 API 작업이 이 엔드포인트로 전송됩니다.

이 엔드포인트가 생성될 때 프라이빗 DNS 호스트 이름을 활성화할 수 있습니다(옵션). VPC 엔드포인트를 생성할 때 VPC 콘솔에서 프라이빗 DNS 이름 활성화(Enable Private DNS Name)를 선택하여 이 설정을 활성화합니다. VPC 엔드포인트에 프라이빗 DNS 호스트 이름을 활성화하는 경우 SDK를 AWS CLI 사용하거나 SDK를 사용할 때 엔드포인트 URL을 지정할 필요가 없도록 SDK를 AWS CLI 최신 버전으로 업데이트하십시오.

프라이빗 DNS 호스트 이름을 활성화하고 2019년 1월 24일 이전에 출시된 SDK 또는 AWS CLI 버전을 사용하는 경우 `--endpoint-url` 파라미터를 사용하여 인터페이스 엔드포인트를 지정해야 합니다. 다음 예제에서는 엔드포인트 URL의 형식을 보여줍니다.

```
aws ecr create-repository --repository-name name --endpoint-url https://
api.ecr.region.amazonaws.com
```

VPC 엔드포인트에 대하여 프라이빗 DNS 호스트 이름을 활성화하지 않는 경우, 그 인터페이스 엔드포인트에 대한 VPC 엔드포인트 ID를 지정하는 `--endpoint-url` 파라미터를 사용해야 합니다. 다음 예제에서는 엔드포인트 URL의 형식을 보여줍니다.

```
aws ecr create-repository --repository-name name --endpoint-url
https://VPC_endpoint_ID.api.ecr.region.vpce.amazonaws.com
```

## Amazon S3 게이트웨이 엔드포인트 생성

Amazon ECS 작업의 경우 Amazon ECR에서 프라이빗 이미지를 가져오려면 Amazon S3에 대해 게이트웨이 엔드포인트를 생성해야 합니다. Amazon ECR은 Amazon S3를 사용하여 이미지 계층을 저장하기 때문에 게이트웨이 엔드포인트가 필요합니다. 컨테이너가 Amazon ECR에서 이미지를 다운로드할 때 Amazon ECR에 액세스하여 이미지 매니페스트를 가져오고 Amazon S3에서 실제 이미지 계층을 다운로드해야 합니다. 다음은 각 Docker 이미지에 대한 계층을 포함한 Amazon S3 버킷의 Amazon 리소스 이름(ARN)입니다.

```
arn:aws:s3:::prod-region-starport-layer-bucket/*
```

Amazon VPC 사용 설명서의 [게이트웨이 엔드포인트 생성](#) 절차를 사용하여 Amazon ECR용 Amazon S3 게이트웨이 엔드포인트를 생성합니다. 엔드포인트를 생성할 때는 VPC에 대한 라우팅 테이블을 선택해야 합니다.

com.amazonaws.*region*.s3

Amazon S3 게이트웨이 엔드포인트는 IAM 정책 문서를 사용하여 서비스 액세스를 제한합니다. 작업 IAM 역할 또는 기타 사용자 정책에 입력한 모든 제한 사항이 여전히 이 정책보다 먼저 적용되기에 모든 액세스(Full Access) 정책을 사용할 수 있습니다. Amazon S3 버킷 액세스 권한을 Amazon ECR 사용에 필요한 최소 필수 권한으로 제한하려면 [Amazon ECR에 대한 최소 Amazon S3 버킷 권한](#)을(를) 참조하십시오.

### Amazon ECR에 대한 최소 Amazon S3 버킷 권한

Amazon S3 게이트웨이 엔드포인트는 IAM 정책 문서를 사용하여 서비스 액세스를 제한합니다. Amazon ECR에 대한 최소 Amazon S3 버킷 권한만 허용하려면 엔드포인트에 대한 IAM 정책 설명을 생성할 때 Amazon ECR이 사용하는 Amazon S3 버킷에 대한 액세스를 제한하십시오.

다음 테이블은 Amazon ECR에 필요한 Amazon S3 버킷 정책 권한을 설명합니다.

권한	설명
arn:aws:s3:::prod- <i>region</i> -starport-layer-bucket/*	각 Docker 이미지에 대한 계층이 포함된 Amazon S3 버킷에 대한 액세스를 제공합니다. us-east-2 는 미국 동부(오하이오) 리전과 같이 Amazon ECR이 지원하는 AWS 리전의 리전 식별자를 나타냅니다.

예

다음 예는 Amazon ECR 작업에 필요한 Amazon S3 버킷에 액세스 권한을 부여하는 방법입니다.

```
{
  "Statement": [
    {
      "Sid": "Access-to-specific-bucket-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::prod-region-starport-layer-bucket/*"]
    }
  ]
}
```

### 로그 엔드포인트 생성 CloudWatch

로그 드라이버를 사용하여 로그 정보를 로그로 전송하는 인터넷 게이트웨이 없이 VPC를 사용하는 Fargate 시작 유형을 사용하는 **awslogs** Amazon ECS 작업을 수행하려면 CloudWatch com.amazonaws를 생성해야 합니다. 로그용 **##** .logs 인터페이스 VPC 엔드포인트 CloudWatch 자세

한 내용은 Amazon CloudWatch Logs 사용 설명서의 인터페이스 VPC 엔드포인트에서 CloudWatch 로그 [사용을](#) 참조하십시오.

## Amazon ECR VPC 엔드포인트의 엔드포인트 정책 생성

VPC 엔드포인트 정책은 엔드포인트를 만들거나 수정 시 엔드포인트에 연결하는 IAM 리소스 정책입니다. 엔드포인트를 생성할 때 정책을 연결하지 않으면 서비스에 대한 전체 액세스를 허용하는 기본 정책을 자동으로 연결합니다. AWS 엔드포인트 정책은 사용자 정책 또는 서비스별 정책을 무시하거나 교체하지 않습니다. 이는 엔드포인트에서 지정된 서비스로의 액세스를 제어하기 위한 별도의 정책입니다. 엔드포인트 정책은 JSON 형식으로 작성해야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 [VPC 엔드포인트를 통해 서비스에 대한 액세스 제어](#)를 참조하세요.

단일 IAM 리소스 정책을 생성하고 이를 Amazon ECR VPC 엔드포인트 모두에 연결하는 것이 좋습니다.

다음은 Amazon ECR에 대한 엔드포인트 정책의 예입니다. 이 정책은 특정 IAM 역할을 활성화하여 Amazon ECR에서 이미지를 가져옵니다.

```
{
  "Statement": [{
    "Sid": "AllowPull",
    "Principal": {
      "AWS": "arn:aws:iam::1234567890:role/role_name"
    },
    "Action": [
      "ecr:BatchGetImage",
      "ecr:GetDownloadUrlForLayer",
      "ecr:GetAuthorizationToken"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

다음 엔드포인트 정책 예제는 지정된 리포지토리가 삭제되는 것을 방지합니다.

```
{
  "Statement": [{
    "Sid": "AllowAll",
    "Principal": "*",
    "Action": "*",
```

```

    "Effect": "Allow",
    "Resource": "*"
  },
  {
    "Sid": "PreventDelete",
    "Principal": "*",
    "Action": "ecr:DeleteRepository",
    "Effect": "Deny",
    "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  }
]
}

```

다음 엔드포인트 정책 예제는 앞의 두 예제를 단일 정책에 결합합니다.

```

{
  "Statement": [{
    "Sid": "AllowAll",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "*",
    "Resource": "*"
  },
  {
    "Sid": "PreventDelete",
    "Effect": "Deny",
    "Principal": "*",
    "Action": "ecr:DeleteRepository",
    "Resource": "arn:aws:ecr:region:1234567890:repository/repository_name"
  },
  {
    "Sid": "AllowPull",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::1234567890:role/role_name"
    },
    "Action": [
      "ecr:BatchGetImage",
      "ecr:GetDownloadUrlForLayer",
      "ecr:GetAuthorizationToken"
    ],
    "Resource": "*"
  }
]
}

```

```
]
}
```

## Amazon ECR에 대한 VPC 엔드포인트 정책을 수정하는 방법

1. <https://console.aws.amazon.com/vpc/>에서 Amazon VPC 콘솔을 엽니다.
2. 탐색 창에서 엔드포인트를 선택합니다.
3. Amazon ECR에 대한 VPC 엔드포인트를 아직 생성하지 않은 경우 [Amazon ECR용 VPC 엔드포인트 생성](#) 단원을 참조하십시오.
4. 정책을 추가할 Amazon ECR VPC 엔드포인트를 선택하고 화면 하단의 정책(Policy) 탭을 선택합니다.
5. 정책 편집(Edit Policy)을 선택하고 정책을 변경합니다.
6. 저장(Save)을 선택하여 정책을 저장합니다.

## 공유 서브넷

공유하는 서브넷의 VPC 엔드포인트는 생성, 설명, 수정 또는 삭제할 수 없습니다. 그러나 공유하는 서브넷의 VPC 엔드포인트를 사용할 수는 있습니다.

## 교차 서비스 혼동된 대리인 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 예서 크로스 서비스 사칭은 AWS대리인 문제로 혼란스러운 결과를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접 호출하는 서비스)가 다른 서비스(직접 호출되는 서비스)를 직접 호출할 때 발생할 수 있습니다. 직접 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

Amazon ECR이 리소스에 다른 서비스를 제공하는 권한을 제한하려면 리소스 정책에서 [aws:SourceArn](#) 및 [aws:SourceAccount](#) 글로벌 조건 컨텍스트 키를 사용하는 것이 좋습니다. 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 [aws:SourceArn](#)를 사용하십시오. 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 [aws:SourceAccount](#)를 사용하세요.

혼동된 대리자 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 [aws:SourceArn](#) 전역 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN

을 모르거나 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드 문자(\*)를 포함한 `aws:SourceArn` 글로벌 조건 컨텍스트 키를 사용합니다. 예를 들어 `arn:aws:servicename:region:123456789012:*`입니다.

만약 `aws:SourceArn` 값에 Amazon S3 버킷 ARN과 같은 계정 ID가 포함되어 있지 않은 경우, 권한을 제한하려면 두 글로벌 조건 컨텍스트 키를 모두 사용해야 합니다.

`aws:SourceArn` 값은 ResourceDescription이어야 합니다.

다음 예제는 Amazon ECR 리포지토리 정책의 `aws:SourceAccount` 글로벌 조건 컨텍스트 키를 사용하여 해당 `aws:SourceArn` 서비스와의 통합에 필요한 Amazon ECR API 작업에 대한 AWS CodeBuild 액세스를 허용하는 동시에 혼동되는 부정 문제를 방지하는 방법을 보여줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccess",
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": [
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:codebuild:region:123456789012:project/project-  
name"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

# Amazon ECR 모니터링

Amazon ECR에서 원시 데이터를 수집하여 읽기 쉬운 거의 실시간 지표로 처리하는 CloudWatch Amazon을 통해 Amazon ECR API 사용을 모니터링할 수 있습니다. 이러한 통계는 2주 동안 기록되므로 기록 정보에 액세스하고 API 사용에 대한 관점을 얻을 수 있습니다. Amazon ECR 메트릭 데이터는 1분 CloudWatch 내에 자동으로 전송됩니다. 에 대한 CloudWatch 자세한 내용은 [Amazon CloudWatch 사용 설명서를](#) 참조하십시오.

Amazon ECR은 권한 부여, 이미지 푸시 및 이미지 가져오기 작업에 대한 API 사용량을 기반으로 하는 지표를 제공합니다.

모니터링은 Amazon ECR과 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어 중요한 부분입니다. 다중 지점 장애가 발생할 경우 더 쉽게 디버깅할 수 있도록 AWS 솔루션을 구성하는 리소스에서 모니터링 데이터를 수집하는 것이 좋습니다. 하지만 Amazon ECR 모니터링을 시작하기 전에 다음 질문에 대한 답변을 포함하는 모니터링 계획을 수립해야 합니다.

- 모니터링의 목표
- 모니터링할 리소스
- 이러한 리소스를 모니터링하는 빈도
- 사용할 모니터링 도구
- 모니터링 작업을 수행할 사람
- 문제 발생 시 알려야 할 대상

다음 단계에서는 다양한 시간과 다양한 부하 조건에서 성능을 측정하여 환경에서 일반 Amazon ECR 성능의 기준선을 설정합니다. Amazon ECR을 모니터링할 때 새 성능 데이터와 비교할 수 있도록 과거 모니터링 데이터를 저장하고 일반적인 성능 패턴과 성능 이상을 식별하고 이를 해결할 방법을 고안합니다.

## 주제

- [서비스 할당량 시각화 및 경보 설정](#)
- [Amazon ECR 사용량 지표](#)
- [Amazon ECR 사용 보고서](#)
- [Amazon ECR 리포지토리 지표](#)
- [아마존 ECR 이벤트 및 EventBridge](#)



- [를 사용하여 Amazon ECR 작업 로깅 AWS CloudTrail](#)

## 서비스 할당량 시각화 및 경보 설정

CloudWatch 콘솔을 사용하여 서비스 할당량을 시각화하고 현재 사용량이 서비스 할당량과 어떻게 비교되는지 확인할 수 있습니다. 할당량에 가까워지면 알림을 받도록 경보를 설정할 수도 있습니다.

서비스 할당량을 시각화하고 선택적으로 경보를 설정하려면

1. <https://console.aws.amazon.com/cloudwatch/> 에서 콘솔을 엽니다. CloudWatch
2. 탐색 창에서 지표(Metrics)를 선택합니다.
3. 모든 지표(All metrics) 탭에서 사용량(Usage)을 선택한 다음 AWS 리소스별(By AWS Resource)을 선택합니다.

서비스 할당량 사용량 지표 목록이 나타납니다.

4. 지표 중 하나 옆에 있는 확인란을 선택합니다.

그래프에는 해당 AWS 리소스의 현재 사용량이 표시됩니다.

5. 그래프에 서비스 할당량을 추가하려면 다음을 수행합니다.
  - a. 그래프로 표시된 지표(Graphed metrics) 탭을 선택합니다.
  - b. 수학적 표현식(Math expression), 빈 표현식으로 시작(Start with an empty expression)을 선택합니다. 그런 다음 새 행의 세부 정보(Details)에 **SERVICE\_QUOTA(m1)**를 입력합니다.

그래프에 새 줄이 추가되어 지표에 표시된 리소스에 대한 서비스 할당량을 표시합니다.

6. 현재 사용량을 할당량의 백분율로 보려면 새 표현식을 추가하거나 현재 SERVICE\_QUOTA 표현식을 변경합니다. 새 표현식의 경우 **m1/60/SERVICE\_QUOTA(m1)\*100**을 사용합니다.
7. (선택 사항) 서비스 할당량에 접근하는 경우 알려주는 경보를 설정하려면 다음을 수행합니다.
  - a. **m1/60/SERVICE\_QUOTA(m1)\*100** 행의 작업(Actions)에서 경보 아이콘을 선택합니다. 이 아이콘은 종처럼 보입니다.

경보 생성 페이지가 나타납니다.

- b. 조건(Conditions)에서 임계값 유형(Threshold type)이 정적(Static)이고 표현식1이 해당할 때마다(Whenever Expression1 is)가 큼(Greater)으로 설정되었는지 확인합니다. 보다(than)에 **80**을 입력합니다. 이렇게 하면 사용량이 할당량의 80%를 초과할 경우 ALARM 상태가 되는 경보가 생성됩니다.

- c. 다음(Next)을 선택합니다.
- d. 다음 페이지에서 Amazon SNS 주제를 선택하거나 새 주제를 생성합니다. 이 주제는 경보가 ALARM 상태로 전환되면 알림을 받습니다. 그런 다음, 다음(Next)을 선택합니다.
- e. 다음 페이지에서 경보의 이름과 설명을 입력하고 다음(Next)을 선택합니다.
- f. 경보 생성(Create alarm)을 선택합니다.

## Amazon ECR 사용량 지표

CloudWatch 사용량 지표를 사용하여 계정의 리소스 사용에 대한 가시성을 제공할 수 있습니다. 이러한 지표를 사용하여 CloudWatch 그래프와 대시보드에서 현재 서비스 사용량을 시각화할 수 있습니다.

Amazon ECR 사용 지표는 AWS 서비스 할당량에 해당합니다. 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수 있습니다. Amazon ECR 서비스 할당량에 대한 자세한 내용은 [Amazon ECR 서비스 할당량](#) 단원을 참조하십시오.

Amazon ECR은 AWS/Usage 네임스페이스에 다음 지표를 게시합니다.

지표	설명
CallCount	계정에서 API 작업 호출 횟수입니다. 리소스는 지표와 연결된 차원에 의해 정의됩니다.  이 지표에 대한 가장 유용한 통계는 SUM입니다. 이 통계는 정의된 기간 동안 모든 기여자의 값의 합계를 나타냅니다.

다음 차원은 Amazon ECR에 의해 게시되는 사용량 지표를 구체화하는 데 사용됩니다.

측정기준	설명
Service	리소스가 포함된 AWS 서비스의 이름. Amazon ECR 사용량 지표의 경우 이 차원 값은 ECR입니다.
Type	보고되는 엔터티의 유형입니다. 현재 Amazon ECR 사용량 지표에 대한 유일한 유효 값은 API입니다.
Resource	실행 중인 리소스의 유형입니다. 현재 Amazon ECR은 다음 API 작업에 대한 API 사용량 정보를 반환합니다.

측정기준	설명
	<ul style="list-style-type: none"> <li>• GetAuthorizationToken</li> <li>• BatchCheckLayerAvailability</li> <li>• InitiateLayerUpload</li> <li>• UploadLayerPart</li> <li>• CompleteLayerUpload</li> <li>• PutImage</li> <li>• BatchGetImage</li> <li>• GetDownloadUrlForLayer</li> </ul>
Class	추적 중인 리소스의 클래스입니다. 현재 Amazon ECR에서는 클래스 차원을 사용하지 않습니다.

## Amazon ECR 사용 보고서

AWS Amazon ECR 리소스의 비용 및 사용량을 분석할 수 있는 Cost Explorer라는 무료 보고 도구를 제공합니다.

Cost Explorer를 사용하여 사용량 및 비용 차트를 볼 수 있습니다. 이전 13개월의 데이터를 볼 수 있으며 향후 3개월 동안의 지출을 예상해볼 수 있습니다. Cost Explorer를 사용하면 시간 경과에 따라 AWS 리소스에 지출하는 금액의 패턴을 보고, 추가 질의가 필요한 영역을 식별하며, 비용을 이해하는 데 사용할 수 있는 추세를 알아볼 수 있습니다. 또한 데이터의 시간 범위를 지정하고 일별 또는 월별 시간 데이터를 볼 수도 있습니다.

비용 및 사용량 보고서의 측정 데이터는 모든 Amazon ECR 리포지토리에서의 사용량을 보여줍니다. 자세한 정보는 [리소스에 결제용 태그 지정](#)을 참조하세요.

AWS 비용 및 사용 보고서 생성에 대한 자세한 내용은 사용 설명서의 [AWS AWS Billing 비용 및 사용 보고서를](#) 참조하십시오.

## Amazon ECR 리포지토리 지표

Amazon ECR은 리포지토리 풀 카운트 지표를 CloudWatch Amazon에 전송합니다. Amazon ECR 메트릭 데이터는 1분 CloudWatch 내에 자동으로 전송됩니다. 에 대한 CloudWatch 자세한 내용은 [Amazon CloudWatch 사용 설명서](#)를 참조하십시오.

## 주제

- [CloudWatch 지표 활성화](#)
- [사용 가능한 지표 및 차원](#)
- [콘솔을 사용하여 Amazon ECR 지표 보기 CloudWatch](#)

## CloudWatch 지표 활성화

Amazon ECR은 모든 리포지토리에 대해 리포지토리 지표를 자동 전송합니다. 수동 단계를 수행할 필요가 없습니다.

## 사용 가능한 지표 및 차원

다음 섹션에는 Amazon ECR이 Amazon으로 전송하는 지표와 차원이 나열되어 있습니다. CloudWatch

### Amazon ECR 지표

Amazon ECR은 리포지토리의 모니터링을 위한 지표를 제공합니다. 풀 횟수를 측정할 수 있습니다.

AWS/ECR 네임스페이스에 포함된 지표는 다음과 같습니다.

#### RepositoryPullCount

리포지토리에 있는 이미지에 대한 총 풀 횟수입니다.

유효한 차원: RepositoryName

유효한 통계: Average, Minimum, Maximum, Sum, Sample Count. 가장 유용한 통계는 Sum입니다.

단위: 정수.

### Amazon ECR 지표 차원

Amazon ECR 지표는 AWS/ECR 네임스페이스를 사용하며 다음 차원의 지표를 제공합니다.

#### RepositoryName

이 차원은 지정한 리포지토리 내의 모든 컨테이너 이미지에 대해 요청하는 데이터를 필터링합니다.

## 콘솔을 사용하여 Amazon ECR 지표 보기 CloudWatch

CloudWatch 콘솔에서 Amazon ECR 리포지토리 지표를 볼 수 있습니다. CloudWatch 콘솔은 리소스를 세밀하게 사용자 지정할 수 있는 디스플레이를 제공합니다. 자세한 내용은 [Amazon CloudWatch 사용 설명서](#)를 참조하십시오.

CloudWatch 콘솔에서 지표를 보려면

1. <https://console.aws.amazon.com/cloudwatch/> 에서 CloudWatch 콘솔을 엽니다.
2. 탐색 창에서 지표(Metrics), 모든 지표(All metrics)를 선택합니다.
3. 찾아보기(Browse) 탭의 AWS 네임스페이스(AWS Namespaces)에서 ECR을 선택합니다.
4. 확인할 지표를 선택합니다. 리포지토리 지표의 범위는 ECR > 리포지토리 메트릭(Repository Metrics)에서 설정됩니다.

## 아마존 ECR 이벤트 및 EventBridge

Amazon을 EventBridge 사용하면 AWS 서비스를 자동화하고 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. AWS 서비스에서 발생하는 이벤트는 거의 EventBridge 실시간으로 전송됩니다. 관심 있는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동 작업을 포함할 수 있습니다. 자동으로 트리거할 수 있는 태스크는 다음과 같습니다.

- 로그의 로그 그룹에 이벤트 추가 CloudWatch
- 함수 호출 AWS Lambda
- Amazon EC2 Run Command 호출
- Amazon Kinesis Data Streams로 이벤트 릴레이
- 스테이트 머신 활성화 AWS Step Functions
- SNS 주제 또는 Amazon SQS 대기열 알림

자세한 내용은 Amazon EventBridge 사용 설명서의 [Amazon EventBridge 시작하기](#)를 참조하십시오.

## Amazon ECR의 샘플 이벤트

다음은 Amazon ECR의 예시 이벤트입니다. 이벤트는 최선의 작업을 기반으로 발생합니다.

완료된 이미지 푸시에 대한 이벤트

각 이미지 푸시가 완료되면 다음 이벤트가 전송됩니다. 자세한 정보는 [Docker 이미지를 Amazon ECR 프라이빗 리포지토리로 푸시하기](#)을 참조하세요.

```
{
  "version": "0",
  "id": "13cde686-328b-6117-af20-0e5566167482",
  "detail-type": "ECR Image Action",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2019-11-16T01:54:34Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "result": "SUCCESS",
    "repository-name": "my-repository-name",
    "image-digest":
      "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
    "action-type": "PUSH",
    "image-tag": "latest"
  }
}
```

### 풀스루 캐시 작업에 대한 이벤트

풀스루 캐시 작업이 시도되면 다음 이벤트가 전송됩니다. 자세한 정보는 [업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화](#)을 참조하세요.

```
{
  "version": "0",
  "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
  "detail-type": "ECR Pull Through Cache Action",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2023-02-29T02:36:48Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ecr:us-west-2:123456789012:repository/docker-hub/alpine"
  ],
  "detail": {
    "rule-version": "1",
    "sync-status": "SUCCESS",
    "ecr-repository-prefix": "docker-hub",
    "repository-name": "docker-hub/alpine",
  }
}
```

```

    "upstream-registry-url": "public.ecr.aws",
    "image-tag": "3.17.2",
    "image-digest":
      "sha256:4aa08ef415aecc80814cb42fa41b658480779d80c77ab15EXAMPLE",
  }
}

```

### 완료된 이미지 스캔에 대한 이벤트(기본 스캔)

레지스트리에 대한 기본 스캔이 사용 설정되면 각 이미지 스캔이 완료될 때 다음 이벤트가 전송됩니다. `finding-severity-counts` 파라미터는 심각도 수준이 있는 경우에만 값을 반환합니다. 예를 들어, 이미지에 CRITICAL 수준의 결과가 없으면 심각 카운트가 반환되지 않습니다. 자세한 정보는 [Amazon ECR의 OS 취약성에 대한 이미지 스캔](#)을 참조하세요.

#### Note

고급 스캔이 사용 설정된 경우 Amazon Inspector에서 발생하는 이벤트에 대한 자세한 내용은 [EventBridge Amazon ECR에서 향상된 스캔을 위해 전송된 이벤트](#) 섹션을 참조하세요.

```

{
  "version": "0",
  "id": "85fc3613-e913-7fc4-a80c-a3753e4aa9ae",
  "detail-type": "ECR Image Scan",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2019-10-29T02:36:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ecr:us-east-1:123456789012:repository/my-repository-name"
  ],
  "detail": {
    "scan-status": "COMPLETE",
    "repository-name": "my-repository-name",
    "finding-severity-counts": {
      "CRITICAL": 10,
      "MEDIUM": 9
    },
    "image-digest":
      "sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
    "image-tags": []
  }
}

```

}

## 고급 스캔을 사용 설정한 리소스의 변경 알림에 대한 이벤트(고급 스캔)

레지스트리에 대해 고급 스캔이 사용 설정된 경우 고급 스캔이 활성화된 리소스가 변경되면 Amazon ECR에서 다음 이벤트를 전송합니다. 여기에는 생성 중인 새 리포지토리, 변경 중인 리포지토리의 스캔 빈도 또는 고급 스캔이 사용 설정된 리포지토리에서 이미지를 생성하거나 삭제하는 시점이 포함됩니다. 자세한 정보는 [이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다.](#)을 참조하세요.

```
{
  "version": "0",
  "id": "0c18352a-a4d4-6853-ef53-0ab8638973bf",
  "detail-type": "ECR Scan Resource Change",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2021-10-14T20:53:46Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "action-type": "SCAN_FREQUENCY_CHANGE",
    "repositories": [{
      "repository-name": "repository-1",
      "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-1",
      "scan-frequency": "SCAN_ON_PUSH",
      "previous-scan-frequency": "MANUAL"
    },
    {
      "repository-name": "repository-2",
      "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-2",
      "scan-frequency": "CONTINUOUS_SCAN",
      "previous-scan-frequency": "SCAN_ON_PUSH"
    },
    {
      "repository-name": "repository-3",
      "repository-arn": "arn:aws:ecr:us-east-1:123456789012:repository/repository-3",
      "scan-frequency": "CONTINUOUS_SCAN",
      "previous-scan-frequency": "SCAN_ON_PUSH"
    }
  ],
  "resource-type": "REPOSITORY",
  "scan-type": "ENHANCED"
}
```



```
}

```

## 이미지 삭제에 대한 이벤트

이미지가 삭제되면 다음 이벤트가 전송됩니다. 자세한 정보는 [Amazon ECR에서 이미지 삭제](#)을 참조하세요.

```
{
  "version": "0",
  "id": "dd3b46cb-2c74-f49e-393b-28286b67279d",
  "detail-type": "ECR Image Action",
  "source": "aws.ecr",
  "account": "123456789012",
  "time": "2019-11-16T02:01:05Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "result": "SUCCESS",
    "repository-name": "my-repository-name",
    "image-digest":
"sha256:7f5b2640fe6fb4f46592dfd3410c4a79dac4f89e4782432e0378abcd1234",
    "action-type": "DELETE",
    "image-tag": "latest"
  }
}
```

## 를 사용하여 Amazon ECR 작업 로깅 AWS CloudTrail

Amazon ECR은 Amazon ECR에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스와 통합됩니다. CloudTrail 다음 Amazon ECR 작업을 이벤트로 캡처합니다.

- Amazon ECR 콘솔의 호출을 포함한 모든 API 호출
- 리포지토리의 암호화 설정으로 인해 수행된 모든 작업
- 수명 주기 정책 규칙으로 인해 수행된 모든 작업(성공 및 실패 작업 모두 포함)

### Important

개별 CloudTrail 이벤트의 크기 제한으로 인해 10개 이상의 이미지가 만료된 수명 주기 정책 작업의 경우 Amazon ECR은 여러 이벤트를 에 전송합니다. CloudTrail 또한 Amazon ECR에서는 이미지당 최대 100개의 태그를 추가할 수 있습니다.

트레일이 생성되면 Amazon ECR용 CloudTrail 이벤트를 포함하여 Amazon S3 버킷으로 이벤트를 지속적으로 전송할 수 있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 가장 최근 이벤트를 계속 볼 수 있습니다. 이 정보를 사용하여 Amazon ECR에 수행된 요청, 요청이 발생하는 IP 주소, 요청을 수행한 사용자, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

자세한 내용은 [AWS CloudTrail 사용 설명서](#)를 참조하십시오.

## 아마존 ECR 정보는 CloudTrail

CloudTrail 계정을 생성하면 AWS 계정에서 활성화됩니다. Amazon ECR에서 활동이 발생하면 해당 활동이 CloudTrail 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고, 검색하고, 다운로드할 수 있습니다. 자세한 내용은 이벤트 [기록으로 CloudTrail 이벤트 보기를](#) 참조하십시오.

Amazon ECR 이벤트를 포함하여 AWS 계정의 지속적인 이벤트 기록을 보려면 트레일을 생성하십시오. 트레일을 사용하면 CloudTrail Amazon S3 버킷에 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성할 때 추적을 단일 리전 또는 모든 리전에 적용할 수 있습니다. 트레일은 AWS 파티션에 이벤트를 기록하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 분석하고 이에 따라 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하세요.

- [AWS 계정 트레일 생성](#)
- [AWS 로그와 CloudTrail 서비스 통합](#)
- [예 대한 Amazon SNS 알림 구성 CloudTrail](#)
- [여러 지역에서 CloudTrail 로그 파일 수신 및 여러 계정으로부터 CloudTrail 로그 파일 수신](#)

모든 Amazon ECR API 작업은 Amazon [Elastic 컨테이너 레지스트리 API](#) 참조에 의해 CloudTrail 기록되고 문서화됩니다. 일반적인 작업을 수행하는 경우 해당 작업의 일부인 각 API 작업에 대한 섹션이 CloudTrail 로그 파일에 생성됩니다. 예를 GetAuthorizationToken 들어 CreateRepository 리포지토리를 만들면 CloudTrail 로그 파일에 SetRepositoryPolicy 섹션이 생성됩니다. 이미지를 리포지토리로 푸시하면 InitiateLayerUpload, UploadLayerPart, CompleteLayerUpload 및 PutImage 섹션이 생성됩니다. 이미지를 가져오면 GetDownloadUrlForLayer 및 BatchGetImage 섹션이 생성됩니다. 이러한 일반적인 작업의 예시는 [CloudTrail 로그 입력 예제](#)를 참조하십시오.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에게 대한 정보가 들어 있습니다. 신원 정보를 이용하면 다음을 쉽게 알아볼 수 있습니다.

- 요청을 루트로 했는지 아니면 사용자 자격 증명으로 했는지 여부
- 역할 또는 연합된 사용자에게 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부

자세한 내용은 [CloudTrailuserIdentity요소를 참조하십시오](#).

## Amazon ECR 로그 파일 항목 이해

트레일은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함되어 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업에 대한 정보, 작업 날짜 및 시간, 요청 매개 변수 및 기타 정보를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

### CloudTrail 로그 입력 예제

다음은 몇 가지 일반적인 Amazon ECR 작업에 대한 CloudTrail 로그 입력 예제입니다.

#### Note

이들 예제는 서식을 조정하여 가독성을 높인 것입니다. CloudTrail 로그 파일에서 모든 항목과 이벤트는 한 줄로 연결됩니다. 또한 이 예제는 단일 Amazon ECR 항목으로 제한된 것입니다. 실제 CloudTrail 로그 파일에서는 여러 서비스의 항목과 이벤트를 볼 수 있습니다. AWS

#### 주제

- [예: 리포지토리 생성 작업](#)
- [예: Amazon ECR 리포지토리를 생성할 때의 AWS KMS CreateGrant API 작업](#)
- [예제: 이미지 푸시 작업](#)
- [예제: 이미지 가져오기 작업](#)
- [예제: 이미지 수명 주기 정책 작업](#)

#### 예: 리포지토리 생성 작업

다음 예제는 CreateRepository 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

```
{
  "eventVersion": "1.04",
```

```
"userIdentity": {
  "type": "AssumedRole",
  "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
  "arn": "arn:aws:sts::123456789012:user/Mary_Major",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2018-07-11T21:54:07Z"
    },
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AIDACKCEVSQ6C2EXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/Admin",
      "accountId": "123456789012",
      "userName": "Admin"
    }
  }
},
"eventTime": "2018-07-11T22:17:43Z",
"eventSource": "ecr.amazonaws.com",
"eventName": "CreateRepository",
"awsRegion": "us-east-2",
"sourceIPAddress": "203.0.113.12",
"userAgent": "console.amazonaws.com",
"requestParameters": {
  "repositoryName": "testrepo"
},
"responseElements": {
  "repository": {
    "repositoryArn": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
    "repositoryName": "testrepo",
    "repositoryUri": "123456789012.dkr.ecr.us-east-2.amazonaws.com/testrepo",
    "createdAt": "Jul 11, 2018 10:17:44 PM",
    "registryId": "123456789012"
  }
},
"requestID": "cb8c167e-EXAMPLE",
"eventID": "e3c6f4ce-EXAMPLE",
"resources": [
  {
    "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
    "accountId": "123456789012"
  }
]
```

```

    }
  ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

예: Amazon ECR 리포지토리를 생성할 때의 AWS KMS CreateGrant API 작업

다음 예는 KMS 암호화가 활성화된 Amazon ECR 리포지토리를 생성할 때의 AWS KMS CreateGrant 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다. KMS 암호화가 활성화되어 생성된 각 리포지토리의 경우 두 개의 CreateGrant 로그 항목이 표시되어야 합니다. CloudTrail

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAIEP6W46J43IG7LXAQ",
    "arn": "arn:aws:iam::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "Mary_Major",
    "sessionContext": {
      "sessionIssuer": {
        },
      "webIdFederationData": {
        },
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-06-10T19:22:10Z"
      }
    }
  },
  "invokedBy": "AWS Internal"
},
"eventTime": "2020-06-10T19:22:10Z",
"eventSource": "kms.amazonaws.com",
"eventName": "CreateGrant",
"awsRegion": "us-west-2",
"sourceIPAddress": "203.0.113.12",
"userAgent": "console.amazonaws.com",
"requestParameters": {
  "keyId": "4b55e5bf-39c8-41ad-b589-18464af7758a",

```

```

    "granteePrincipal": "ecr.us-west-2.amazonaws.com",
    "operations": [
      "GenerateDataKey",
      "Decrypt"
    ],
    "retiringPrincipal": "ecr.us-west-2.amazonaws.com",
    "constraints": {
      "encryptionContextSubset": {
        "aws:ecr:arn": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo"
      }
    }
  },
  "responseElements": {
    "grantId": "3636af9adfee1accb67b83941087dcd45e7fadc4e74ff0103bb338422b5055f3"
  },
  "requestID": "047b7dea-b56b-4013-87e9-a089f0f6602b",
  "eventID": "af4c9573-c56a-4886-baca-a77526544469",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:us-west-2:123456789012:key/4b55e5bf-39c8-41ad-
b589-18464af7758a"
    }
  ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

예제: 이미지 푸시 작업

다음 예제는 해당 작업을 사용하는 이미지 푸시를 보여주는 CloudTrail 로그 항목을 보여줍니다.

PutImage

#### Note

이미지를 푸시할 때는 로그에 InitiateLayerUploadUploadLayerPart, 및 CompleteLayerUpload 참조도 표시됩니다. CloudTrail

```
{
```

```

    "eventVersion": "1.04",
    "userIdentity": {
      "type": "IAMUser",
      "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
      "arn": "arn:aws:sts::123456789012:user/Mary_Major",
      "accountId": "123456789012",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "userName": "Mary_Major",
      "sessionContext": {
        "attributes": {
          "mfaAuthenticated": "false",
          "creationDate": "2019-04-15T16:42:14Z"
        }
      }
    },
    "eventTime": "2019-04-15T16:45:00Z",
    "eventSource": "ecr.amazonaws.com",
    "eventName": "PutImage",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "203.0.113.12",
    "userAgent": "console.amazonaws.com",
    "requestParameters": {
      "repositoryName": "testrepo",
      "imageTag": "latest",
      "registryId": "123456789012",
      "imageManifest": "{\n  \"schemaVersion\": 2,\n  \"mediaType\": \"application/\n  vnd.docker.distribution.manifest.v2+json\",\n  \"config\": {\n    \"mediaType\":\n  \"application/vnd.docker.container.image.v1+json\",\n    \"size\": 5543,\n    \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a\n  \"\n  },\n  \"layers\": [\n    {\n      \"mediaType\": \"application/\n  vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 43252507,\n      \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e\n  \"\n    },\n    {\n      \"mediaType\": \"application/\n  vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 846,\n      \"digest\n  \": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd\n  \"\n    },\n    {\n      \"mediaType\": \"application/\n  vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 615,\n      \"digest\n  \": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n  \"\n    },\n    {\n      \"mediaType\": \"application/\n  vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 850,\n      \"digest\n  \": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a\n  \"\n    },\n    {\n      \"mediaType\": \"application/\n  vnd.docker.image.rootfs.diff.tar.gzip\",\n      \"size\": 168,\n      \"digest\":\n  \"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\"\n  \"\n    },

```

```

\n      {\n          \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\",\n          \"size\": 37720774,\n          \"digest\":
  \"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\"\n
      },\n      {\n          \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n          \"size\": 30432107,\n
  \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\"\n      },\n      {\n          \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n          \"size\": 197,\n          \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecfe7d
\"\n      },\n      {\n          \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n          \"size\": 154,\n          \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\"\n
      },\n      {\n          \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n          \"size\": 176,\n          \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\"\n      },\n      {\n          \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n          \"size\": 183,\n          \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\"\n
      },\n      {\n          \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n          \"size\": 212,\n          \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\"\n
      },\n      {\n          \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n          \"size\": 212,\n          \"digest\":
  \"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\"\n
      }
    ]
  },
  \"responseElements\": {
    \"image\": {
      \"repositoryName\": \"testrepo\",
      \"imageManifest\": \"{\\n  \"schemaVersion\": 2,\\n  \"mediaType\": \"application/
vnd.docker.distribution.manifest.v2+json\",,\n  \"config\": {\\n    \"mediaType\":
  \"application/vnd.docker.container.image.v1+json\",,\n    \"size\": 5543,\\n
  \"digest\": \"sha256:000b9b805af1cdb60628898c9f411996301a1c13afd3dbef1d8a16ac6dbf503a
\"\n  },\\n  \"layers\": [\\n    {\\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n      \"size\": 43252507,\\n
      \"digest\": \"sha256:3b37166ec61459e76e33282dda08f2a9cd698ca7e3d6bc44e6a6e7580cdeff8e
\"\n    },\\n    {\\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n      \"size\": 846,\\n      \"digest
\": \"sha256:504facff238fde83f1ca8f9f54520b4219c5b8f80be9616ddc52d31448a044bd
\"\n    },\\n    {\\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n      \"size\": 615,\\n      \"digest
\": \"sha256:ebbcacd28e101968415b0c812b2d2dc60f969e36b0b08c073bf796e12b1bb449\"\n
    },\\n    {\\n      \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\",,\n      \"size\": 850,\\n      \"digest

```



```

\": \"sha256:c7fb3351ecad291a88b92b600037e2435c84a347683d540042086fe72c902b8a
\n      },\n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 168, \n        \"digest\":
\"sha256:2e3debadcbf7e542e2aefbce1b64a358b1931fb403b3e4aeca27cb4d809d56c2\" \n      },
\n      {\n        \"mediaType\": \"application/vnd.docker.image.rootfs.diff.tar.gzip
\", \n        \"size\": 37720774, \n        \"digest\":
\"sha256:f8c9f51ad524d8ae9bf4db69cd3e720ba92373ec265f5c390ffb21bb0c277941\" \n
      }, \n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 30432107, \n
        \"digest\": \"sha256:813a50b13f61cf1f8d25f19fa96ad3aa5b552896c83e86ce413b48b091d7f01b
\" \n      }, \n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 197, \n        \"digest
\": \"sha256:7ab043301a6187ea3293d80b30ba06c7bf1a0c3cd4c43d10353b31bc0cecf7d
\" \n      }, \n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 154, \n        \"digest
\": \"sha256:67012cca8f31dc3b8ee2305e7762fee20c250513effdedb38a1c37784a5a2e71\" \n
      }, \n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 176, \n        \"digest
\": \"sha256:3bc892145603fffc9b1c97c94e2985b4cb19ca508750b15845a5d97becbd1a0e
\" \n      }, \n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 183, \n        \"digest
\": \"sha256:6f1c79518f18251d35977e7e46bfa6c6b9cf50df2a79d4194941d95c54258d18\" \n
      }, \n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 212, \n        \"digest
\": \"sha256:b7bcfbc2e2888afebede4dd1cd5eebf029bb6315feeaf0b56e425e11a50afe42\" \n
      }, \n      {\n        \"mediaType\": \"application/
vnd.docker.image.rootfs.diff.tar.gzip\", \n        \"size\": 212, \n        \"digest\":
\"sha256:2b220f8b0f32b7c2ed8eaafe1c802633bbd94849b9ab73926f0ba46cdae91629\" \n      } \n
    ] \n  },
  \"registryId\": \"123456789012\",
  \"imageId\": {
    \"imageDigest\":
\"sha256:98c8b060c21d9adbb6b8c41b916e95e6307102786973ab93a41e8b86d1fc6d3e\",
    \"imageTag\": \"latest\"
  }
},
\"requestID\": \"cf044b7d-5f9d-11e9-9b2a-95983139cc57\",
\"eventID\": \"2bfd4ee2-2178-4a82-a27d-b12939923f0f\",
\"resources\": [{
  \"ARN\": \"arn:aws:ecr:us-east-2:123456789012:repository/testrepo\",
  \"accountId\": \"123456789012\"
}],
\"eventType\": \"AwsApiCall\",

```

```
"recipientAccountId": "123456789012"
}
```

예제: 이미지 가져오기 작업

다음 예제는 해당 작업을 사용하는 이미지 가져오기를 보여주는 CloudTrail 로그 항목을 보여줍니다.

BatchGetImage

#### Note

이미지를 가져올 때 아직 로컬에 이미지가 없는 경우 로그에 `GetDownloadUrlForLayer` CloudTrail 참조도 표시됩니다.

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:account_name",
    "arn": "arn:aws:sts::123456789012:user/Mary_Major",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "Mary_Major",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-04-15T16:42:14Z"
      }
    }
  },
  "eventTime": "2019-04-15T17:23:20Z",
  "eventSource": "ecr.amazonaws.com",
  "eventName": "BatchGetImage",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "203.0.113.12",
  "userAgent": "console.amazonaws.com",
  "requestParameters": {
    "imageIds": [{
      "imageTag": "latest"
    }],
    "acceptedMediaTypes": [
      "application/json",
```

```

"application/vnd.oci.image.manifest.v1+json",
"application/vnd.oci.image.index.v1+json",
"application/vnd.docker.distribution.manifest.v2+json",
"application/vnd.docker.distribution.manifest.list.v2+json",
"application/vnd.docker.distribution.manifest.v1+prettyjws"
],
"repositoryName": "testrepo",
"registryId": "123456789012"
},
"responseElements": null,
"requestID": "2a1b97ee-5fa3-11e9-a8cd-cd2391aeda93",
"eventID": "c84f5880-c2f9-4585-9757-28fa5c1065df",
"resources": [{
  "ARN": "arn:aws:ecr:us-east-2:123456789012:repository/testrepo",
  "accountId": "123456789012"
}],
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

### 예제: 이미지 수명 주기 정책 작업

다음 예제는 수명 주기 정책 규칙에 따라 이미지가 만료되는 시기를 보여주는 CloudTrail 로그 항목을 보여줍니다. 이 이벤트 유형은 이벤트 이름 필드에서 PolicyExecutionEvent를 필터링하여 찾을 수 있습니다.

#### Important

개별 CloudTrail 이벤트의 크기 제한으로 인해 10개 이상의 이미지가 만료된 수명 주기 정책 작업의 경우 Amazon ECR은 여러 이벤트를 에 전송합니다. CloudTrail 또한 Amazon ECR에서는 이미지당 최대 100개의 태그를 추가할 수 있습니다.

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "accountId": "123456789012",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2020-03-12T20:22:12Z",
  "eventSource": "ecr.amazonaws.com",
  "eventName": "PolicyExecutionEvent",

```

```
"awsRegion": "us-west-2",
"sourceIPAddress": "AWS Internal",
"userAgent": "AWS Internal",
"requestParameters": null,
"responseElements": null,
"eventID": "9354dd7f-9aac-4e9d-956d-12561a4923aa",
"readOnly": true,
"resources": [
  {
    "ARN": "arn:aws:ecr:us-west-2:123456789012:repository/testrepo",
    "accountId": "123456789012",
    "type": "AWS::ECR::Repository"
  }
],
"eventType": "AwsServiceEvent",
"recipientAccountId": "123456789012",
"serviceEventDetails": {
  "repositoryName": "testrepo",
  "lifecycleEventPolicy": {
    "lifecycleEventRules": [
      {
        "rulePriority": 1,
        "description": "remove all images > 2",
        "lifecycleEventSelection": {
          "tagStatus": "Any",
          "tagPrefixList": [],
          "countType": "Image count more than",
          "countNumber": 2
        },
        "action": "expire"
      }
    ],
    "lastEvaluatedAt": 0,
    "policyVersion": 1,
    "policyId": "ceb86829-58e7-9498-920c-aa042e33037b"
  },
  "lifecycleEventImageActions": [
    {
      "lifecycleEventImage": {
        "digest":
"sha256:ddba4d27a7ffc3f86dd6c2f92041af252a1f23a8e742c90e6e1297bfa1bc0c45",
        "tagStatus": "Tagged",
        "tagList": [
          "alpine"
        ]
      }
    ]
  ]
}
```

```
    ],
    "pushedAt": 1584042813000
  },
  "rulePriority": 1
},
{
  "lifecycleEventImage": {
    "digest":
"sha256:6ab380c5a5acf71c1b6660d645d2cd79cc8ce91b38e0352cbf9561e050427baf",
    "tagStatus": "Tagged",
    "tagList": [
      "centos"
    ],
    "pushedAt": 1584042842000
  },
  "rulePriority": 1
}
]
}
}
```

## SDK와 함께 아마존 AWS ECR 사용

AWS 소프트웨어 개발 키트 (SDK) 는 널리 사용되는 여러 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예시
<a href="#">AWS SDK for C++</a>	<a href="#">AWS SDK for C++ 코드 예제</a>
<a href="#">AWS CLI</a>	<a href="#">AWS CLI 코드 예제</a>
<a href="#">AWS SDK for Go</a>	<a href="#">AWS SDK for Go 코드 예제</a>
<a href="#">AWS SDK for Java</a>	<a href="#">AWS SDK for Java 코드 예제</a>
<a href="#">AWS SDK for JavaScript</a>	<a href="#">AWS SDK for JavaScript 코드 예제</a>
<a href="#">AWS SDK for Kotlin</a>	<a href="#">AWS SDK for Kotlin 코드 예제</a>
<a href="#">AWS SDK for .NET</a>	<a href="#">AWS SDK for .NET 코드 예제</a>
<a href="#">AWS SDK for PHP</a>	<a href="#">AWS SDK for PHP 코드 예제</a>
<a href="#">AWS Tools for PowerShell</a>	<a href="#">PowerShell 코드 예제를 위한 도구</a>
<a href="#">AWS SDK for Python (Boto3)</a>	<a href="#">AWS SDK for Python (Boto3) 코드 예제</a>
<a href="#">AWS SDK for Ruby</a>	<a href="#">AWS SDK for Ruby 코드 예제</a>
<a href="#">AWS SDK for Rust</a>	<a href="#">AWS SDK for Rust 코드 예제</a>
<a href="#">AWS SDK for SAP ABAP</a>	<a href="#">AWS SDK for SAP ABAP 코드 예제</a>
<a href="#">AWS SDK for Swift</a>	<a href="#">AWS SDK for Swift 코드 예제</a>

**i** 가용성 예제

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

# SDK를 사용하는 AWS Amazon ECR의 코드 예제

다음 코드 예제는 AWS 소프트웨어 개발 키트 (SDK) 와 함께 Amazon ECR을 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 을 참조하십시오. [SDK와 함께 아마존 AWS ECR 사용](#) 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

## 코드 예시

- [SDK를 사용한 AWS Amazon ECR에서의 작업](#)
  - [AWS SDK 또는 DescribeRepositories CLI와 함께 사용](#)
  - [AWS SDK 또는 ListImages CLI와 함께 사용](#)

## SDK를 사용한 AWS Amazon ECR에서의 작업

다음 코드 예제는 AWS SDK를 사용하여 개별 Amazon ECR 작업을 수행하는 방법을 보여줍니다. 이 발췌문은 Amazon ECR API를 호출하며 컨텍스트에서 실행해야 하는 대규모 프로그램에서 발췌한 코드입니다. 각 예제에는 코드 설정 GitHub 및 실행 지침을 찾을 수 있는 링크가 포함되어 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 [Amazon Elastic 컨테이너 레지스트리 \(Amazon ECR\) API 참조](#)를 참조하십시오.

## 예제

- [AWS SDK 또는 DescribeRepositories CLI와 함께 사용](#)
- [AWS SDK 또는 ListImages CLI와 함께 사용](#)

## AWS SDK 또는 **DescribeRepositories** CLI와 함께 사용

다음 코드 예제는 DescribeRepositories의 사용 방법을 보여줍니다.



## CLI

### AWS CLI

레지스트리의 리포지토리를 설명하는 방법

이 예시에서는 계정의 기본 레지스트리에 있는 리포지토리를 설명합니다.

명령:

```
aws ecr describe-repositories
```

출력:

```
{
  "repositories": [
    {
      "registryId": "012345678910",
      "repositoryName": "ubuntu",
      "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/
ubuntu"
    },
    {
      "registryId": "012345678910",
      "repositoryName": "test",
      "repositoryArn": "arn:aws:ecr:us-west-2:012345678910:repository/test"
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 [DescribeRepositories](#) 참조를 참조하십시오.

## Rust

### SDK for Rust

#### Note

자세한 내용은 에서 확인할 수 GitHub 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

async fn show_repos(client: &aws_sdk_ecr::Client) -> Result<(),
aws_sdk_ecr::Error> {
    let rsp = client.describe_repositories().send().await?;

    let repos = rsp.repositories();

    println!("Found {} repositories:", repos.len());

    for repo in repos {
        println!("  ARN: {}", repo.repository_arn().unwrap());
        println!("  Name: {}", repo.repository_name().unwrap());
    }

    Ok(())
}

```

- API에 대한 자세한 내용은 Rust용 AWS SDK API 레퍼런스를 참조하십시오 [DescribeRepositories](#).

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. [SDK와 함께 아마존 AWS ECR 사용](#) 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

## AWS SDK 또는 **ListImages** CLI와 함께 사용

다음 코드 예제는 ListImages의 사용 방법을 보여줍니다.

CLI

AWS CLI

리포지토리의 이미지를 나열하는 방법

다음 list-images 예시에서는 cluster-autoscaler 리포지토리의 이미지 목록을 표시합니다.

```

aws ecr list-images \
  --repository-name cluster-autoscaler

```

출력:

```
{
  "imageIds": [
    {
      "imageDigest":
"sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
      "imageTag": "v1.13.8"
    },
    {
      "imageDigest":
"sha256:99c6fb4377e9a420a1eb3b410a951c9f464eff3b7dbc76c65e434e39b94b6570",
      "imageTag": "v1.13.7"
    },
    {
      "imageDigest":
"sha256:4a1c6567c38904384ebc64e35b7eeddd8451110c299e3368d2210066487d97e5",
      "imageTag": "v1.13.6"
    }
  ]
}
```

- API 세부 정보는 AWS CLI 명령 [ListImages](#)참조를 참조하십시오.

## Rust

### SDK for Rust

#### Note

자세한 내용은 에서 확인할 수 GitHub 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_images(
    client: &aws_sdk_ecr::Client,
    repository: &str,
) -> Result<(), aws_sdk_ecr::Error> {
    let rsp = client
        .list_images()
        .repository_name(repository)
        .send()
        .await?;
```

```
let images = rsp.image_ids();

println!("found {} images", images.len());

for image in images {
    println!(
        "image: {}:{}",
        image.image_tag().unwrap(),
        image.image_digest().unwrap()
    );
}

Ok(())
}
```

- API에 대한 자세한 내용은 Rust용AWS SDK API 레퍼런스를 참조하십시오 [ListImages](#).

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. [SDK와 함께 아마존 AWS ECR 사용](#) 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

## Amazon ECR 서비스 할당량

다음 표에서는 Amazon Elastic Container Registry(Amazon ECR)의 기본 서비스 할당량을 제공합니다.

명칭	기본값	조정 가능	Description
복제 구성의 규칙당 필터 수	지원되는 각 리전: 100	아니요	복제 구성의 최대 규칙당 필터 수입니다.
리포지토리당 이미지 수	지원되는 각 리전: 10,000개	<a href="#">예</a>	리포지토리당 사용 가능한 이미지의 최대 개수입니다.
계층 파트	지원되는 리전별: 4,200개	아니요	계층 파트의 최대 개수입니다. 이것은 Amazon ECR API 작업을 직접 사용하여 이미지 푸시 작업에 대한 멀티파트 업로드를 시작하는 경우에만 적용됩니다.
수명 주기 정책 길이	지원되는 리전별: 30,720개	아니요	수명 주기 정책에 사용할 수 있는 문자의 최대 개수입니다.
최대 계층 파트 크기	지원되는 각 리전: 10	아니요	계층 파트의 최대 크기 (MiB)입니다. 이것은 Amazon ECR API 작업을 직접 사용하여 이미지 푸시 작업에 대한 멀티파트 업로드를 시작하는 경우에만 적용됩니다.

명칭	기본값	조정 가능	Description
최대 계층 크기	각각 지원되는 리전: 52,000	아니요	계층의 최대 크기(MiB)입니다.
최소 계층 파트 크기	지원되는 각 리전: 5	아니요	계층 파트의 최소 크기(MiB)입니다. 이것은 Amazon ECR API 작업을 직접 사용하여 이미지 푸시 작업에 대한 멀티파트 업로드를 시작하는 경우에만 적용됩니다.
레지스트리당 폴스루 캐시 규칙	지원되는 각 리전: 50	아니요	폴스루 캐시 규칙의 최대 수입입니다.
BatchCheckLayerAvailability 요청 비율	지원되는 각 리전: 초당 1,000	<a href="#">예</a>	현재 리전에서 초당 생성할 수 있는 BatchCheckLayerAvailability 요청의 최대 수입입니다. 이미지가 리포지토리에 푸시되면 각 이미지 계층이 검사되어 이전에 업로드되었는지 확인합니다. 업로드된 경우 이미지 계층을 건너뛰니다.

명칭	기본값	조정 가능	Description
BatchGetImage 요청 비율	지원되는 리전별: 초당 2,000개	<a href="#">예</a>	현재 리전에서 초당 수행할 수 있는 BatchGetImage 요청의 최대 수입입니다. 이미지를 가져오면 BatchGetImage API가 한 번 호출되어 이미지 매니페스트를 검색합니다. 이 API에 대한 할당량 증가를 요청하는 경우 GetDownloadUriForLayer 사용량도 검토해야 합니다.
CompleteLayerUpload 요청 비율	지원되는 리전별: 초당 100개	<a href="#">예</a>	현재 리전에서 초당 생성할 수 있는 CompleteLayerUpload 요청의 최대 수입입니다. 이미지가 푸시되면 CompleteLayerUpload API가 각 새 이미지 계층당 한 번 호출되어 업로드가 완료되었는지 확인합니다.
GetAuthorizationToken 요청 비율	지원되는 리전별: 초당 500개	<a href="#">예</a>	현재 리전에서 초당 수행할 수 있는 GetAuthorizationToken 요청의 최대 수입입니다.

명칭	기본값	조정 가능	Description
GetDownloadUrlForLayer 요청 비율	지원되는 리전별: 초당 3,000개	<a href="#">예</a>	현재 리전에서 초당 수행할 수 있는 GetDownloadUrlForLayer 요청의 최대 수입입니다. 이미지를 가져오면 GetDownloadUrlForLayer API가 아직 캐시되지 않은 이미지 계층당 한 번 호출됩니다. 이 API에 대한 할당량 증가를 요청하는 경우 BatchGetImage 사용량도 검토해야 합니다.
InitiateLayerUpload 요청 비율	지원되는 리전별: 초당 100개	<a href="#">예</a>	현재 리전에서 초당 생성할 수 있는 InitiateLayerUpload 요청의 최대 수입입니다. 이미지가 푸시되면 InitiateLayerUpload API는 아직 업로드되지 않은 이미지 계층당 한 번 호출됩니다. 이미지 계층이 업로드되었는지 여부는 BatchCheckLayerAvailability API 작업에 의해 결정됩니다.



명칭	기본값	조정 가능	Description
PutImage 요청 비율	지원되는 각 리전: 초당 10개	<a href="#">예</a>	현재 리전에서 초당 생성할 수 있는 PutImage 요청의 최대 수입입니다. 이미지가 푸시되고 모든 새 이미지 계층이 업로드되면 PutImage API가 한 번 호출되어 이미지 매니페스트 및 이미지와 관련된 태그를 생성하거나 업데이트합니다.
UploadLayerPart 요청 비율	지원되는 리전별: 초당 500개	<a href="#">예</a>	현재 리전에서 초당 생성할 수 있는 UploadLayerPart 요청의 최대 수입입니다. 이미지가 푸시되면 각 새 이미지 계층이 부분적으로 업로드되고 UploadLayerPart API가 각 새 이미지 계층 부분에 대해 한 번씩 호출됩니다.
이미지 스캔 비율	지원되는 리전별: 1개	아니요	이미지당 24시간 동안 스캔하는 최대 이미지 수입입니다.
등록된 리포지토리	지원되는 각 리전: 10,000개	<a href="#">예</a>	현재 리전의 이 계정에서 생성할 수 있는 리포지토리의 최대 수.
수명 주기 정책당 규칙	지원되는 각 리전: 50	아니요	수명 주기 정책에 사용할 수 있는 규칙의 최대 개수입니다.

명칭	기본값	조정 가능	Description
복제 구성당 규칙 수	지원되는 각 리전: 10개	아니요	복제 구성에 포함할 수 있는 최대 규칙 수입니다.
이미지당 태그 수	지원되는 각 리전: 1,000개	아니요	이미지당 태그의 최대 개수입니다.
복제 구성의 모든 규칙에 대한 고유한 대상 수	지원되는 각 리전: 25개	아니요	복제 구성의 모든 규칙에 대한 고유한 대상의 최대 수입니다.

## AWS Management Console에서 Amazon ECR 서비스 할당량 관리

Amazon ECR은 중앙 위치에서 할당량을 보고 관리할 수 있는 AWS 서비스인 Service Quotas와 통합됩니다. 자세한 내용은 Service Quotas 사용 설명서의 [Service Quotas라는?](#)을 참조하세요.

Service Quotas를 사용하면 모든 Amazon ECR 서비스 할당량의 값을 쉽게 찾을 수 있습니다.

### Amazon ECR 서비스 할당량 보기(AWS Management Console)

1. <https://console.aws.amazon.com/servicequotas/>에서 Service Quotas 콘솔을 엽니다.
2. 탐색 창에서 AWS 서비스를 선택합니다.
3. AWS서비스목록에서 Amazon Elastic Container Registry(Amazon ECR)을 검색하고 선택합니다.

서비스 할당량 목록에서 서비스 할당량 이름, 적용된 값(제공된 경우), AWS 기본 할당량 및 할당량 값 조정 가능 여부를 확인할 수 있습니다.

4. 설명 등 서비스 할당량에 대한 추가 정보를 보려면 할당량 이름을 선택합니다.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 [할당량 증가 요청](#)을 참조하세요.

## API 사용량 지표를 모니터링하기 위한 CloudWatch 경보 생성

Amazon ECR 은 레지스트리 인증, 이미지 푸시 및 이미지 가져오기 작업과 관련된 각 API에 대한 AWS 서비스 할당량에 해당하는 CloudWatch 사용 지표를 제공합니다. Service Quotas 콘솔에서 그래프에 사용량을 시각화하고 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수 있습니다. 자세한 내용은 [Amazon ECR 사용량 지표](#)을(를) 참조하세요.

다음 단계를 사용하여 Amazon ECR API 사용량 지표 중 하나를 기반으로 CloudWatch 경보를 생성합니다.

Amazon ECR 사용량 할당량(AWS Management Console)을 기반으로 경보를 생성하려면

1. <https://console.aws.amazon.com/servicequotas/>에서 Service Quotas 콘솔을 엽니다.
2. 탐색 창에서 AWS 서비스를 선택합니다.
3. AWS서비스목록에서 Amazon Elastic Container Registry(Amazon ECR)을 검색하고 선택합니다.
4. Service Quotas 목록에서 경보를 만들려는 Amazon ECR 사용량 할당량을 선택합니다.
5. Amazon CloudWatch Events 경보 섹션에서 생성을 선택합니다.
6. 경보 임계값(Alarm threshold)에서 경보 값으로 설정할 적용된 할당량 값의 백분율을 선택합니다.
7. 경보 이름(Alarm name)에서 경보 이름을 입력한 다음 생성(Create)을 선택합니다.

# Amazon ECR 문제 해결

이 장은 Amazon ECR에 대한 진단 정보를 찾는 데 도움이 되며 일반적인 문제 및 오류 메시지에 대한 문제 해결 단계를 제공합니다.

주제

- [Amazon ECR 사용 시 Docker 명령 및 문제 해결](#)
- [Amazon ECR 오류 메시지 문제 해결](#)

## Amazon ECR 사용 시 Docker 명령 및 문제 해결

경우에 따라 Amazon ECR에 대해 Docker 명령을 실행하면 오류 메시지가 표시될 수 있습니다. 몇 가지 일반적인 오류 메시지 및 잠재적인 해결 방안이 아래 설명되어 있습니다.

주제

- [Docker 로그에는 예상 오류 메시지가 포함되지 않습니다.](#)
- [Amazon ECR 리포지토리로부터 이미지를 가져올 때 오류: "Filesystem Verification Failed" 또는 "404: Image Not Found"](#)
- [Amazon ECR에서 이미지를 가져올 때 오류: "Filesystem Layer Verification Failed" 발생](#)
- [리포지토리에 푸시할 때 HTTP 403 오류 또는 "no basic auth credentials" 오류 발생](#)

Docker 로그에는 예상 오류 메시지가 포함되지 않습니다.

Docker 관련 문제의 디버깅을 시작하려면 먼저 호스트 인스턴스에서 실행되는 Docker 데몬에서 Docker 디버깅 출력을 켜십시오. Amazon ECS 컨테이너 인스턴스에서 Amazon ECR에서 가져온 이미지를 사용하는 경우 Amazon Elastic 컨테이너 서비스 개발자 안내서의 [Docker 데몬에서 자세한 출력 구성을](#) 참조하십시오.

## Amazon ECR 리포지토리로부터 이미지를 가져올 때 오류: "Filesystem Verification Failed" 또는 "404: Image Not Found"

Docker 1.9 이상을 사용하여 docker pull 명령을 사용하여 Amazon ECR 리포지토리에서 이미지를 가져올 때 Filesystem verification failed 오류가 발생할 수 있습니다. Docker 버전 1.9 이하를 사용할 경우에는 404: Image not found 오류가 표시될 수 있습니다.

몇 가지 가능한 원인 및 관련 설명이 아래 나와 있습니다.

### 로컬 디스크 가득 참

docker pull을 실행하고 있는 로컬 디스크가 가득 찬 경우, 로컬 파일에서 계산된 SHA-1 해시가 Amazon ECR에서 계산한 것과 다를 수 있습니다. 가져오려는 도커 이미지를 저장할 만한 여유 공간이 로컬 디스크에 충분히 남아 있는지 확인하십시오. 오래된 이미지를 삭제하여 새로운 이미지를 위한 공간을 마련할 수도 있습니다. docker images 명령을 사용하여 로컬에 다운로드한 모든 도커 이미지 목록을 크기와 함께 표시합니다.

### 네트워크 오류로 인해 클라이언트가 원격 리포지토리에 연결할 수 없음

Amazon ECR 리포지토리를 호출하려면 인터넷에 연결되어야 합니다. 네트워크 설정을 확인하고, 다른 도구 및 애플리케이션이 인터넷의 리소스에 액세스할 수 있는지 확인하십시오. 프라이빗 서브넷의 Amazon EC2 인스턴스에서 docker pull을 실행하는 경우 서브넷에 인터넷에 대한 라우팅이 있는지 확인하세요. Network Address Translation(NAT) 서버 또는 관리형 NAT 게이트웨이를 사용하십시오.

현재 Amazon ECR 리포지토리를 호출하려면 회사 방화벽을 통해 Amazon Simple Storage Service(Amazon S3)로의 네트워크 액세스가 필요합니다. 조직에서 서비스 엔드포인트를 허용하는 방화벽 소프트웨어나 NAT 디바이스를 사용하는 경우, 현재 리전의 Amazon S3 서비스 엔드포인트를 허용해야 합니다.

HTTP 프록시 뒤로 Docker를 사용하고 있는 경우, 적절한 프록시 설정을 사용하여 Docker를 구성할 수 있습니다. 자세한 내용은 Docker 설명서의 [HTTP 프록시](#)를 참조하십시오.

## Amazon ECR에서 이미지를 가져올 때 오류: "Filesystem Layer Verification Failed" 발생

docker pull 명령을 사용하여 이미지를 가져올 때 image image-name not found 오류가 표시될 수 있습니다. Docker 로그를 살펴보면 다음과 같은 오류가 있을 수 있습니다.

```
filesystem layer verification failed for digest sha256:2b96f...
```

이 오류는 이미지의 하나 이상의 계층을 다운로드하지 못했음을 나타냅니다. 몇 가지 가능한 원인 및 관련 설명이 아래 나와 있습니다.

## 이전 버전의 Docker를 사용하고 있음

이 오류는 1.10 버전 이전의 Docker 버전을 사용하고 있을 때 적은 비율로 발생할 수 있습니다. Docker 클라이언트를 1.10 이상으로 업그레이드하십시오.

### 클라이언트에 네트워크 또는 디스크 오류 발생

앞의 Filesystem verification failed 메시지의 설명대로, 디스크가 가득 찼거나 네트워크 문제가 있는 경우 하나 이상의 계층이 다운로드되지 않을 수 있습니다. 위의 권장 사항을 따라 파일 시스템이 가득 차지 않았는지와 네트워크 내에서 Amazon S3에 대한 액세스를 활성화했는지 확인하십시오.

## 리포지토리에 푸시할 때 HTTP 403 오류 또는 "no basic auth credentials" 오류 발생

aws ecr get-login-password 명령을 사용하여 Docker에 대해 성공적으로 인증을 한 경우에도 docker push 또는 docker pull 명령을 실행하면 HTTP 403 (Forbidden) 오류 또는 no basic auth credentials 오류 메시지가 표시되는 경우가 있습니다. 다음은 이러한 문제의 알려진 원인 몇 가지입니다.

### 다른 리전에 대해 인증 받음

인증 요청은 특정 리전으로 묶여 있으며 그 외 리전에 사용할 수 없습니다. 예를 들어, 미국 서부(오레곤)로부터 권한 부여 토큰을 받은 경우, 미국 동부(버지니아 북부)의 리포지토리에 대해 인증 받는데 사용할 수 없습니다. 이 문제를 해결하려면 리포지토리가 있는 리전과 동일한 리전에서 인증 토큰을 검색했는지 확인합니다. 자세한 정보는 [the section called "레지스트리 인증"](#)을 참조하세요.

권한이 없는 리포지토리로 푸시하도록 인증했습니다.

리포지토리로 푸시하는 데 필요한 권한이 없습니다. 자세한 정보는 [Amazon ECR의 프라이빗 리포지토리 정책](#)을 참조하세요.

### 토큰이 만료됨

GetAuthorizationToken 작업을 사용하여 받은 토큰의 기본 권한 부여 토큰 만료 기간은 12시간입니다.

### wincred 자격 증명 관리자의 버그

Windows용 Docker의 일부 버전에서는 wincred라는 자격 증명 관리자를 사용하는데, 이는 aws ecr get-login-password에서 생성하는 Docker 로그인 명령을 올바르게 처리하지 않습니다(자세한 내용은 <https://github.com/docker/docker/issues/22910> 참조). 출력인 Docker 로그인 명령을 실행할

수 있지만, 이미지를 푸시하거나 가져오려고 시도하면 해당 명령이 실패합니다. 이 버그는 `aws ecr get-login-password`의 출력인 Docker 로그인 명령에서 레지스트리 인수의 `https://` 스키마를 제거하면 해결할 수 있습니다. HTTPS 스키마가 없는 예제 Docker 로그인 명령은 아래와 같습니다.

```
docker login -u AWS -p <password> <aws_account_id>.dkr.ecr.<region>.amazonaws.com
```

## Amazon ECR 오류 메시지 문제 해결

Amazon ECR 콘솔을 통해 시작한 API 호출이 AWS CLI 종료되거나 오류 메시지가 표시되는 경우가 있습니다. 몇 가지 일반적인 오류 메시지 및 잠재적인 해결 방안이 아래 설명되어 있습니다.

### HTTP 429: 요청이 너무 많거나 ThrottlingException

하나 이상의 Amazon ECR 작업 또는 API 호출에서 `ThrottlingException` 오류 또는 오류를 받을 수 있습니다. 429: Too Many Requests 이는 짧은 간격으로 Amazon ECR에서 단일 엔드포인트를 반복적으로 호출하고 있으며, 요청에 병목 현상이 발생하고 있음을 나타냅니다. 병목 현상은 단일 사용자로부터 단일 엔드포인트로의 호출이 일정 기간 동안 특정 임계값을 초과할 때 발생합니다.

Amazon ECR의 각 API 작업에는 이와 관련된 속도 제한이 있습니다. 예를 들어

[GetAuthorizationToken](#) 작업의 제한은 20TPS(초당 트랜잭션)이며 200TPS까지 확장 가능합니다. 각 리전의 각 계정에는 최대 200개의 `GetAuthorizationToken` 크레딧을 저장할 수 있는 버킷이 생성됩니다. 이러한 크레딧은 초당 20의 속도로 보충됩니다. 버킷에 200개의 크레딧이 있으면 1초 동안 초당 200개의 `GetAuthorizationToken` API 트랜잭션을 수행한 다음 초당 20개의 트랜잭션을 무기한 유지할 수 있습니다. Amazon ECR API의 속도 제한에 대한 자세한 내용은 을 참조하십시오.

#### [Amazon ECR 서비스 할당량](#)

병목 현상 오류를 해결하려면 증분 백오프가 있는 재시도 함수를 코드에 구현하십시오. 자세한 내용은 AWS SDK 및 [도구 참조 안내서의 재시도 동작](#)을 참조하십시오. 또 다른 옵션은 Service Quotas 콘솔을 사용하여 속도 한도 증가를 요청하는 것입니다. 자세한 내용은 을 참조하십시오 [AWS Management Console에서 Amazon ECR 서비스 할당량 관리](#).

### HTTP 403: "User [arn] is not authorized to perform [operation]"

Amazon ECR로 작업을 수행하려고 시도할 때 다음 오류가 표시될 수 있습니다.

```
$ aws ecr get-login-password
A client error (AccessDeniedException) occurred when calling the GetAuthorizationToken operation:
```

```
User: arn:aws:iam::account-number:user/username is not authorized to perform:  
ecr:GetAuthorizationToken on resource: *
```

이는 사용자에게 Amazon ECR를 사용할 권한이 부여되지 않았음을 나타내거나 해당 권한이 올바르게 설정되지 않았음을 나타냅니다. 특히, Amazon ECR 리포지토리에 대해 작업을 수행하고 있는 경우, 사용자에게 해당 리포지토리에 액세스할 수 있는 권한이 부여되었는지 확인하십시오. Amazon ECR에 대한 권한 생성 및 확인에 대한 자세한 내용은 [Amazon Elastic Container Registry용 Identity and Access Management](#) 단원을 참조하십시오.

## HTTP 404: "Repository Does Not Exist" 오류 발생

현재는 존재하지 않는 Docker Hub 리포지토리를 지정하는 경우 Docker Hub가 이를 자동으로 생성합니다. Amazon ECR에서는 먼저 새로운 리포지토리가 명시적으로 생성되어야 이를 사용할 수 있습니다. 이렇게 하면 새로운 리포지토리가 실수(예: 오타)로 생성되지 않으며, 또한 새로운 리포지토리에 적절한 보안 액세스 정책이 명시적으로 지정되도록 해줍니다. 리포지토리 생성에 대한 자세한 내용은 [Amazon ECR 프라이빗 리포지토리](#) 단원을 참조하십시오.

**오류: TTY가 아닌 장치에서 대화형 로그인을 수행할 수 없습니다.**

오류 `Cannot perform an interactive login from a non TTY device`이 발생하면 다음 문제 해결 단계가 도움이 될 것입니다.

- AWS CLI 버전 2를 사용하고 있고 시스템에 버전 1과 충돌하는 버전이 없는지 확인하십시오. AWS CLI 자세한 내용은 [최신 버전의 AWS CLI 설치 또는 업데이트](#)를 참조하세요.
- 유효한 자격 AWS CLI 증명으로 구성했는지 확인하십시오. 자세한 내용은 [최신 버전의 AWS CLI 설치 또는 업데이트](#)를 참조하세요.
- AWS CLI 명령 구문이 올바른지 확인하세요.



## 문서 기록

다음 표에서는 Amazon ECR의 최신 릴리스 이후 이 설명서에서 변경된 중요 사항에 대해 설명합니다. 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다.

변경 사항	설명	날짜
중국 지역에 지역 간 및 계정 간 복제가 추가되었습니다.	Amazon ECR은 복제되는 리포지토리를 필터링하기 위한 지원을 중국 지역에 추가했습니다.	2024년 5월 15일
캐시 규칙을 폴스루할 수 있는 GitLab 컨테이너 레지스트리를 추가했습니다.	Amazon ECR은 GitLab 컨테이너 레지스트리에 대한 폴스루 캐시 규칙 생성 지원을 추가했습니다.	2024년 5월 8일
와일드카드 사용에 대한 지원을 추가하기 위한 Amazon ECR 수명 주기 정책 업데이트	Amazon ECR은 수명 주기 정책 규칙의 tagPatternList 파라미터를 사용하여 수명 주기 정책에 와일드카드에 대한 지원을 추가했습니다. 자세한 정보는 <a href="#">Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.</a> 을 참조하세요.	2023년 12월 18일
Amazon ECR 리포지토리 생성 템플릿	Amazon ECR은 리포지토리 생성 템플릿에 대한 지원을 추가했습니다. 자세한 정보는 <a href="#">폴스루 캐시 작업 중에 생성된 리포지토리를 제어하기 위한 템플릿</a> 을 참조하세요.	2023년 11월 15일
인증된 업스트림 레지스트리를 지원하는 Amazon ECR 폴스루 캐시 추가	Amazon ECR은 폴스루 캐시 규칙에 인증이 필요한 업스트림 레지스트리 사용을 지원을 추가했습니다. 자세한 정보는 <a href="#">업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화</a> 을 참조하세요.	2023년 11월 15일
<a href="#">AWSECRPullThroughCache_ServiceRolePolicy</a> -기존 정책 업데이트	Amazon ECR에서 AWSECRPullThroughCache_ServiceRolePolicy 정책에 대한 새로운 권한을 추가했습니다. 이러한 권한을 통해 Amazon ECR은 Secrets Manager 보안 암호의 암호화된 콘텐츠를 검색할 수 있습니다. 이는 폴스루 캐시 규칙을 사용	2023년 11월 15일

변경 사항	설명	날짜
	하여 인증이 필요한 업스트림 레지스트리에서 이미지를 캐시할 때 필요합니다.	
Amazon ECR 이미지 서명	Amazon ECR과 AWS Signer Notary 클라이언트를 사용한 컨테이너 이미지 서명 생성 및 푸시에 대한 지원이 추가되었습니다. 자세한 정보는 <a href="#">Amazon ECR 사실 리포지토리에 저장된 이미지에 서명</a> 을 참조하세요.	2023년 6월 6일
풀스루 캐시 규칙에 Kubernetes 컨테이너 레지스트리 추가	Amazon ECR에 Kubernetes 컨테이너 레지스트리 관련 풀스루 캐시 규칙 생성에 대한 지원이 추가되었습니다. 자세한 정보는 <a href="#">업스트림 레지스트리를 Amazon ECR 프라이빗 레지스트리와 동기화</a> 을 참조하세요.	2023년 6월 1일
Amazon ECR 고급 스캔 기간 지원	Amazon Inspector에 고급 스캔이 활성화되었을 때 리포지토리가 모니터링되는 기간을 설정하는 기능에 대한 지원이 추가되었습니다. 자세한 정보는 <a href="#">Amazon Inspector에서 이미지의 향상된 스캔 지속 시간 변경</a> 을 참조하세요.	2022년 6월 28일
Amazon ECR이 리포지토리 풀 카운트 지표를 Amazon에 전송합니다. CloudWatch	Amazon ECR은 리포지토리 풀 카운트 지표를 CloudWatch Amazon에 전송합니다. 자세한 정보는 <a href="#">Amazon ECR 리포지토리 지표</a> 을 참조하세요.	2022년 1월 6일
확장 복제 지원	Amazon ECR에 복제되는 리포지토리 필터링에 대한 지원이 추가되었습니다. 자세한 내용은 <a href="#">Amazon ECR에서의 프라이빗 이미지 복제</a> 섹션을 참조하세요.	2021년 9월 21일
AWS Amazon ECR에 대한 관리형 정책	Amazon ECR은 AWS 관리형 정책에 대한 문서를 추가했습니다. 자세한 정보는 <a href="#">AWS Amazon Elastic 컨테이너 레지스트리에 대한 관리형 정책</a> 을 참조하세요.	2021년 6월 24일
교차 리전 및 교차 계정 복제	Amazon ECR에 프라이빗 레지스트리에 대한 복제 설정 구성에 대한 지원이 추가되었습니다. 자세한 정보는 <a href="#">Amazon ECR의 프라이빗 레지스트리 설정</a> 을 참조하세요.	2020년 12월 8일

변경 사항	설명	날짜
OCI 아티팩트 지원	<p>Amazon ECR에 Open Container Initiative(OCI) 아티팩트를 푸시 및 풀하기 위한 지원이 추가되었습니다. 새 파라미터 <code>artifactMediaType</code> 이(가) 아티팩트의 유형을 나타내기 위해 <code>DescribeImages</code> API 응답에 추가되었습니다.</p> <p>자세한 정보는 <a href="#">헬름 차트를 Amazon ECR 프라이빗 리포지토리로 푸시하기</a>을 참조하세요.</p>	2020년 8월 24일
저장 중 암호화	<p>Amazon ECR에 AWS Key Management Service 에 저장된 고객 관리형 키로 서버 측 암호화를 사용하여 리포지토리에 대한 암호화를 구성할 수 있는 지원이 추가되었습니다(AWS KMS).</p> <p>자세한 정보는 <a href="#">저장된 데이터 암호화</a>을 참조하세요.</p>	2020년 7월 29일
다중 아키텍처 이미지	<p>Amazon ECR에 다중 아키텍처 이미지에 사용되는 Docker 매니페스트 목록을 생성하고 푸시하는 지원이 추가되었습니다.</p> <p>자세한 정보는 <a href="#">다중 아키텍처 이미지를 Amazon ECR 사설 리포지토리로 푸시하기</a>을 참조하세요.</p>	2020년 4월 28일
Amazon ECR 사용량 지표	<p>Amazon ECR은 계정의 리소스 CloudWatch 사용에 대한 가시성을 제공하는 사용량 지표를 추가했습니다. 또한 사용량이 적용된 서비스 할당량에 가까워지면 알림을 받을 수 있도록 CloudWatch 및 Service Quotas 콘솔 모두에서 CloudWatch 경보를 생성할 수 있습니다.</p> <p>자세한 정보는 <a href="#">Amazon ECR 사용량 지표</a>을 참조하세요.</p>	2020년 2월 28일
Amazon ECR Service Quotas 업데이트됨	<p>API별 할당량을 포함하도록 Amazon ECR Service Quotas를 업데이트했습니다.</p> <p>자세한 정보는 <a href="#">Amazon ECR 서비스 할당량</a>을 참조하세요.</p>	2020년 2월 19일

변경 사항	설명	날짜
추가된 <code>get-login-password</code> 명령	<p>권한 부여 토큰을 검색하는 간단하고 안전한 방법을 제공하는 <code>get-login-password</code>에 대한 지원이 추가되었습니다.</p> <p>자세한 정보는 <a href="#">권한 부여 토큰 사용</a>을 참조하세요.</p>	2020년 2월 4일
이미지 스캔	<p>컨테이너 이미지의 소프트웨어 취약성을 식별하는 데 도움이 되는 이미지 스캔에 대한 지원이 추가되었습니다. Amazon ECR은 오픈 소스 CoreOS Clair 프로젝트의 CVE(일반적인 취약성 및 노출) 데이터베이스를 사용하고 스캔 결과 목록을 제공합니다.</p> <p>자세한 정보는 <a href="#">이미지를 스캔하여 Amazon ECR의 소프트웨어 취약성을 확인합니다.</a>을 참조하세요.</p>	2019년 10월 24일
VPC 엔드포인트 정책	<p>Amazon ECR 인터페이스 VPC 엔드포인트에서 IAM 정책 설정에 대한 지원이 추가되었습니다.</p> <p>자세한 정보는 <a href="#">Amazon ECR VPC 엔드포인트의 엔드포인트 정책 생성</a>을 참조하세요.</p>	2019년 9월 26일
이미지 태그 변경 가능성	<p>이미지 태그를 덮어쓰지 않도록 리포지토리를 변경 불가능하게 구성하기 위한 지원이 추가되었습니다.</p> <p>자세한 정보는 <a href="#">Amazon ECR에서 이미지 태그의 덮어쓰기 방지</a>을 참조하세요.</p>	2019년 7월 25일
인터페이스 VPC 엔드포인트(AWS PrivateLink)	<p>에 의해 구동되는 인터페이스 VPC 엔드포인트 구성에 대한 지원이 추가되었습니다. AWS PrivateLink 따라서 NAT 인스턴스, VPN 연결 또는 AWS Direct Connect(를) 통해 인터넷에 액세스하지 않고도 VPC와 Amazon ECR 간에 프라이빗 연결을 생성할 수 있습니다.</p> <p>자세한 정보는 <a href="#">Amazon ECR 인터페이스 VPC 엔드포인트 ()AWS PrivateLink</a>을 참조하세요.</p>	2019년 1월 25일

변경 사항	설명	날짜
리소스에 태깅	Amazon ECR에 리포지토리에 메타데이터 태그를 추가하기 위한 지원이 추가되었습니다.  자세한 정보는 <a href="#">Amazon ECR의 프라이빗 리포지토리에 태그 지정</a> 을 참조하세요.	2018년 12월 18일
Amazon ECR 이름 변경	Amazon Elastic Container Registry 이름이 변경되었습니다(이전 이름: Amazon EC2 Container Registry).	2017년 11월 21일
수명 주기 정책	Amazon ECR 수명 주기 정책을 통해 리포지토리의 이미지에 대한 수명 주기 관리를 지정할 수 있습니다.  자세한 정보는 <a href="#">Amazon ECR의 수명 주기 정책을 사용하여 이미지 정리를 자동화합니다.</a> 을 참조하세요.	2017년 10월 11일
Amazon ECR에서 Docker Image Manifest 2, Schema 2 지원	Amazon ECR에서 이제 Docker Image Manifest V2 Schema 2(Docker 버전 1.10 이상에서 사용됨)를 지원합니다.  자세한 정보는 <a href="#">Amazon ECR의 컨테이너 이미지 매니페스트 형식 지원</a> 을 참조하세요.	2017년 1월 27일
Amazon ECR 정식 출시	Amazon Elastic 컨테이너 레지스트리 (Amazon ECR)는 안전하고 확장 가능하며 신뢰할 수 있는 AWS 관리형 도커 레지스트리 서비스입니다.	2015년 12월 21일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.