OpenShift Container Platform 4.17

레지스트리

OpenShift Container Platform의 레지스트리 설정

Last Updated: 2024-12-12

OpenShift Container Platform 4.17 레지스트리

OpenShift Container Platform의 레지스트리 설정

법적 공지

초록

이 문서에서는OpenShift Container Platform의 내부 레지스트리를 설정하고 관리하는 방법에 대해 설명합니다. 또한 OpenShift Container Platform과 관련된 레지스트리에 대한 일반적인 정보도 소개합니다.

차례

1장. OPENSHIFT 이미지 레지스트리 개요

OpenShift Container Platform은 소스 코드에서 이미지를 빌드 및 배포하고 라이프 사이클을 관리할 수 있습니다. 이미지를 로컬로 관리하기 위해 OpenShift Container Platform 환경에 배포할 수 있는 내부 통합 컨테이너 이미지 레지스트리를 제공합니다. 이 개요에는 OpenShift Container Platform에서 일반적으로 사용되는 레지스트리의 참조 정보와 링크가 포함되어 있으며 OpenShift 이미지 레지스트리에 중점을 두고 있습니다.

1.1. OPENSHIFT 이미지 레지스트리의 일반 용어집

이 용어집은 레지스트리 콘텐츠에 사용되는 공통 용어를 정의합니다.

컨테이너

소프트웨어와 모든 종속 항목으로 구성된 경량 및 실행 가능한 이미지입니다. 컨테이너는 운영 체제를 가상화하므로 데이터 센터, 퍼블릭 또는 프라이빗 클라우드 또는 로컬 호스트에서 컨테이너를 실행할 수 있습니다.

이미지 리포지터리

이미지 리포지터리는 이미지를 식별하는 관련 컨테이너 이미지 및 태그의 컬렉션입니다.

미러 레지스트리

미러 레지스트리는 OpenShift Container Platform 이미지의 미러를 보유한 레지스트리입니다.

네임스페이스

네임스페이스는 단일 클러스터 내에서 리소스 그룹을 격리합니다.

Pod

Pod는 Kubernetes에서 가장 작은 논리 단위입니다. Pod는 작업자 노드에서 실행할 하나 이상의 컨테이너로 구성됩니다.

프라이빗 레지스트리

레지스트리는 컨테이너 이미지 레지스트리 API를 구현하는 서버입니다. 프라이빗 레지스트리는 사용자가 콘텐츠에 액세스할 수 있도록 인증이 필요한 레지스트리입니다.

퍼블릭 레지스트리

레지스트리는 컨테이너 이미지 레지스트리 API를 구현하는 서버입니다. 퍼블릭 레지스트리는 콘텐츠를 공개적으로 제공하는 레지스트리입니다.

Quay.io

Red Hat에서 제공하고 유지 관리하는 공용 Red Hat Quay Container Registry 인스턴스는 대부분의 컨테이너 이미지 및 Operator를 OpenShift Container Platform 클러스터에 제공합니다.

OpenShift 이미지 레지스트리

OpenShift 이미지 레지스트리는 이미지를 관리하기 위해 OpenShift Container Platform에서 제공하는 레지스트리입니다.

레지스트리 인증

개인 이미지 리포지토리에서 이미지를 푸시하고 가져오려면 레지스트리에서 자격 증명을 사용하여 사용자를 인증해야 합니다.

경로

OpenShift Container Platform 인스턴스 외부의 사용자 및 애플리케이션에서 Pod에 대한 네트워크 액세스를 허용하는 서비스를 노출합니다.

축소

복제 수를 줄이기 위해 다음을 수행합니다.

확장

복제 수를 늘리려면 다음을 수행합니다.

서비스

서비스는 Pod 집합에서 실행 중인 애플리케이션을 노출합니다.

1.2. 통합된 OPENSHIFT 이미지 레지스트리

OpenShift Container Platform은 클러스터에서 표준 워크로드로 실행되는 내장 컨테이너 이미지 레지스트리를 제공합니다. 레지스트리는 인프라 Operator에 의해 설정 및 관리됩니다. 사용자가 기존 클러스터 인프라의 상단에서 실행되는 이미지를 관리하여 실제 워크로드를 처리할 수있는 기본 솔루션을 제공합니다. 이 레지스트리는 다른 클러스터 워크로드처럼 확장 또는 축소할 수 있으며 특정 인프라 프로비저닝을 필요로하지 않습니다. 또한 클러스터 사용자 인증 및 권한 부여 시스템에 통합되어 이미지 리소스에 대한 사용자 권한을 정의하여 이미지를 만들고 액세스 권한을 제어할 수 있습니다.

일반적으로 레지스트리는 클러스터에 빌드된 이미지의 게시 대상과 클러스터에서 실행되는 워크로드의 이미지 소스로 사용됩니다. 새 이미지가 레지스트리로 푸시되면 클러스터에 새 이미지에 대한 알림이 전송되고 다른 구성 요소는 업데이트된 이미지에 응답하여 이를 사용할 수 있습니다.

이미지 데이터는 두 위치에 저장됩니다. 실제 이미지 데이터는 클라우드 스토리지 또는 파일 시스템 볼륨과 같은 설정 가능한 스토리지 위치에 저장됩니다. 표준 클러스터 API에서 노출하고 액세스 제어를 수행하는 데 사용되는 이미지 메타데이터는 표준 API 리소스, 특히 이미지 및 이미지 스트림으로 저장됩니다.

추가 리소스

- OpenShift Container Platform의 이미지 레지스트리 Operator

1.3. 타사 레지스트리

OpenShift Container Platform은 타사 레지스트리의 이미지를 사용하여 컨테이너를 생성할 수 있지만 이러한 레지스트리가 통합된 OpenShift 이미지 레지스트리와 동일한 이미지 알림 지원을 제공하지는 않습니다. 이 경우 OpenShift Container Platform은 이미지 스트림 생성 시 원격 레지스트리에서 태그를 가져옵니다. 가져온 태그를 새로 고치려면 **oc import-image <stream>**을 실행합니다. 새 이미지가 감지되면 이전 빌드 및 배포가 다시 생성됩니다.

1.3.1. 인증

OpenShift Container Platform은 레지스트리와 통신하여 사용자가 지정한 인증 정보를 사용하여 개인 이미지 저장소에 액세스할 수 있습니다. 이를 통해 OpenShift Container Platform은 프라이빗 리포지토리에서 이미지 푸시 및 풀 작업을 수행할 수 있습니다.

1.3.1.1. Podman을 사용한 레지스트리 인증

일부 컨테이너 이미지 레지스트리에는 액세스 권한이 필요합니다. Podman은 컨테이너 및 컨테이너 이미지를 관리하고 이미지 레지스트리와 상호 작용하기 위한 오픈 소스 툴입니다. Podman을 사용하여 인증 정보를 인증하고 레지스트리 이미지를 가져온 후 로컬 이미지를 로컬 파일 시스템에 저장할 수 있습니다. 다음은 Podman으로 레지스트리를 인증하는 일반적인 예입니다.

프로세스

1. Red Hat Ecosystem Catalog 를 사용하여 Red Hat Repository에서 특정 컨테이너 이미지를 검색하고 필요한 이미지를 선택합니다.

2. **Get this image** 를 클릭하여 컨테이너 이미지에 대한 명령을 찾습니다.

3. 다음 명령을 실행하여 로그인하고 사용자 이름과 암호를 입력하여 인증합니다.

```
$ podman login registry.redhat.io
 Username:<your_registry_account_username>
 Password:<your_registry_account_password>
```

4. 다음 명령을 실행하여 이미지를 다운로드하여 로컬에 저장합니다.

```
$ podman pull registry.redhat.io/<repository_name>
```

1.4. RED HAT QUAY 레지스트리

엔터프라이즈급 컨테이너 이미지 레지스트리가 필요한 경우 Red Hat Quay는 호스팅 서비스와 자체 데이터 센터 또는 클라우드 환경에 설치할 수있는 소프트웨어로 사용할 수 있습니다. Red Hat Quay의 고급 기능에는 지역 복제, 이미지 스캔 및 이미지 롤백 기능이 포함되어 있습니다.

Quay.io 사이트를 방문하여 호스팅된 Quay 레지스트리 계정을 설정합니다. 그 후 Quay 튜토리얼에 따라 Quay 레지스트리에 로그인하고 이미지 관리를 시작합니다.

원격 컨테이너 이미지 레지스트리와 마찬가지로 OpenShift Container Platform에서 Red Hat Quay 레지스트리에 액세스할 수 있습니다.

추가 리소스

- Red Hat Quay 제품 문서

1.5. 인증이 활성화된 RED HAT 레지스트리

Red Hat Ecosystem Catalog의 Container 이미지 섹션을 통해 제공되는 모든 컨테이너 이미지는 이미지 레지스트리의 **registry.redhat.io.**에 호스트됩니다.

레지스트리 **registry.redhat.io**는 OpenShift Container Platform의 이미지 및 호스팅되는 콘텐츠에 액세스하려면 인증이 필요합니다. 새 레지스트리로 마이그레이션한 후 기존 레지스트리를 일정 기간 동안 사용할 수 있습니다.

참고

OpenShift Container Platform은 **registry.redhat.io**에서 이미지를 가져 오므로 이를 사용할 수 있도록 클러스터를 설정해야합니다.

새 레지스트리는 다음과 같은 방법으로 인증에 표준 OAuth 메커니즘을 사용합니다.

- **인증 토큰**: 관리자에 의해 생성되는 토큰으로 이는 시스템에 컨테이너 이미지 레지스트리에 대한 인증 기능을 제공하는 서비스 계정입니다. 서비스 계정은 사용자 계정 변경의 영향을 받지 않으므로 인증에 토큰을 사용하는 것은 안정적이고 유연한 인증 방법입니다. 이는 프로덕션 클러스터에 대해 지원되는 유일한 인증 옵션입니다.

- **웹 사용자 이름 및 암호**: 이는 **access.redhat.com**과 같은 리소스에 로그인하는 데 사용하는 표준 인증 정보 집합입니다. OpenShift Container Platform에서 이 인증 방법을 사용할 수는 있지만 프로덕션 배포에는 지원되지 않습니다. 이 인증 방법은 OpenShift Container Platform 외부의 독립형 프로젝트에서만 사용해야합니다.

사용자 이름 및 암호 또는 인증 토큰 중 하나의 인증 정보를 사용하여 **podman login**을 사용하고 새 레지스트리의 콘텐츠에 액세스합니다.

모든 이미지 스트림은 설치 풀 시크릿을 사용하여 인증하는 새 레지스트리를 가리킵니다.

다음 위치 중 하나에 인증 정보를 배치해야합니다.

- **openshift 네임 스페이스**. **openshift** 네임스페이스의 이미지 스트림을 가져올 수 있도록 인증 정보가 **openshift** 네임스페이스에 있어야 합니다.
- **호스트**: Kubernetes에서 이미지를 가져올 때 호스트의 인증 정보를 사용하므로 호스트에 인증 정보가 있어야합니다.

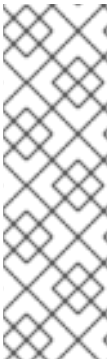
추가 리소스

- 레지스트리 서비스 계정

2장. OPENSHIFT CONTAINER PLATFORM의 이미지 레지스트리 OPERATOR

2.1. 클라우드 플랫폼 및 OPENSTACK의 이미지 레지스트리

Image Registry Operator는 OpenShift 이미지 레지스트리의 단일 인스턴스를 설치하고 레지스트리 스토리지 설정을 포함하여 모든 레지스트리 구성을 관리합니다.



참고

스토리지는 AWS, Azure, GCP, IBM® 또는 OpenStack에 설치 관리자 프로비저닝 인프라 클러스터를 설치할 때만 자동으로 구성됩니다.

AWS, Azure, GCP, IBM® 또는 OpenStack에 설치 관리자 프로비저닝 인프라 클러스터를 설치하거나 업그레이드할 때 Image Registry Operator는 **spec.storage.managementState** 매개변수를 **Managed** 로 설정합니다. **spec.storage.managementState** 매개 변수가 **Unmanaged**로 설정된 경우 이미지 레지스트리 Operator는 스토리지와 관련된 작업을 수행하지 않습니다.

컨트롤 플레인이 관리 클러스터에 배포된 후 Operator는 클러스터에서 탐지된 구성을 기반으로 기본 **configs.imageregistry.operator.openshift.io** 리소스 인스턴스를 생성합니다.

전체 **configs.imageregistry.operator.openshift.io** 리소스를 정의하는 데 사용할 수 있는 정보가 충분하지 않으면 불완전한 리소스가 정의되고 Operator는 누락된 항목에 대한 정보로 리소스 상태를 업데이트합니다.

중요

pruner를 관리하기위한 이미지 레지스트리 Operator의 동작은 이미지 레지스트리 Operator의 **ClusterOperator** 개체에 지정된 **ManagementState**와는 별개입니다. 이미지 레지스트리 Operator가 **Managed** 상태가 아닌 경우 이미지 pruner는 **Pruning** 사용자 정의 리소스로 설정 및 관리할 수 있습니다.

그러나 이미지 레지스트리 Operator의 **managementState**는 배포된 이미지 pruner 작업의 동작을 변경합니다.

- **Managed**: 이미지 pruner의 **--prune-registry** 플래그가 **true**로 설정됩니다.
- **Removed**: 이미지 pruner의 **--prune-registry** 플래그가 **false** 로 설정되어 있습니다. 즉 etcd에서 이미지 메타데이터만 정리됩니다.

2.2. 베어 메탈, NUTANIX 및 VSPHERE의 이미지 레지스트리

2.2.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 Image Registry Operator 구성을 편집해야 합니다. 이 작업이 완료되면 스토리지를 구성해야 합니다.

2.3. 가용성 영역에 대한 이미지 레지스트리 OPERATOR 배포

이미지 레지스트리 Operator의 기본 구성에서 모든 Pod가 영향을 받는 전체 영역 실패의 경우 지연된 복구 시간을 방지하기 위해 이미지 레지스트리 Pod를 토폴로지 영역에 분배합니다.

영역 관련 토폴로지 제약 조건을 사용하여 배포할 경우 Image Registry Operator의 기본값은 다음과 같습니다.

영역 관련 토폴로지 제약 조건을 사용하여 배포된 이미지 레지스트리 Operator

```
topologySpreadConstraints:
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: kubernetes.io/hostname
  whenUnsatisfiable: DoNotSchedule
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: node-role.kubernetes.io/worker
  whenUnsatisfiable: DoNotSchedule
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: DoNotSchedule
```

이미지 레지스트리 Operator는 베어 메탈 및 vSphere 인스턴스에 적용되는 영역 관련 토폴로지 제약 조건 없이 배포할 때 기본적으로 다음과 같이 설정됩니다.

영역 관련 토폴로지 제약 조건없이 이미지 레지스트리 Operator 배포

```
topologySpreadConstraints:
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: kubernetes.io/hostname
  whenUnsatisfiable: DoNotSchedule
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: node-role.kubernetes.io/worker
  whenUnsatisfiable: DoNotSchedule
```

클러스터 관리자는 **configs.imageregistry.operator.openshift.io/cluster** 사양 파일을 구성하여 기본 **topologySpreadConstraints** 를 덮어쓸 수 있습니다. 이 경우 제공하는 제약 조건만 적용됩니다.

2.4. 추가 리소스

- Pod 토폴로지 분배 제약 조건 구성

2.5. 이미지 레지스트리 OPERATOR 설정 매개 변수

ingresscontrollers.operator.openshift.io 리소스에서 제공되는 설정 매개변수는 다음과 같습니다.

매개변수	설명
managementState	**Managed**: Operator는 설정 리소스가 업데이트되면 레지스트리를 업데이트합니다. **Unmanaged**: Operator는 설정 리소스에 대한 변경을 무시합니다. **Removed**: Operator는 레지스트리 인스턴스를 제거하고 Operator가 프로비저닝한 모든 스토리지를 삭제합니다.
logLevel	레지스트리 인스턴스의 **logLevel**을 설정합니다. 기본값은 **Normal**입니다. **logLevel** 에 대한 다음 값이 지원됩니다. • **Normal** • **Debug** • **Trace** • **TraceAll**
httpSecret	기본적으로 생성되는 업로드의 보안을 위해 레지스트리에 필요한 값입니다.
operatorLogLevel	**operatorLogLevel** 구성 매개변수는 Operator 자체에 대한 의도 기반 로깅과 Operator가 자체적으로 해석해야 하는 집계된 로깅 선택을 관리하는 간단한 방법을 제공합니다. 이 구성 매개변수의 기본값은 **Normal** 입니다. 세분화된 제어를 제공하지 않습니다. **operatorLogLevel**에 대한 다음 값이 지원됩니다. • **Normal** • **Debug** • **Trace** • **TraceAll**
proxy	마스터 API 및 업스트림 레지스트리를 호출할 때 사용할 프록시를 정의합니다.

매개변수	설명
유사성	**affinity** 매개변수를 사용하여 이미지 레지스트리 Operator Pod에 대한 Pod 예약 기본 설정 및 제약 조건을 구성할 수 있습니다. 유사성 설정은 **podAffinity** 또는 **podAntiAffinity** 사양을 사용할 수 있습니다. 두 옵션 모두 **preferredDuringSchedulingIgnoredDuringExecution** 규칙 또는 **requiredDuringSchedulingIgnoredDuringExecution** 규칙을 사용할 수 있습니다.
storage	**Storagetype**: 레지스트리 스토리지 설정에 대한 세부 정보 (예: S3 버킷 위치 정보 등)입니다. 일반적으로 기본적으로 설정됩니다.
readOnly	레지스트리 인스턴스가 새 이미지를 푸시하거나 기존 이미지를 삭제하려는 시도를 거부해야하는지 여부를 나타냅니다.
requests	API 요청 제한 세부 사항입니다. 추가 요청을 대기열에 추가하기 전에 지정된 레지스트리 인스턴스가 처리할 병렬 요청 수를 제어합니다.
defaultRoute	기본 호스트 이름을 사용하여 외부 경로를 정의할지 여부를 결정합니다. 활성화된 경우 경로는 암호화된 데이터를 다시 암호화합니다. 기본값은 **false**입니다.
routes	생성할 추가 경로의 배열입니다. 경로의 호스트 이름과 인증서를 지정합니다.
rolloutStrategy	이미지 레지스트리 배포에 대한 롤아웃 전략을 정의합니다. 기본값은 **RollingUpdate** 입니다.
replicas	레지스트리의 복제본 수입니다.
disableRedirect	백엔드로 리디렉션하지 않고 레지스트리를 통해 모든 데이터를 라우팅할지 여부를 제어합니다. 기본값은 **false**입니다.
spec.storage.managementState	AWS 또는 Azure에 설치 관리자 프로비저닝 인프라를 사용하여 클러스터를 새로 설치하거나 업그레이드할 때 Image Registry Operator에서 **spec.storage.managementState** 매개변수를 **Managed**로 설정합니다. ● **Managed**: Image Registry Operator에서 기본 스토리지를 관리합니다. Image Registry Operator의 **managementState**가 **Removed**로 설정되면 스토리지가 삭제됩니다. ○ **managementState**가 **Managed**로 설정된 경우 Image Registry Operator는 기본 스토리지 장치에 일부 기본 구성을 적용합니다. 예를 들어 **Managed**로 설정된 경우 Operator는 레지스트리에서 암호화를 사용할 수 있도록 S3 버킷에서 활성화합니다. 제공하는 스토리지에 기본 설정을 적용하지 않으려면 **managementState**를 **Unmanaged**로 설정해야 합니다. ● **Unmanaged**: Image Registry Operator에서 스토리지 설정을 무시합니다. Image Registry Operator의 **managementState**가 **Removed**로 설정되어도 스토리지가 삭제되지 않습니다. 버킷 또는 컨테이너 이름과 같은 기본 스토리지 장치 구성을 제공한 후 **spec.storage.managementState**에는 아직 값을 설정하지 않은 경우, Image Registry Operator에서 이를 **Unmanaged**로 구성합니다.

2.6. CRD를 사용하여 이미지 레지스트리 기본 경로 활성화

OpenShift Container Platform에서 **Registry** Operator는 OpenShift 이미지 레지스트리 기능을 제어합니다. Operator는 **configs.imageregistry.operator.openshift.io** CRD (Custom Resource Definition)에 의해 정의됩니다.

이미지 레지스트리 기본 경로를 자동으로 활성화해야하는 경우 이미지 레지스트리 Operator CRD 패치를 적용합니다.

프로세스

- 이미지 레지스트리 Operator CRD에 패치를 적용합니다.

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec":
{"defaultRoute":true}}'
```

2.7. 이미지 레지스트리 액세스를 위한 추가 신뢰 저장소 구성

image.config.openshift.io/cluster 사용자 지정 리소스에는 이미지 레지스트리 액세스 중에 신뢰할 수 있는 추가 인증 기관이 포함된 구성 맵에 대한 참조가 포함될 수 있습니다.

사전 요구 사항

- 인증 기관(CA)은 PEM으로 인코딩되어야 합니다.

프로세스

openshift-config 네임 스페이스에 구성 맵을 만들고 **image.config.openshift.io** 사용자 지정 리소스에서 **AdditionalTrustedCA**의 해당 이름을 사용하여 외부 레지스트리에 연결할 때 신뢰할 수있는 추가 CA를 제공할 수 있습니다.

구성 맵 키는 이 CA가 신뢰할 수있는 포트가 있는 레지스트리의 호스트 이름이며 PEM 인증서 콘텐츠는 신뢰할 수 있는 각 추가 레지스트리 CA의 값입니다.

이미지 레지스트리 CA 구성 맵의 예

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-registry-ca
data:
  registry.example.com: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com..5000: | 1
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

1 레지스트리에 **registry-with-port.example.com:5000** 같은 포트가 있는 경우 **:**이 **..**로 교체되어야 합니다.

다음 절차에 따라 추가 CA를 구성할 수 있습니다.

- 추가 CA를 구성하려면 다음을 실행합니다.

```
$ oc create configmap registry-config --from-file=<external_registry_address>=ca.crt -n
openshift-config
```

```
$ oc edit image.config.openshift.io cluster
```

```
spec:
  additionalTrustedCA:
    name: registry-config
```

2.8. 이미지 레지스트리 OPERATOR의 스토리지 인증 정보 설정

configs.imageregistry.operator.openshift.io 및 ConfigMap 리소스 외에도 스토리지 인증 정보 설정은 **openshift-image-registry** 네임 스페이스 내에 있는 별도의 시크릿 리소스를 통해 Operator에게 제공됩니다.

image-registry-private-configuration-user 시크릿은 스토리지 액세스 및 관리에 필요한 인증 정보를 제공합니다. 기본 인증 정보가 검색되면 Operator가 사용하는 기본 인증 정보를 덮어씁니다.

프로세스

- 필수 키가 포함된 OpenShift Container Platform 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=KEY1=value1 --from-literal=KEY2=value2 --namespace openshift-image-registry
```

2.9. 추가 리소스

- AWS 사용자 프로비저닝 인프라의 레지스트리 설정
- GCP 사용자 프로비저닝 인프라의 레지스트리 설정
- Azure 사용자 프로비저닝 인프라의 레지스트리 설정
- 베어 메탈의 레지스트리 설정
- vSphere의 레지스트리 설정
- RHOSP의 레지스트리 구성
- Red Hat OpenShift Data Foundation의 레지스트리 설정
- Nutanix의 레지스트리 구성

3장. 레지스트리 설정 및 구성

3.1. AWS 사용자 프로비저닝 인프라의 레지스트리 설정

3.1.1. 이미지 레지스트리 Operator의 시크릿 설정

configs.imageregistry.operator.openshift.io 및 ConfigMap 리소스 외에도 **openshift-image-registry** 네임 스페이스 내에 있는 별도의 시크릿 리소스에 의해 설정이 Operator에게 제공됩니다.

image-registry-private-configuration-user 시크릿은 스토리지 액세스 및 관리에 필요한 인증 정보를 제공합니다. 기본 인증 정보가 검색되면 Operator가 사용하는 기본 인증 정보를 덮어씁니다.

Amazon 스토리지 S3의 경우 시크릿에는 다음 두 개의 키가 포함되어야 합니다.

- **REGISTRY_STORAGE_S3_ACCESSKEY**
- **REGISTRY_STORAGE_S3_SECRETKEY**

프로세스

- 필수 키가 포함된 OpenShift Container Platform 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=myaccesskey --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=mysecretkey --namespace openshift-image-registry
```

3.1.2. 사용자 프로비저닝 인프라로 AWS의 레지스트리 스토리지 설정

설치하는 동안 클라우드 자격 증명만으로도 Amazon S3 버킷을 생성할 수 있으며 Registry Operator가 자동으로 스토리지를 구성합니다.

Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할수 없는 경우 다음 프로시저에 따라 S3 버킷을 생성하고 스토리지를 구성할 수 있습니다.

사전 요구 사항

- AWS에 사용자 프로비저닝된 인프라가 있는 클러스터가 있어야 합니다.
- Amazon S3 스토리지의 경우 시크릿에는 두 개의 키가 포함되어야 합니다.
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

프로세스

Registry Operator가 S3 버킷을 생성하고 스토리지를 자동으로 구성할 수 없는 경우 다음 프로시저를 사용합니다.

1. 1일이 지난 완료되지 않은 다중 파트 업로드를 중단하도록 Bucket Lifecycle Policy 를 설정합니다.
2. **configs.imageregistry.operator.openshift.io/cluster**에 스토리지 설정을 입력합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

설정 예

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

주의

AWS에서 레지스트리 이미지를 보안을 위해 S3 버킷에 공용 액세스를 차단합니다.

3.1.3. AWS S3의 이미지 레지스트리 Operator 설정 매개 변수

다음 설정 매개 변수는 AWS S3 레지스트리 스토리지에 사용할 수 있습니다.

이미지 레지스트리 **spec.storage.s3** 구성 매개변수에는 백엔드 스토리지에 AWS S3 서비스를 사용하도록 레지스트리를 구성하는 정보가 들어 있습니다. 자세한 내용은 S3 스토리지 드라이버 설명서를 참조하십시오.

매개변수	설명
bucket	버킷은 레지스트리의 데이터를 저장할 버킷 이름입니다. 이는 선택 사항이며 지정되지 않은 경우 생성됩니다.
chunkSizeMiB	ChunkSizeMiB는 S3 API의 multipart 업로드 청크 크기입니다. 기본값은 **10** MiB이고 최소 **5** MiB입니다.
region	리전은 버킷이 있는 AWS 리전입니다. 이는 선택 사항이며 설치된 AWS 리전에 따라 설정됩니다.
regionEndpoint	RegionEndpoint는 S3 호환 스토리지 서비스의 엔드 포인트입니다. 이는 지정된 지역에 따라 선택 사항 및 기본값입니다.
virtualHostedStyle	VirtualHostedStyle은 사용자 지정 RegionEndpoint에서 S3 가상 호스팅 스타일 버킷 경로 사용을 활성화합니다. 이는 선택 사항이며 기본값은 false입니다. 이 매개 변수를 설정하여 OpenShift Container Platform을 숨겨진 지역에 배포합니다.
encrypt	encrypt는 레지스트리가 이미지를 암호화된 형식으로 저장할지 여부를 지정합니다. 이는 선택 사항이며 기본값은 false입니다.
keyID	KeyID는 암호화에 사용할 KMS 키 ID입니다. 이는 선택 사항입니다. Encrypt는 true이어야합니다. 그렇지 않으면 이 매개 변수가 무시됩니다.

매개변수	설명
cloudFront	CloudFront는 Amazon Cloudfront를 레지스트리에 스토리지 미들웨어로 설정합니다. 이는 선택 사항입니다.
trustedCA	**trustedCA** 에서 참조하는 구성 맵의 네임스페이스는**openshift-config** 입니다. 구성 맵의 번들 키는 **ca-bundle.crt** 입니다. 이는 선택 사항입니다.

참고

regionEndpoint 매개변수의 값이 Rados Gateway URL로 구성된 경우 명시적 포트를 지정할 수 없습니다. 예를 들면 다음과 같습니다.

```
regionEndpoint: http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-storage.svc.cluster.local
```

3.2. GCP 사용자 프로비저닝 인프라의 레지스트리 설정

3.2.1. 이미지 레지스트리 Operator의 시크릿 설정

configs.imageregistry.operator.openshift.io 및 ConfigMap 리소스 외에도 **openshift-image-registry** 네임 스페이스 내에 있는 별도의 시크릿 리소스에 의해 설정이 Operator에게 제공됩니다.

image-registry-private-configuration-user 시크릿은 스토리지 액세스 및 관리에 필요한 인증 정보를 제공합니다. 기본 인증 정보가 검색되면 Operator가 사용하는 기본 인증 정보를 덮어씁니다.

GCP 저장소의 GCS의 경우 보안 시크릿에는 GCP에서 제공하는 사용자 인증 정보 파일의 콘텐츠 값에 해당하는 하나의 키가 포함되어야 합니다.

- **REGISTRY_STORAGE_GCS_KEYFILE**

프로세스

- 필수 키가 포함된 OpenShift Container Platform 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-file=REGISTRY_STORAGE_GCS_KEYFILE=<path_to_keyfile> --namespace openshift-image-registry
```

3.2.2. 사용자 프로비저닝 인프라를 사용하여 GCP의 레지스트리 스토리지 설정

Registry Operator가 GCP(Google Cloud Platform) 버킷을 생성할 수 없는 경우 스토리지 미디어를 수동으로 설정하고 레지스트리 CR(사용자 정의 리소스)에서 설정을 구성해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라가 있는 GCP의 클러스터.
- GCP의 레지스트리 스토리지를 설정하려면 레지스트리 Operator 클라우드 사용자 인증 정보를 지정해야합니다.

- GCP 저장소의 GCS의 경우 보안 시크릿에는 GCP에서 제공하는 사용자 인증 정보 파일의 콘텐츠 값에 해당하는 하나의 키가 포함되어야 합니다.
  - **REGISTRY_STORAGE_GCS_KEYFILE**

프로세스

1. 1일이 지난 불완전한 다중 파트 업로드를 중단하도록 Object Lifecycle Management 정책 을 설정합니다.
2. **configs.imageregistry.operator.openshift.io/cluster**에 스토리지 설정을 입력합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

설정 예

```
# ...
storage:
  gcs:
    bucket: <bucket-name>
    projectID: <project-id>
    region: <region-name>
# ...
```

주의

공용 액세스 방지 를 설정하여 Google Cloud Storage 버킷을 사용하는 레지스트리 이미지를 보호할 수 있습니다.

3.2.3. GCP GCS의 이미지 레지스트리 Operator 설정 매개 변수

다음 설정 매개 변수는 GCP GCS 레지스트리 스토리지에 사용할 수 있습니다.

매개변수	설명
bucket	버킷은 레지스트리의 데이터를 저장할 버킷 이름입니다. 이는 선택 사항이며 지정되지 않은 경우 생성됩니다.
region	리전은 버킷이 있는 GCS 위치입니다. 이는 선택 사항이며 설치된 GCS 지역에 따라 설정됩니다.
projectID	ProjectID는 이 버킷이 연결되어야하는 GCP 프로젝트의 프로젝트 ID입니다. 이는 선택 사항입니다.
keyID	KeyID는 암호화에 사용할 KMS 키 ID입니다. 버킷은 기본적으로 GCP에서 암호화되어 있으므로 이는 선택 사항입니다. 이를 통해 사용자 지정 암호화 키를 사용할 수 있습니다.

3.3. OPENSTACK 사용자 프로비저닝 인프라의 레지스트리 설정

자체 RHOSP(Red Hat OpenStack Platform) 인프라에서 실행되는 클러스터의 레지스트리를 구성할 수 있습니다.

3.3.1. 이미지 레지스트리 Operator 리디렉션 구성

리디렉션을 비활성화하면 OpenShift Container Platform 클러스터 빌드와 같은 클라이언트 또는 개발자 머신과 같은 외부 시스템이 RHOSP(Red Hat OpenStack Platform) Swift 스토리지에서 직접 이미지를 가져오기 위해 리디렉션되는지 여부를 제어하도록 Image Registry Operator를 구성할 수 있습니다. 이 구성은 선택 사항이며 클라이언트가 스토리지의 SSL/TLS 인증서를 신뢰하는지 여부에 따라 다릅니다.

참고

클라이언트가 스토리지 인증서를 신뢰하지 않는 경우 **disableRedirect** 옵션을 이미지 레지스트리를 통한 **true** 프록시 트래픽으로 설정할 수 있습니다. 결과적으로 이미지 레지스트리에는 증가된 부하를 처리하기 위해 더 많은 리소스, 특히 네트워크 대역폭이 필요할 수 있습니다.

또는 클라이언트가 스토리지 인증서를 신뢰하는 경우 레지스트리에서 리디렉션을 허용할 수 있습니다. 이렇게 하면 레지스트리 자체의 리소스 수요가 줄어듭니다.

일부 사용자는 리디렉션을 비활성화하는 대신 자체 서명된 CA(인증 기관)를 신뢰하도록 클라이언트를 구성하는 것을 선호할 수 있습니다. 자체 서명된 CA를 사용하는 경우 사용자 정의 CA를 신뢰하거나 리디렉션을 비활성화해야 합니다.

프로세스

- 이미지 레지스트리가 Swift 스토리지에 의존하는 대신 트래픽을 프록시하도록 하려면 다음 명령을 실행하여 **config.imageregistry** 오브젝트의 **spec.disableRedirect** 필드 값을 **true** 로 변경합니다.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"disableRedirect":true}}'
```

3.3.2. 이미지 레지스트리 Operator의 시크릿 설정

configs.imageregistry.operator.openshift.io 및 ConfigMap 리소스 외에도 **openshift-image-registry** 네임 스페이스 내에 있는 별도의 시크릿 리소스에 의해 설정이 Operator에게 제공됩니다.

image-registry-private-configuration-user 시크릿은 스토리지 액세스 및 관리에 필요한 인증 정보를 제공합니다. 기본 인증 정보가 검색되면 Operator가 사용하는 기본 인증 정보를 덮어씁니다.

RHOSP(Red Hat OpenStack Platform) 스토리지의 Swift의 경우 시크릿에는 다음 두 개의 키가 포함되어야 합니다.

- **REGISTRY_STORAGE_SWIFT_USERNAME**
- **REGISTRY_STORAGE_SWIFT_PASSWORD**

프로세스

- 필수 키가 포함된 OpenShift Container Platform 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_SWIFT_USERNAME=<username> --from-
literal=REGISTRY_STORAGE_SWIFT_PASSWORD=<password> -n openshift-image-
registry
```

3.3.3. 사용자 프로비저닝 인프라를 사용하는 RHOSP의 레지스트리 스토리지

Registry Operator가 Swift 버킷을 생성할 수 없는 경우 스토리지 미디어를 수동으로 설정하고 레지스트리 CR(사용자 정의 리소스)에서 설정을 구성해야 합니다.

사전 요구 사항

- 사용자 프로비저닝 인프라가 있는 RHOSP(Red Hat OpenStack Platform)의 클러스터입니다.
- RHOSP의 레지스트리 스토리지를 구성하려면 Registry Operator 클라우드 인증 정보를 제공해야 합니다.
- RHOSP 스토리지의 Swift의 경우 시크릿에는 다음 두 개의 키가 포함되어야 합니다.
  - **REGISTRY_STORAGE_SWIFT_USERNAME**
  - **REGISTRY_STORAGE_SWIFT_PASSWORD**

프로세스

- **configs.imageregistry.operator.openshift.io/cluster**에 스토리지 설정을 입력합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

설정 예

```
# ...
storage:
  swift:
    container: <container-id>
# ...
```

3.3.4. RHOSP Swift의 이미지 레지스트리 Operator 구성 매개변수

RHOSP(Red Hat OpenStack Platform) Swift 레지스트리 스토리지에 다음 설정 매개변수를 사용할 수 있습니다.

매개변수	설명
authURL	인증 토큰을 가져올 URL을 정의합니다. 이 값은 선택 사항입니다.
authVersion	RHOSP의 Auth 버전을 지정합니다(예: **authVersion: "3"**). 이 값은 선택 사항입니다.
container	레지스트리 데이터를 저장하기 위한 Swift 컨테이너의 이름을 정의합니다. 이 값은 선택 사항입니다.

매개변수	설명
domain	Identity v3 API의 RHOSP 도메인 이름을 지정합니다. 이 값은 선택 사항입니다.
domainID	Identity v3 API의 RHOSP 도메인 ID를 지정합니다. 이 값은 선택 사항입니다.
tenant	레지스트리에서 사용할 RHOSP 테넌트 이름을 정의합니다. 이 값은 선택 사항입니다.
tenantID	레지스트리에서 사용할 RHOSP 테넌트 ID를 정의합니다. 이 값은 선택 사항입니다.
regionName	컨테이너가 있는 RHOSP 리전을 정의합니다. 이 값은 선택 사항입니다.

3.4. AZURE 사용자 프로비저닝 인프라의 레지스트리 설정

3.4.1. 이미지 레지스트리 Operator의 시크릿 설정

configs.imageregistry.operator.openshift.io 및 ConfigMap 리소스 외에도 **openshift-image-registry** 네임 스페이스 내에 있는 별도의 시크릿 리소스에 의해 설정이 Operator에게 제공됩니다.

image-registry-private-configuration-user 시크릿은 스토리지 액세스 및 관리에 필요한 인증 정보를 제공합니다. 기본 인증 정보가 검색되면 Operator가 사용하는 기본 인증 정보를 덮어씁니다.

Azure 레지스트리 스토리지의 경우 보안 시크릿은 값이 Azure에서 제공하는 인증 정보 파일의 콘텐츠 값에 해당하는 하나의 키를 포함해야합니다.

- **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

프로세스

- 필요한 키가 포함된 OpenShift Container Platform 보안 시크릿을 만듭니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_AZURE_ACCOUNTKEY=<accountkey> --namespace openshift-image-registry
```

3.4.2. Azure의 레지스트리 스토리지 설정

설치하는 동안 클라우드 인증 정보만으로도 Azure Blob Storage를 생성할 수 있으며 레지스트리 Operator가 자동으로 스토리지를 설정합니다.

전제 조건

- 사용자 프로비저닝 인프라가 있는 Azure의 클러스터.
- Azure의 레지스트리 스토리지를 설정하려면 레지스트리 Operator 클라우드 인증 정보를 지정해야합니다.
- Azure 스토리지의 경우 시크릿에는 하나의 키가 포함되어야 합니다.

- **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

프로세스

1. Azure 스토리지 컨테이너를 생성합니다.
2. **configs.imageregistry.operator.openshift.io/cluster**에 스토리지 설정을 입력합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

설정 예

```
storage:
  azure:
    accountName: <storage-account-name>
    container: <container-name>
```

3.4.3. Azure Government의 레지스트리 스토리지 설정

설치하는 동안 클라우드 인증 정보만으로도 Azure Blob Storage를 생성할 수 있으며 레지스트리 Operator가 자동으로 스토리지를 설정합니다.

전제 조건

- Government 리전에서 사용자가 프로비저닝한 인프라가 있는 Azure의 클러스터입니다.
- Azure의 레지스트리 스토리지를 설정하려면 레지스트리 Operator 클라우드 인증 정보를 지정해야합니다.
- Azure 스토리지의 경우 시크릿에는 하나의 키가 포함되어야 합니다.
  - **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

프로세스

1. Azure 스토리지 컨테이너를 생성합니다.
2. **configs.imageregistry.operator.openshift.io/cluster**에 스토리지 설정을 입력합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

설정 예

```
storage:
  azure:
    accountName: <storage-account-name>
    container: <container-name>
    cloudName: AzureUSGovernmentCloud (1)
```

(1) **cloudName**은 적절한 Azure API 엔드포인트에서 Azure SDK를 설정하는데 사용되는 Azure 클라우드 환경의 이름입니다. 기본값은 **AzurePublicCloud** 입니다. 올바른 인증 정보를 사용하여 **cloudName**을 **AzureUSGovernmentCloud**, **AzureChinaCloud** 또는 **AzureGermanCloud**로 설정할 수 있습니다.

3.5. RHOSP의 레지스트리 구성

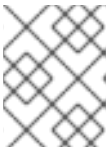
3.5.1. RHOSP에서 실행되는 클러스터에서 사용자 지정 스토리지를 사용하여 이미지 레지스트리 구성

RHOSP(Red Hat OpenStack Platform)에 클러스터를 설치한 후 레지스트리 스토리지의 특정 가용성 영역에 있는 Cinder 볼륨을 사용할 수 있습니다.

프로세스

1. 사용할 스토리지 클래스 및 가용성 영역을 지정하는 YAML 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: custom-csi-storageclass
provisioner: cinder.csi.openstack.org
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
parameters:
  availability: <availability_zone_name>
```



참고

OpenShift Container Platform은 선택한 가용성 영역이 있는지 확인하지 않습니다. 구성을 적용하기 전에 가용성 영역의 이름을 확인합니다.

2. 명령줄에서 구성을 적용합니다.

```
$ oc apply -f <storage_class_file_name>
```

출력 예

```
storageclass.storage.k8s.io/custom-csi-storageclass created
```

3. 스토리지 클래스와 **openshift-image-registry** 네임스페이스를 사용하는 PVC(영구 볼륨 클레임)를 지정하는 YAML 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-imageregistry
  namespace: openshift-image-registry 1
  annotations:
    imageregistry.openshift.io: "true"
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Filesystem
  resources:
```

```
    requests:
      storage: 100Gi 2
  storageClassName: <your_custom_storage_class> 3
```

1. 네임스페이스 **openshift-image-registry** 를 입력합니다. 이 네임스페이스를 사용하면 Cluster Image Registry Operator에서 PVC를 사용할 수 있습니다.
2. 선택 사항: 볼륨 크기를 조정합니다.
3. 생성한 스토리지 클래스의 이름을 입력합니다.

4. 명령줄에서 구성을 적용합니다.

```
$ oc apply -f <pvc_file_name>
```

출력 예

```
persistentvolumeclaim/csi-pvc-imageregistry created
```

5. 이미지 레지스트리 구성의 원래 영구 볼륨 클레임을 새 클레임으로 교체합니다.

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type 'json' -p='[{"op":
"replace", "path": "/spec/storage/pvc/claim", "value": "csi-pvc-imageregistry"}]'
```

출력 예

```
config.imageregistry.operator.openshift.io/cluster patched
```

다음 몇 분 동안 구성이 업데이트됩니다.

검증

레지스트리에서 사용자가 정의한 리소스를 사용하고 있는지 확인하려면 다음을 수행합니다.

1. PVC 클레임 값이 PVC 정의에 제공한 이름과 동일한지 확인합니다.

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

출력 예

```
...
status:
    ...
    managementState: Managed
    pvc:
      claim: csi-pvc-imageregistry
...
```

2. PVC의 상태가 **Bound** 인지 확인합니다.

```
$ oc get pvc -n openshift-image-registry csi-pvc-imageregistry
```

출력 예

```
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS            AGE
csi-pvc-imageregistry   Bound    pvc-72a8f9c9-f462-11e8-b6b6-fa163e18b7b5   100Gi
RWO            custom-csi-storageclass   11m
```

3.6. 베어 메탈의 레지스트리 설정

3.6.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 Image Registry Operator 구성을 편집해야 합니다. 이 작업이 완료되면 스토리지를 구성해야 합니다.

3.6.2. 이미지 레지스트리의 관리 상태 변경

이미지 레지스트리를 시작하려면 Image Registry Operator 구성의 **managementState**를 **Removed**에서 **Managed**로 변경해야 합니다.

절차

- **managementState** Image Registry Operator 구성을 **Removed**에서 **Managed**로 변경합니다. 예를 들면 다음과 같습니다.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

3.6.3. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

3.6.3.1. 베어메탈 및 기타 수동 설치를 위한 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

- 베어 메탈과 같이 수동으로 프로비저닝된 RHCOS(Red Hat Enterprise Linux CoreOS) 노드를 사용하는 클러스터가 있어야 합니다.
- Red Hat OpenShift Data Foundation과 같이 클러스터용 영구 스토리지를 프로비저닝합니다.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

절차

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.

참고

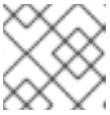
공유 스토리지를 사용하는 경우 보안 설정을 검토하여 외부 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resources found in openshift-image-registry namespace
```



참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim:
```

image-registry-storage PVC의 자동 생성을 허용하도록 **claim** 필드를 비워 둡니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION        AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.17           True        False         False      6h50m
```

5. 이미지를 빌드 및 푸시할 수 있도록 레지스트리의 관리가 설정되어 있는지 확인하십시오.

- 다음을 실행합니다.

```
$ oc edit configs.imageregistry/cluster
```

다음으로 라인을 변경하십시오.

```
managementState: Removed
```

다음으로 변경

```
managementState: Managed
```

3.6.3.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

> **주의**
>
> 프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

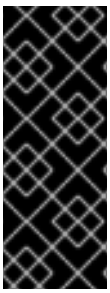
Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

몇 분 후에 명령을 다시 실행하십시오.

3.6.3.3. 베어메탈용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨 또는 블록 영구 볼륨은 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

이미지 레지스트리와 함께 블록 스토리지 볼륨을 사용하도록 선택하는 경우 파일 시스템 PVC(영구 볼륨 클레임)를 사용해야 합니다.

프로세스

1. 다음 명령을 입력하여 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하고, **Recreate** 롤아웃 전략을 사용하도록 레지스트리를 패치하고, 하나의 (**1**) 복제본으로만 실행됩니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 PV를 프로비저닝하고 해당 볼륨의 PVC를 생성합니다. 요청된 블록 볼륨은 RWO(ReadWriteOnce) 액세스 모드를 사용합니다.

   a. VMware vSphere **PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** 개체를 표시하는 고유한 이름입니다.
   2. **PersistentVolumeClaim** 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.
   3. 영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.
   4. 영구 볼륨 클레임의 크기입니다.

   b. 다음 명령을 입력하여 파일에서 **PersistentVolumeClaim** 오브젝트를 생성합니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 구성을 편집하려면 다음 명령을 입력합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 사용자 정의 PVC를 생성하면 **image-registry-storage** PVC의 기본 자동 생성을 위해 **claim** 필드를 비워 둘 수 있습니다.

3.6.3.4. Red Hat OpenShift Data Foundation에서 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Ceph RGW 오브젝트 스토리지를 제공합니다.

프로세스

1. **ocs-storagecluster-ceph-rgw** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage 1
spec:
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

1 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 다음 명령을 입력하여 **경로** 호스트를 가져옵니다.

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 다음 명령을 입력하여 Ceph RGW 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.6.3.5. Red Hat OpenShift Data Foundation에서 Noobaa 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Noobaa 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Noobaa 오브젝트 스토리지를 제공합니다.

프로세스

1. **openshift-storage.noobaa.io** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
  namespace: openshift-storage (1)
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

(1) 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 다음 명령을 입력하여 경로 호스트를 가져옵니다.

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 다음 명령을 입력하여 Nooba 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

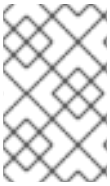
```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}' --type=merge
```

3.6.4. Red Hat OpenShift Data Foundation에서 CephFS 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 CephFS 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.



참고

CephFS는 PVC(영구 볼륨 클레임) 스토리지를 사용합니다. Ceph RGW 또는 Noobaa와 같은 다른 옵션을 사용할 수 있는 경우 이미지 레지스트리 스토리지에 PVC를 사용하지 않는 것이 좋습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- 오브젝트 스토리지 및 CephFS 파일 스토리지를 제공하기 위해 OpenShift Data Foundation Operator 를 설치했습니다.

프로세스

1. **cephfs** 스토리지 클래스를 사용할 PVC를 생성합니다. 예를 들면 다음과 같습니다.

-

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 다음 명령을 입력하여 CephFS 파일 시스템 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.6.5. 추가 리소스

- 권장되는 구성 가능한 스토리지 기술
- OpenShift Data Foundation을 사용하도록 이미지 레지스트리 구성

3.7. VSPHERE의 레지스트리 설정

3.7.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 Image Registry Operator 구성을 편집해야 합니다. 이 작업이 완료되면 스토리지를 구성해야 합니다.

3.7.2. 이미지 레지스트리의 관리 상태 변경

이미지 레지스트리를 시작하려면 Image Registry Operator 구성의 **managementState**를 **Removed**에서 **Managed**로 변경해야 합니다.

절차

- **managementState** Image Registry Operator 구성을 **Removed**에서 **Managed**로 변경합니다. 예를 들면 다음과 같습니다.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

3.7.3. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

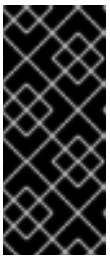
업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

3.7.3.1. VMware vSphere용 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- 클러스터 관리자 권한이 있어야 합니다.
- VMware vSphere에 클러스터가 있어야 합니다.
- Red Hat OpenShift Data Foundation과 같이 클러스터용 영구 스토리지 프로비저닝이 있어야 합니다.



중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- "100Gi" 용량이 필요합니다.

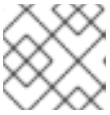
중요

테스트 결과, RHEL의 NFS 서버를 핵심 서비스용 스토리지 백엔드로 사용하는 데 문제가 있는 것으로 나타납니다. 여기에는 OpenShift Container Registry and Quay, 스토리지 모니터링을 위한 Prometheus, 로깅 스토리지를 위한 Elasticsearch가 포함됩니다. 따라서 RHEL NFS를 사용하여 핵심 서비스에서 사용하는 PV를 백업하는 것은 권장되지 않습니다.

마켓플레이스의 다른 NFS 구현에는 이러한 문제가 나타나지 않을 수 있습니다. 이러한 OpenShift Container Platform 핵심 구성 요소에 대해 완료된 테스트에 대한 자세한 내용은 개별 NFS 구현 공급업체에 문의하십시오.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.



참고

공유 스토리지를 사용하는 경우 보안 설정을 검토하여 외부 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```

참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 **claim** 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME             VERSION                             AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                                 True        False         False     6h50m
```

3.7.3.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

몇 분 후에 명령을 다시 실행하십시오.

3.7.3.3. VMware vSphere용 블록 레지스트리 스토리지 구성

클러스터 관리자로서 업그레이드 중에 이미지 레지스트리가 vSphere VMDK(Virtual Machine Disk)와 같은 블록 스토리지 유형을 사용할 수 있도록 허용하기 위해 **Recreate** 롤아웃 전략을 사용할 수 있습니다.

중요

블록 스토리지 볼륨이 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

프로세스

1. 다음 명령을 입력하여 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하고, **Recreate** 롤아웃 전략을 사용하도록 레지스트리를 패치하고, 복제본 **1** 개로 실행합니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 PV를 프로비저닝하고 해당 볼륨의 PVC를 생성합니다. 요청된 블록 볼륨은 RWO(ReadWriteOnce) 액세스 모드를 사용합니다.

   a. VMware vSphere **PersistentVolumeClaim** 개체를 정의하려면 다음 내용이 포함된 **pvc.yaml** 파일을 생성합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
```

```
  resources:
    requests:
      storage: 100Gi 4
```

1. **PersistentVolumeClaim** 개체를 표시하는 고유한 이름입니다.
2. **PersistentVolumeClaim** 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.
3. 영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.
4. 영구 볼륨 클레임의 크기입니다.

b. 다음 명령을 입력하여 파일에서 **PersistentVolumeClaim** 오브젝트를 생성합니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 구성을 편집하려면 다음 명령을 입력합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1. 사용자 정의 PVC를 생성하면 **image-registry-storage** PVC의 기본 자동 생성을 위해 **claim** 필드를 비워 둘 수 있습니다.

올바른 PVC를 참조하도록 레지스트리 스토리지를 구성하는 방법은 vSphere 용 레지스트리 구성을 참조하십시오.

3.7.3.4. Red Hat OpenShift Data Foundation에서 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.

- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Ceph RGW 오브젝트 스토리지를 제공합니다.

프로세스

1. **ocs-storagecluster-ceph-rgw** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage (1)
spec:
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

(1) 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 다음 명령을 입력하여 **경로** 호스트를 가져옵니다.

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 다음 명령을 입력하여 Ceph RGW 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.7.3.5. Red Hat OpenShift Data Foundation에서 Noobaa 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Noobaa 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Noobaa 오브젝트 스토리지를 제공합니다.

프로세스

1. **openshift-storage.noobaa.io** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
  namespace: openshift-storage 1
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

1 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 다음 명령을 입력하여 경로 호스트를 가져옵니다.

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 다음 명령을 입력하여 Nooba 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.7.4. Red Hat OpenShift Data Foundation에서 CephFS 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph

- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 CephFS 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

참고

CephFS는 PVC(영구 볼륨 클레임) 스토리지를 사용합니다. Ceph RGW 또는 Noobaa와 같은 다른 옵션을 사용할 수 있는 경우 이미지 레지스트리 스토리지에 PVC를 사용하지 않는 것이 좋습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- 오브젝트 스토리지 및 CephFS 파일 스토리지를 제공하기 위해 OpenShift Data Foundation Operator 를 설치했습니다.

프로세스

1. **cephfs** 스토리지 클래스를 사용할 PVC를 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 다음 명령을 입력하여 CephFS 파일 시스템 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.7.5. 추가 리소스

- 권장되는 구성 가능한 스토리지 기술
- OpenShift Data Foundation을 사용하도록 이미지 레지스트리 구성

3.8. RED HAT OPENSHIFT DATA FOUNDATION의 레지스트리 설정

Red Hat OpenShift Data Foundation 스토리지를 사용하도록 베어 메탈 및 vSphere에 OpenShift 이미지 레지스트리를 구성하려면 OpenShift Data Foundation을 설치한 다음 Ceph 또는 Noobaa를 사용하여 이미지 레지스트리를 구성해야 합니다.

3.8.1. Red Hat OpenShift Data Foundation에서 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Ceph RGW 오브젝트 스토리지를 제공합니다.

프로세스

1. **ocs-storagecluster-ceph-rgw** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage (1)
spec:
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

(1) 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 다음 명령을 입력하여 **경로** 호스트를 가져옵니다.

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 다음 명령을 입력하여 Ceph RGW 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.8.2. Red Hat OpenShift Data Foundation에서 Noobaa 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Noobaa 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Noobaa 오브젝트 스토리지를 제공합니다.

프로세스

1. **openshift-storage.noobaa.io** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
  namespace: openshift-storage (1)
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

(1) 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 다음 명령을 입력하여 경로 호스트를 가져옵니다.

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 다음 명령을 입력하여 Nooba 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

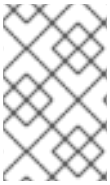
```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.8.3. Red Hat OpenShift Data Foundation에서 CephFS 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 CephFS 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.



참고

CephFS는 PVC(영구 볼륨 클레임) 스토리지를 사용합니다. Ceph RGW 또는 Noobaa와 같은 다른 옵션을 사용할 수 있는 경우 이미지 레지스트리 스토리지에 PVC를 사용하지 않는 것이 좋습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- 오브젝트 스토리지 및 CephFS 파일 스토리지를 제공하기 위해 OpenShift Data Foundation Operator 를 설치했습니다.

프로세스

1. **cephfs** 스토리지 클래스를 사용할 PVC를 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
```

```
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 다음 명령을 입력하여 CephFS 파일 시스템 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.8.4. 추가 리소스

- OpenShift Data Foundation을 사용하도록 이미지 레지스트리 구성
- Multicloud Object Gateway(NooBaa)의 성능 튜닝 가이드

3.9. NUTANIX의 레지스트리 구성

이 문서에 설명된 단계를 수행하여 사용자는 컨테이너 이미지 배포, 보안 및 액세스 제어를 최적화하여 OpenShift Container Platform에서 Nutanix 애플리케이션에 대한 강력한 기반을 활성화할 수 있습니다.

3.9.1. 설치 중 제거된 이미지 레지스트리

공유 가능한 개체 스토리지를 제공하지 않는 플랫폼에서 OpenShift Image Registry Operator는 자체적으로 **Removed**로 부트스트랩합니다. 이를 통해 **openshift-installer**가 이러한 플랫폼 유형에서 설치를 완료할 수 있습니다.

설치 후 **managementState**를 **Removed**에서 **Managed**로 전환하도록 Image Registry Operator 구성을 편집해야 합니다. 이 작업이 완료되면 스토리지를 구성해야 합니다.

3.9.2. 이미지 레지스트리의 관리 상태 변경

이미지 레지스트리를 시작하려면 Image Registry Operator 구성의 **managementState**를 **Removed**에서 **Managed**로 변경해야 합니다.

절차

- **managementState** Image Registry Operator 구성을 **Removed**에서 **Managed**로 변경합니다. 예를 들면 다음과 같습니다.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

3.9.3. 이미지 레지스트리 스토리지 구성

기본 스토리지를 제공하지 않는 플랫폼에서는 처음에 Image Registry Operator를 사용할 수 없습니다. 설치한 후에 스토리지를 사용하도록 레지스트리를 구성하여 Registry Operator를 사용 가능하도록 만들어야 합니다.

프로덕션 클러스터에 필요한 영구 볼륨을 구성하는 과정의 지침이 표시됩니다. 해당하는 경우, 프로덕션 환경 외 클러스터에서만 사용할 수 있는 저장 위치로서 빈 디렉터리를 구성하는 과정의 지침이 표시됩니다.

업그레이드 중에 **Recreate** 롤아웃 전략을 사용하여 이미지 레지스트리의 블록 스토리지 유형 사용을 허용하기 위한 추가 지침이 제공됩니다.

3.9.3.1. Nutanix의 레지스트리 스토리지 구성

클러스터 관리자는 설치한 후 스토리지를 사용하도록 레지스트리를 구성해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- Nutanix에 클러스터가 있어야 합니다.
- Red Hat OpenShift Data Foundation과 같이 클러스터용 영구 스토리지를 프로비저닝합니다.

중요

OpenShift Container Platform은 복제본이 하나만 있는 경우 이미지 레지스트리 스토리지에 대한 **ReadWriteOnce** 액세스를 지원합니다. **ReadWriteOnce** 액세스에는 레지스트리가 **Recreate** 롤아웃 전략을 사용해야 합니다. 두 개 이상의 복제본으로 고 가용성을 지원하는 이미지 레지스트리를 배포하려면 **ReadWriteMany** 액세스가 필요합니다.

- 100Gi 용량이 있어야 합니다.

프로세스

1. 스토리지를 사용하도록 레지스트리를 구성하기 위해 **configs.imageregistry/cluster** 리소스에서 **spec.storage.pvc**를 변경합니다.

참고

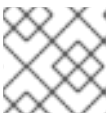
공유 스토리지를 사용하는 경우 보안 설정을 검토하여 외부 액세스를 방지합니다.

2. 레지스트리 pod가 없는지 확인합니다.

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

출력 예

```
No resourses found in openshift-image-registry namespace
```



참고

출력에 레지스트리 Pod가 있는 경우 이 절차를 계속할 필요가 없습니다.

3. 레지스트리 구성을 확인합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 **image-registry-storage** 영구 볼륨 클레임 (PVC)을 자동으로 생성할 수 있도록 **claim** 필드를 비워 둡니다. PVC는 기본 스토리지 클래스를 기반으로 생성됩니다. 그러나 기본 스토리지 클래스에서 RADOS 블록 장치(RBD)와 같은 ReadWriteOnce(ReadWriteOnce) 볼륨을 제공할 수 있으므로 복제본이 두 개 이상될 때 문제가 발생할 수 있습니다.

4. **clusteroperator** 상태를 확인합니다.

```
$ oc get clusteroperator image-registry
```

출력 예

```
NAME              VERSION                              AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry    4.13                                 True        False         False     6h50m
```

3.9.3.2. 프로덕션 환경 외 클러스터에서 이미지 레지스트리의 스토리지 구성

이미지 레지스트리 Operator에 대한 스토리지를 구성해야 합니다. 프로덕션 환경 외 클러스터의 경우, 이미지 레지스트리를 빈 디렉터리로 설정할 수 있습니다. 이렇게 하는 경우 레지스트리를 다시 시작하면 모든 이미지가 손실됩니다.

프로세스

- 이미지 레지스트리 스토리지를 빈 디렉터리로 설정하려면 다음을 수행하십시오.

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

주의

프로덕션 환경 외 클러스터에 대해서만 이 옵션을 구성하십시오.

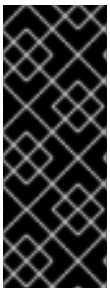
Image Registry Operator가 구성 요소를 초기화하기 전에 이 명령을 실행하면 **oc patch** 명령이 실패하며 다음 오류가 발생합니다.

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

몇 분 후에 명령을 다시 실행하십시오.

3.9.3.3. Nutanix 볼륨의 블록 레지스트리 스토리지 구성

클러스터 관리자로 업그레이드하는 동안 이미지 레지스트리가 Nutanix 볼륨과 같은 블록 스토리지 유형을 사용하도록 허용하려면 **Recreate** 롤아웃 전략을 사용할 수 있습니다.



중요

블록 스토리지 볼륨 또는 블록 영구 볼륨은 지원되지만 프로덕션 클러스터에서 이미지 레지스트리와 함께 사용하는 것은 권장되지 않습니다. 레지스트리가 블록 스토리지에 구성된 설치는 레지스트리가 둘 이상의 복제본을 가질 수 없기 때문에 가용성이 높지 않습니다.

이미지 레지스트리와 함께 블록 스토리지 볼륨을 사용하도록 선택하는 경우 파일 시스템 PVC(영구 볼륨 클레임)를 사용해야 합니다.

프로세스

1. 다음 명령을 입력하여 이미지 레지스트리 스토리지를 블록 스토리지 유형으로 설정하고, **Recreate** 롤아웃 전략을 사용하도록 레지스트리를 패치하고, 하나의 (**1**) 복제본으로만 실행됩니다.

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. 블록 스토리지 장치에 PV를 프로비저닝하고 해당 볼륨의 PVC를 생성합니다. 요청된 블록 볼륨은 RWO(ReadWriteOnce) 액세스 모드를 사용합니다.

   a. 다음 내용으로 **pvc.yaml** 파일을 생성하여 Nutanix **PersistentVolumeClaim** 오브젝트를 정의합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage 1
  namespace: openshift-image-registry 2
spec:
  accessModes:
  - ReadWriteOnce 3
  resources:
    requests:
      storage: 100Gi 4
```

   1. **PersistentVolumeClaim** 개체를 표시하는 고유한 이름입니다.
   2. **PersistentVolumeClaim** 오브젝트의 네임스페이스로 **openshift-image-registry**입니다.
   3. 영구 볼륨 클레임의 액세스 모드입니다. **ReadWriteOnce**를 사용하면 단일 노드에서 읽기 및 쓰기 권한으로 볼륨을 마운트할 수 있습니다.
   4. 영구 볼륨 클레임의 크기입니다.

   b. 다음 명령을 입력하여 파일에서 **PersistentVolumeClaim** 오브젝트를 생성합니다.

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 올바른 PVC를 참조하도록 레지스트리 구성을 편집하려면 다음 명령을 입력합니다.

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

출력 예

```
storage:
  pvc:
    claim: 1
```

1 사용자 정의 PVC를 생성하면 **image-registry-storage** PVC의 기본 자동 생성을 위해 **claim** 필드를 비워 둘 수 있습니다.

3.9.3.4. Red Hat OpenShift Data Foundation에서 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Ceph RGW 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Ceph RGW 오브젝트 스토리지를 제공합니다.

프로세스

1. **ocs-storagecluster-ceph-rgw** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage 1
spec:
```

```
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

1 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 다음 명령을 입력하여 **경로** 호스트를 가져옵니다.

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 다음 명령을 입력하여 Ceph RGW 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}' --type=merge
```

3.9.3.5. Red Hat OpenShift Data Foundation에서 Noobaa 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 Noobaa 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.
- **oc** CLI를 설치했습니다.
- OpenShift Data Foundation Operator 를 설치하여 오브젝트 스토리지 및 Noobaa 오브젝트 스토리지를 제공합니다.

프로세스

1. **openshift-storage.noobaa.io** 스토리지 클래스를 사용하여 오브젝트 버킷 클레임을 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
  namespace: openshift-storage (1)
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

(1) 또는 **openshift-image-registry** 네임스페이스를 사용할 수 있습니다.

2. 다음 명령을 입력하여 버킷 이름을 가져옵니다.

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 다음 명령을 입력하여 AWS 인증 정보를 가져옵니다.

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 다음 명령을 입력하여 **openshift-image-registry project**에서 새 버킷에 대한 AWS 자격 증명으로 **image-registry-private-configuration-user** 시크릿을 생성합니다.

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --namespace openshift-image-registry
```

5. 다음 명령을 입력하여 경로 호스트를 가져옵니다.

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 다음 명령을 입력하여 수신 인증서를 사용하는 구성 맵을 생성합니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq '.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n openshift-config
```

7. 다음 명령을 입력하여 Nooba 오브젝트 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":{"managementState":"Managed","replicas":2,"storage":{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedCA":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.9.4. Red Hat OpenShift Data Foundation에서 CephFS 스토리지를 사용하도록 이미지 레지스트리 Operator 구성

Red Hat OpenShift Data Foundation은 OpenShift 이미지 레지스트리에 사용할 수 있는 여러 스토리지 유형을 통합합니다.

- 공유 및 분산 파일 시스템 및 온-프레미스 개체 스토리지인 Ceph
- NooBaa: 멀티클라우드 오브젝트 게이트웨이를 제공

이 문서에서는 CephFS 스토리지를 사용하도록 이미지 레지스트리를 구성하는 절차를 간략하게 설명합니다.

참고

CephFS는 PVC(영구 볼륨 클레임) 스토리지를 사용합니다. Ceph RGW 또는 Noobaa와 같은 다른 옵션을 사용할 수 있는 경우 이미지 레지스트리 스토리지에 PVC를 사용하지 않는 것이 좋습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift Container Platform 웹 콘솔에 액세스할 수 있습니다.

- **oc** CLI를 설치했습니다.
- 오브젝트 스토리지 및 CephFS 파일 스토리지를 제공하기 위해 OpenShift Data Foundation Operator 를 설치했습니다.

프로세스

1. **cephfs** 스토리지 클래스를 사용할 PVC를 생성합니다. 예를 들면 다음과 같습니다.

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 다음 명령을 입력하여 CephFS 파일 시스템 스토리지를 사용하도록 이미지 레지스트리를 구성합니다.

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.9.5. 추가 리소스

- 권장되는 구성 가능한 스토리지 기술
- OpenShift Data Foundation을 사용하도록 이미지 레지스트리 구성

4장. 레지스트리 액세스

로그 및 메트릭보기, 레지스트리 보안 및 공개 등 레지스트리에 액세스하는 방법은 다음 섹션에 설명된 내용을 사용하십시오.

레지스트리에 직접 액세스하여 **podman** 명령을 시작할 수 있습니다. 이를 통해 **podman push** 또는 **podman pull**과 같은 작업을 사용하여 통합 레지스트리에서 이미지를 직접 푸시하거나 풀할 수 있습니다. 이렇게 하려면 **podman login** 명령을 사용하여 레지스트리에 로그인해야 합니다. 수행할 수있는 작업은 다음 섹션에 설명된대로 사용자 권한에 따라 달라집니다.

4.1. 사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- IDP(ID 공급자)를 구성해야합니다.
- 예를 들어 **podman pull** 명령을 사용하는 경우 이미지를 가져오려면 사용자에게 **registry-viewer** 역할이 있어야합니다. 이 역할을 추가하려면 다음 명령을 실행합니다.

```
$ oc policy add-role-to-user registry-viewer <user_name>
```

- 예를 들어 **podman push** 명령을 사용하는 경우와 같이 이미지를 작성하거나 푸시하는 경우:
  - 사용자에게 **registry-editor** 역할이 있어야 합니다. 이 역할을 추가하려면 다음 명령을 실행합니다.

```
$ oc policy add-role-to-user registry-editor <user_name>
```

  - 클러스터에 이미지를 내보낼 수 있는 기존 프로젝트가 있어야 합니다.

4.2. 클러스터에서 직접 레지스트리에 액세스

클러스터 내부에서 레지스트리에 액세스할 수 있습니다.

절차

내부 경로를 사용하여 클러스터에서 레지스트리에 액세스합니다.

1. 노드 이름을 가져와서 노드에 액세스합니다.

```
$ oc get nodes
```

```
$ oc debug nodes/<node_name>
```

2. 노드의 **oc** 및 **podman** 과 같은 툴에 대한 액세스를 활성화하려면 루트 디렉터리를 **/host** 로 변경하십시오.

```
sh-4.2# chroot /host
```

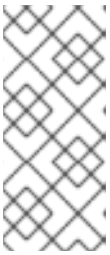
3. 액세스 토큰을 사용하여 컨테이너 이미지 레지스트리에 로그인합니다.

```
sh-4.2# oc login -u kubeadmin -p <password_from_install_log> https://api-int.
<cluster_name>.<base_domain>:6443
```

```
sh-4.2# podman login -u kubeadmin -p $(oc whoami -t) image-registry.openshift-image-registry.svc:5000
```

다음과 같은 로그인 확인 메시지가 표시되어야합니다.

```
Login Succeeded!
```

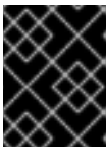


참고

사용자 이름에 모든 값을 지정할 수 있으므로 토큰에는 필요한 모든 정보가 포함됩니다. 콜론이 포함된 사용자 이름을 지정하면 로그인에 실패합니다.

이미지 레지스트리 Operator가 경로를 생성하므로 **default-route-openshift-image-registry.<cluster_name>**과 유사합니다.

4. 레지스트리에 대해 **podman pull** 및 **podman push** 작업을 수행합니다.



중요

모든 이미지를 가져올 수 있지만 **system:registry** 역할이 추가된 경우 프로젝트의 레지스트리에만 이미지를 푸시할 수 있습니다.

다음 예에서는 다음을 사용합니다.

구성 요소	값
<registry_ip>	**172.30.124.220**
<port>	**5000**
<project>	**openshift**
<image>	**image**
<tag>	생략됨 (기본값 **latest**)

a. 모든 이미지를 가져옵니다.

```
sh-4.2# podman pull <name.io>/<image>
```

b. **<registry_ip>:<port>/<project>/<image>** 형식으로 새 이미지에 태그를 지정합니다. OpenShift Container Platform이 레지스트리에 이미지를 올바르게 배치하고 나중에 액세스할 수 있도록 이 풀 사양에 프로젝트 이름이 표시되어야합니다.

```
sh-4.2# podman tag <name.io>/<image> image-registry.openshift-image-registry.svc:5000/openshift/<image>
```

참고

사용자가 이미지를 작성하거나 푸시할 수 있도록 지정된 프로젝트에 대한 **system:image-builder** 역할이 있어야합니다. 그렇지 않으면 다음 단계의 **podman push**가 실패합니다. 테스트를 위해 이미지를 푸시할 새 프로젝트를 만들 수 있습니다.

c. 새로 태그가 지정된 이미지를 레지스트리로 푸시합니다.

```
sh-4.2# podman push image-registry.openshift-image-
registry.svc:5000/openshift/<image>
```

참고

이미지를 내부 레지스트리로 내보내는 경우 리포지터리 이름은 < **project>/<name> 형식을 사용해야** 합니다. 리포지토리 이름에 여러 프로젝트 수준을 사용하면 인증 오류가 발생합니다.

4.3. 레지스트리 POD 상태 확인

클러스터 관리자는 **openshift-image-registry** 프로젝트에서 실행중인 이미지 레지스트리 pod를 나열하고 해당 상태를 확인할 수 있습니다.

전제 조건

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

- **openshift-image-registry** 프로젝트의 pod를 나열하고 상태를 확인합니다.

```
$ oc get pods -n openshift-image-registry
```

출력 예

```
NAME READY STATUS RESTARTS AGE
image-registry-79fb4469f6-llrln 1/1 Running 0 77m
node-ca-hjksc 1/1 Running 0 73m
node-ca-tftj6 1/1 Running 0 77m
node-ca-wb6ht 1/1 Running 0 77m
node-ca-zvt9q 1/1 Running 0 74m
```

4.4. 레지스트리 로그보기

oc logs 명령을 사용하여 레지스트리의 로그를 확인할 수 있습니다.

프로세스

- 배포에서 **oc logs** 명령을 사용하여 컨테이너 이미지 레지스트리의 로그를 표시합니다.

```
$ oc logs deployments/image-registry -n openshift-image-registry
```

출력 예

```
2015-05-01T19:48:36.300593110Z time="2015-05-01T19:48:36Z" level=info
msg="version=v2.0.0+unknown"
2015-05-01T19:48:36.303294724Z time="2015-05-01T19:48:36Z" level=info msg="redis not
configured" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
2015-05-01T19:48:36.303422845Z time="2015-05-01T19:48:36Z" level=info msg="using
inmemory layerinfo cache" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
2015-05-01T19:48:36.303433991Z time="2015-05-01T19:48:36Z" level=info msg="Using
OpenShift Auth handler"
2015-05-01T19:48:36.303439084Z time="2015-05-01T19:48:36Z" level=info msg="listening
on :5000" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
```

4.5. 레지스트리 메트릭 액세스

OpenShift Container 레지스트리는 Prometheus 메트릭에 대한 엔드 포인트를 제공합니다. Prometheus는 독립형 오픈 소스 시스템 모니터링 및 경고 툴킷입니다.

메트릭은 레지스트리 엔드 포인트의 ***/extensions/v2/metrics*** 경로에 표시됩니다.

프로세스

클러스터 역할을 사용하여 메트릭 쿼리를 실행하여 메트릭에 액세스할 수 있습니다.

클러스터 역할

1. 메트릭에 액세스하는 데 필요한 클러스터 역할이 없는 경우 클러스터 역할을 생성합니다.

```
$ cat <<EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-scraper
rules:
- apiGroups:
  - image.openshift.io
  resources:
  - registry/metrics
  verbs:
  - get
EOF
```

2. 이 역할을 사용자에게 추가하려면 다음 명령을 실행합니다.

```
$ oc adm policy add-cluster-role-to-user prometheus-scraper <username>
```

메트릭 쿼리

1. 사용자 토큰을 가져옵니다.

```
openshift:
$ oc whoami -t
```

2. 노드 또는 Pod 내부에서 메트릭 쿼리를 실행합니다. 예를 들면 다음과 같습니다.

```
$ curl --insecure -s -u <user>:<secret> \ 1
    https://image-registry.openshift-image-registry.svc:5000/extensions/v2/metrics | grep
imageregistry | head -n 20
```

출력 예

```
# HELP imageregistry_build_info A metric with a constant '1' value labeled by major, minor,
git commit & git version from which the image registry was built.
# TYPE imageregistry_build_info gauge
imageregistry_build_info{gitCommit="9f72191",gitVersion="v3.11.0+9f72191-135-
dirty",major="3",minor="11+"} 1
# HELP imageregistry_digest_cache_requests_total Total number of requests without scope
to the digest cache.
# TYPE imageregistry_digest_cache_requests_total counter
imageregistry_digest_cache_requests_total{type="Hit"} 5
imageregistry_digest_cache_requests_total{type="Miss"} 24
# HELP imageregistry_digest_cache_scoped_requests_total Total number of scoped
requests to the digest cache.
# TYPE imageregistry_digest_cache_scoped_requests_total counter
imageregistry_digest_cache_scoped_requests_total{type="Hit"} 33
imageregistry_digest_cache_scoped_requests_total{type="Miss"} 44
# HELP imageregistry_http_in_flight_requests A gauge of requests currently being served by
the registry.
# TYPE imageregistry_http_in_flight_requests gauge
imageregistry_http_in_flight_requests 1
# HELP imageregistry_http_request_duration_seconds A histogram of latencies for requests
to the registry.
# TYPE imageregistry_http_request_duration_seconds summary
imageregistry_http_request_duration_seconds{method="get",quantile="0.5"} 0.01296087
imageregistry_http_request_duration_seconds{method="get",quantile="0.9"} 0.014847248
imageregistry_http_request_duration_seconds{method="get",quantile="0.99"} 0.015981195
imageregistry_http_request_duration_seconds_sum{method="get"} 12.260727916000022
```

1 **<user>** 오브젝트는 임의의 값일 수 있지만 **<secret>** 태그는 사용자 토큰을 사용해야 합니다.

4.6. 추가 리소스

- 프로젝트의 Pod가 다른 프로젝트의 이미지를 참조하도록 허용하는 방법에 대한 자세한 내용은 프로젝트 간에 Pod가 이미지를 참조하도록 허용을 참조하십시오.
- **kubeadmin**은 삭제될 때 까지 레지스트리에 액세스할 수 있습니다. 자세한 내용은 kubeadmin 사용자 제거를 참조하십시오.
- ID 공급자 구성에 대한 자세한 내용은 ID 공급자 구성 이해를 참조하십시오.

5장. 레지스트리 공개

기본적으로 OpenShift 이미지 레지스트리는 TLS를 통해 트래픽을 제공하도록 클러스터 설치 중에 보호됩니다. 이전 OpenShift Container Platform 버전과 달리 레지스트리는 설치시 클러스터 외부에 공개되지 않습니다.

5.1. 기본 레지스트리 수동 노출

클러스터 내에서 기본 OpenShift 이미지 레지스트리에 로그인하는 대신 라우팅을 사용하여 외부 액세스 권한을 확보할 수 있습니다. 이 외부 액세스를 사용하면 경로 주소를 사용하여 클러스터 외부에서 레지스트리에 로그인하고 라우팅 호스트를 사용하여 기존 프로젝트에 이미지에 태그를 지정하고 푸시할 수 있습니다.

전제 조건

- 다음 사전 요구 사항이 자동으로 수행됩니다.
  - 레지스트리 Operator를 배포합니다.
  - Ingress Operator를 배포합니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

configs.imageregistry.operator.openshift.io 리소스에서 **defaultRoute** 매개변수를 사용하여 경로를 노출할 수 있습니다.

defaultRoute 를 사용하여 레지스트리를 공개하려면 다음을 수행합니다.

1. **defaultRoute** 를 **true** 로 설정합니다.

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":
{"defaultRoute":true}}' --type=merge
```

2. 기본 레지스트리 경로를 가져옵니다.

```
$ HOST=$(oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}')
```

3. Ingress Operator의 인증서를 가져옵니다.

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

4. 다음 명령을 사용하여 클러스터의 기본 인증서를 활성화하여 경로를 신뢰합니다.

```
$ sudo update-ca-trust enable
```

5. 기본 경로를 사용하여 podman으로 로그인합니다.

```
$ sudo podman login -u kubeadmin -p $(oc whoami -t) $HOST
```

5.2. 수동으로 보안 레지스트리 공개

클러스터 내에서 OpenShift 이미지 레지스트리에 로그인하는 대신 라우팅을 사용하여 외부 액세스 권한을 얻을 수 있습니다. 이를 통해 라우팅 주소를 사용하여 클러스터 외부에서 레지스트리에 로그인하고 라우팅 호스트를 사용하여 기존 프로젝트에 이미지를 태그 지정하거나 푸시할 수 있습니다.

전제 조건

- 다음 사전 요구 사항이 자동으로 수행됩니다.
  - 레지스트리 Operator를 배포합니다.
  - Ingress Operator를 배포합니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

configs.imageregistry.operator.openshift.io 리소스에서 **DefaultRoute** 매개 변수를 사용하거나 사용자 지정 라우팅을 사용하여 라우팅을 공개할 수 있습니다.

DefaultRoute를 사용하여 레지스트리를 공개하려면 다음을 수행합니다.

1. **DefaultRoute**를 **True**로 설정합니다.

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":
{"defaultRoute":true}}' --type=merge
```

2. **podman**으로 로그인합니다.

```
$ HOST=$(oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}')
```

```
$ podman login -u kubeadmin -p $(oc whoami -t) --tls-verify=false $HOST (1)
```

(1) **--tls-verify=false**는 클러스터의 기본 라우팅 인증서를 신뢰할 수없는 경우 필요합니다. Ingress Operator를 사용하여 신뢰할 수있는 사용자 지정 인증서를 기본 인증서로 설정할 수 있습니다.

사용자 지정 라우팅을 사용하여 레지스트리를 공개하려면 다음을 수행합니다.

1. 라우팅의 TLS 키로 보안 시크릿을 만듭니다.

```
$ oc create secret tls public-route-tls \
    -n openshift-image-registry \
    --cert=</path/to/tls.crt> \
    --key=</path/to/tls.key>
```

이 단계는 선택 사항입니다. 보안 시크릿을 생성하지 않으면 라우팅은 Ingress Operator의 기본 TLS 구성을 사용합니다.

2. 레지스트리 Operator에서 다음을 수행합니다.

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

```
spec:
  routes:
```

```
    - name: public-routes
      hostname: myregistry.mycorp.organization
      secretName: public-route-tls
...
```

참고

레지스트리 라우팅에 대한 사용자 지정 TLS 구성을 제공하는 경우에만 **secretName**을 설정합니다.

문제 해결

- TLS 시크릿을 생성하는 동안 오류 발생