OpenShift Container Platform 4.5

Registry

Configuring registries for OpenShift Container Platform

Last Updated: 2021-06-29

OpenShift Container Platform 4.5 Registry

Configuring registries for OpenShift Container Platform

法律通告

摘要

This document provides instructions for configuring and managing the internal registry for OpenShift Container Platform. It also provides a general overview of registries associated with OpenShift Container Platform.

目录

第 1 章 IMAGE REGISTRY

1.1. INTEGRATED OPENSHIFT CONTAINER PLATFORM REGISTRY

OpenShift Container Platform provides a built-in container image registry that runs as a standard workload on the cluster. The registry is configured and managed by an infrastructure Operator. It provides an out-of-the-box solution for users to manage the images that run their workloads, and runs on top of the existing cluster infrastructure. This registry can be scaled up or down like any other cluster workload and does not require specific infrastructure provisioning. In addition, it is integrated into the cluster user authentication and authorization system, which means that access to create and retrieve images is controlled by defining user permissions on the image resources.

The registry is typically used as a publication target for images built on the cluster, as well as being a source of images for workloads running on the cluster. When a new image is pushed to the registry, the cluster is notified of the new image and other components can react to and consume the updated image.

Image data is stored in two locations. The actual image data is stored in a configurable storage location, such as cloud storage or a filesystem volume. The image metadata, which is exposed by the standard cluster APIs and is used to perform access control, is stored as standard API resources, specifically images and imagestreams.

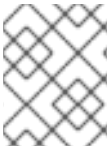
Additional resources

- Image Registry Operator in OpenShift Container Platform

第 2 章 IMAGE REGISTRY OPERATOR IN OPENSHIFT CONTAINER PLATFORM

2.1. IMAGE REGISTRY ON CLOUD PLATFORMS AND OPENSTACK

The Image Registry Operator installs a single instance of the OpenShift Container Platform registry, and manages all registry configuration, including setting up registry storage.



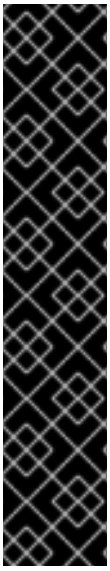
注意

Storage is only automatically configured when you install an installer-provisioned infrastructure cluster on AWS, GCP, Azure, or OpenStack.

After the control plane deploys, the Operator will create a default **configs.imageregistry.operator.openshift.io** resource instance based on configuration detected in the cluster.

If insufficient information is available to define a complete **configs.imageregistry.operator.openshift.io** resource, the incomplete resource will be defined and the Operator will update the resource status with information about what is missing.

The Image Registry Operator runs in the **openshift-image-registry** namespace, and manages the registry instance in that location as well. All configuration and workload resources for the registry reside in that namespace.



重要

The Image Registry Operator's behavior for managing the pruner is orthogonal to the **managementState** specified on the **ClusterOperator** object for the Image Registry Operator. If the Image Registry Operator is not in the **Managed** state, the image pruner can still be configured and managed by the **Pruning** custom resource.

However, the **managementState** of the Image Registry Operator alters the behavior of the deployed image pruner job:

- **Managed**: the **--prune-registry** flag for the image pruner is set to **true**.
- **Removed**: the **--prune-registry** flag for the image pruner is set to **false**, meaning it only prunes image metatdata in etcd.
- **Unmanaged**: the **--prune-registry** flag for the image pruner is set to **false**.

2.2. IMAGE REGISTRY ON BARE METAL AND VSPHERE

2.2.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.

注意

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing configs.imageregistry.operator.openshift.io."

2.3. IMAGE REGISTRY OPERATOR CONFIGURATION PARAMETERS

The **configs.imageregistry.operator.openshift.io** resource offers the following configuration parameters.

Parameter	Description
managementState	**Managed**: The Operator updates the registry as configuration resources are updated. **Unmanaged**: The Operator ignores changes to the configuration resources. **Removed**: The Operator removes the registry instance and tear down any storage that the Operator provisioned.
logging	Sets **loglevel** of the registry instance.
httpSecret	Value needed by the registry to secure uploads, generated by default.
proxy	Defines the Proxy to be used when calling master API and upstream registries.
storage	**Storagetype**: Details for configuring registry storage, for example S3 bucket coordinates. Normally configured by default.
readOnly	Indicates whether the registry instance should reject attempts to push new images or delete existing ones.
requests	API Request Limit details. Controls how many parallel requests a given registry instance will handle before queuing additional requests.
defaultRoute	Determines whether or not an external route is defined using the default hostname. If enabled, the route uses re-encrypt encryption. Defaults to false.
routes	Array of additional routes to create. You provide the hostname and certificate for the route.
replicas	Replica count for the registry.

2.4. ENABLE THE IMAGE REGISTRY DEFAULT ROUTE WITH THE CUSTOM RESOURCE DEFINITION

In OpenShift Container Platform, the **Registry** Operator controls the registry feature. The Operator is defined by the **configs.imageregistry.operator.openshift.io** Custom Resource Definition (CRD).

If you need to automatically enable the Image Registry default route, patch the Image Registry Operator CRD.

Procedure

- Patch the Image Registry Operator CRD:

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

2.5. CONFIGURING ADDITIONAL TRUST STORES FOR IMAGE REGISTRY ACCESS

The **image.config.openshift.io/cluster** custom resource can contain a reference to a config map that contains additional certificate authorities to be trusted during image registry access.

Prerequisites

- The certificate authorities (CA) must be PEM-encoded.

Procedure

You can create a config map in the **openshift-config** namespace and use its name in **AdditionalTrustedCA** in the **image.config.openshift.io** custom resource to provide additional CAs that should be trusted when contacting external registries.

The config map key is the host name of a registry with the port for which this CA is to be trusted, and the base64-encoded certificate is the value, for each additional registry CA to trust.

Image registry CA config map example

```
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 If the registry has the port, such as **registry-with-port.example.com:5000**, **:** should be replaced with **..**.

You can configure additional CAs with the following procedure.

1. To configure an additional CA:

```
$ 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
```

2.6. CONFIGURING STORAGE CREDENTIALS FOR THE IMAGE REGISTRY OPERATOR

In addition to the **configs.imageregistry.operator.openshift.io** and ConfigMap resources, storage credential configuration is provided to the Operator by a separate secret resource located within the **openshift-image-registry** namespace.

The **image-registry-private-configuration-user** secret provides credentials needed for storage access and management. It overrides the default credentials used by the Operator, if default credentials were found.

Procedure

- Create an OpenShift Container Platform secret that contains the required keys.

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

2.7. ADDITIONAL RESOURCES

- Configuring the registry for AWS user-provisioned infrastructure
- Configuring the registry for GCP user-provisioned infrastructure
- Configuring the registry for Azure user-provisioned infrastructure
- Configuring the registry for bare metal
- Configuring the registry for vSphere

第 3 章 SETTING UP AND CONFIGURING THE REGISTRY

3.1. CONFIGURING THE REGISTRY FOR AWS USER-PROVISIONED INFRASTRUCTURE

3.1.1. Configuring a secret for the Image Registry Operator

In addition to the **configs.imageregistry.operator.openshift.io** and ConfigMap resources, configuration is provided to the Operator by a separate secret resource located within the **openshift-image-registry** namespace.

The **image-registry-private-configuration-user** secret provides credentials needed for storage access and management. It overrides the default credentials used by the Operator, if default credentials were found.

For S3 on AWS storage, the secret is expected to contain two keys:

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

Procedure

- Create an OpenShift Container Platform secret that contains the required keys.

```
$ 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. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an Amazon S3 bucket and the Registry Operator will automatically configure storage.

If the Registry Operator cannot create an S3 bucket and automatically configure storage, you can create an S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.
- For Amazon S3 storage, the secret is expected to contain two keys:
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

Procedure

Use the following procedure if the Registry Operator cannot create an S3 bucket and automatically configure storage.

1. Set up a Bucket Lifecycle Policy to abort incomplete multipart uploads that are one day old.

2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

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

Example configuration

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

警告

To secure your registry images in AWS, block public access to the S3 bucket.

3.1.3. Image Registry Operator configuration parameters for AWS S3

The following configuration parameters are available for AWS S3 registry storage.

ImageRegistryConfigStorageS3 holds the information to configure the registry to use the AWS S3 service for back-end storage. See the S3 storage driver documentation for more information.

Parameter	Description
bucket	Bucket is the bucket name in which you want to store the registry's data. It is optional and is generated if not provided.
region	Region is the AWS region in which your bucket exists. It is optional and is set based on the installed AWS Region.
regionEndpoint	RegionEndpoint is the endpoint for S3 compatible storage services. It is optional and defaults based on the Region that is provided.
virtualHostedStyle	VirtualHostedStyle enables using S3 virtual hosted style bucket paths with a custom RegionEndpoint. It is optional and defaults to false. Set this parameter to deploy OpenShift Container Platform to hidden regions.
encrypt	Encrypt specifies whether or not the registry stores the image in encrypted format. It is optional and defaults to false.
keyID	KeyID is the KMS key ID to use for encryption. It is optional. Encrypt must be true, or this parameter is ignored.
ImageRegistryConfigStorageS3CloudFront	CloudFront configures Amazon Cloudfront as the storage middleware in a registry. It is optional.

3.2. CONFIGURING THE REGISTRY FOR GCP USER-PROVISIONED INFRASTRUCTURE

3.2.1. Configuring a secret for the Image Registry Operator

In addition to the **configs.imageregistry.operator.openshift.io** and ConfigMap resources, configuration is provided to the Operator by a separate secret resource located within the **openshift-image-registry** namespace.

The **image-registry-private-configuration-user** secret provides credentials needed for storage access and management. It overrides the default credentials used by the Operator, if default credentials were found.

For GCS on GCP storage, the secret is expected to contain one key whose value is the contents of a credentials file provided by GCP:

- **REGISTRY_STORAGE_GCS_KEYFILE**

Procedure

- Create an OpenShift Container Platform secret that contains the required keys.

```
$ 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. Registry storage for GCP with user-provisioned infrastructure

You must set up the storage medium manually and configure the settings in the registry custom resource (CR).

Prerequisites

- A cluster on GCP with user-provisioned infrastructure.
- To configure registry storage for GCP, you need to provide Registry Operator cloud credentials.
- For GCS on GCP storage, the secret is expected to contain one key whose value is the contents of a credentials file provided by GCP:
  - **REGISTRY_STORAGE_GCS_KEYFILE**

3.2.3. Image Registry Operator configuration parameters for GCP GCS

Procedure

The following configuration parameters are available for GCP GCS registry storage.

Parameter	Description
bucket	Bucket is the bucket name in which you want to store the registry's data. It is optional and is generated if not provided.

Parameter	Description
region	Region is the GCS location in which your bucket exists. It is optional and is set based on the installed GCS Region.
projectID	ProjectID is the Project ID of the GCP project that this bucket should be associated with. It is optional.
keyID	KeyID is the KMS key ID to use for encryption. It is optional because buckets are encrypted by default on GCP. This allows for the use of a custom encryption key.

3.3. CONFIGURING THE REGISTRY FOR AZURE USER-PROVISIONED INFRASTRUCTURE

3.3.1. Configuring a secret for the Image Registry Operator

In addition to the **configs.imageregistry.operator.openshift.io** and ConfigMap resources, configuration is provided to the Operator by a separate secret resource located within the **openshift-image-registry** namespace.

The **image-registry-private-configuration-user** secret provides credentials needed for storage access and management. It overrides the default credentials used by the Operator, if default credentials were found.

For Azure registry storage, the secret is expected to contain one key whose value is the contents of a credentials file provided by Azure:

- **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

Procedure

- Create an OpenShift Container Platform secret that contains the required key.

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

3.3.2. Configuring registry storage for Azure

During installation, your cloud credentials are sufficient to create Azure Blob Storage, and the Registry Operator automatically configures storage.

Prerequisites

- A cluster on Azure with user-provisioned infrastructure.
- To configure registry storage for Azure, provide Registry Operator cloud credentials.
- For Azure storage the secret is expected to contain one key:

- **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

Procedure

1. Create an Azure storage container.
2. Fill in the storage configuration in **configs.imageregistry.operator.openshift.io/cluster**:

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

Example configuration

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

3.4. CONFIGURING THE REGISTRY FOR BARE METAL

3.4.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.

注意

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing configs.imageregistry.operator.openshift.io."

3.4.2. Changing the image registry's management state

To start the image registry, you must change the Image Registry Operator configuration's **managementState** from **Removed** to **Managed**.

Procedure

- Change **managementState** Image Registry Operator configuration from **Removed** to **Managed**. For example:

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

3.4.3. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

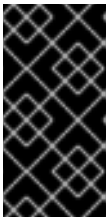
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

3.4.4. Configuring registry storage for bare metal

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on bare metal.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



重要

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have 100Gi capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

注意

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

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

注意

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

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

-

Example output

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

Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

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

3.4.5. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

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

警告

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

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

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

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

Then, change the line

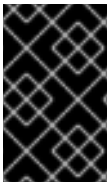
```
managementState: Removed
```

to

```
managementState: Managed
```

3.4.6. Configuring block registry storage for bare metal

To allow the image registry to use block storage types during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



重要

Block storage volumes are supported but not recommended for use with the image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only one (**1**) replica:

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

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.

3. Edit the registry configuration so that it references the correct PVC.

3.4.7. Additional resources

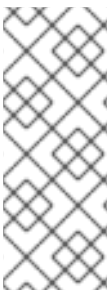
For more details about configuring registry storage for bare metal, see Recommended configurable storage technology.

3.5. CONFIGURING THE REGISTRY FOR VSPHERE

3.5.1. Image registry removed during installation

On platforms that do not provide shareable object storage, the OpenShift Image Registry Operator bootstraps itself as **Removed**. This allows **openshift-installer** to complete installations on these platform types.

After installation, you must edit the Image Registry Operator configuration to switch the **managementState** from **Removed** to **Managed**.



注意

The Prometheus console provides an **ImageRegistryRemoved** alert, for example:

"Image Registry has been removed. **ImageStreamTags**, **BuildConfigs** and **DeploymentConfigs** which reference **ImageStreamTags** may not work as expected. Please configure storage and update the config to **Managed** state by editing configs.imageregistry.operator.openshift.io."

3.5.2. Changing the image registry's management state

To start the image registry, you must change the Image Registry Operator configuration's **managementState** from **Removed** to **Managed**.

Procedure

- Change **managementState** Image Registry Operator configuration from **Removed** to **Managed**. For example:

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

3.5.2.1. Image registry storage configuration

The Image Registry Operator is not initially available for platforms that do not provide default storage. After installation, you must configure your registry to use storage so that the Registry Operator is made available.

Instructions are shown for configuring a persistent volume, which is required for production clusters. Where applicable, instructions are shown for configuring an empty directory as the storage location, which is available for only non-production clusters.

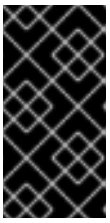
Additional instructions are provided for allowing the image registry to use block storage types by using the **Recreate** rollout strategy during upgrades.

3.5.3. Configuring registry storage for VMware vSphere

As a cluster administrator, following installation you must configure your registry to use storage.

Prerequisites

- Cluster administrator permissions.
- A cluster on VMware vSphere.
- Persistent storage provisioned for your cluster, such as Red Hat OpenShift Container Storage.



重要

OpenShift Container Platform supports **ReadWriteOnce** access for image registry storage when you have only one replica. To deploy an image registry that supports high availability with two or more replicas, **ReadWriteMany** access is required.

- Must have "100Gi" capacity.

重要

Testing shows issues with using the NFS server on RHEL as storage backend for core services. This includes the OpenShift Container Registry and Quay, Prometheus for monitoring storage, and Elasticsearch for logging storage. Therefore, using RHEL NFS to back PVs used by core services is not recommended.

Other NFS implementations on the marketplace might not have these issues. Contact the individual NFS implementation vendor for more information on any testing that was possibly completed against these OpenShift Container Platform core components.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.

注意

When using shared storage, review your security settings to prevent outside access.

2. Verify that you do not have a registry pod:

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

注意

If the storage type is **emptyDIR**, the replica number cannot be greater than **1**.

3. Check the registry configuration:

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

Example output

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

1 Leave the **claim** field blank to allow the automatic creation of an **image-registry-storage** PVC.

4. Check the **clusteroperator** status:

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

3.5.4. Configuring storage for the image registry in non-production clusters

You must configure storage for the Image Registry Operator. For non-production clusters, you can set the image registry to an empty directory. If you do so, all images are lost if you restart the registry.

Procedure

1. To set the image registry storage to an empty directory:

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

警告

Configure this option for only non-production clusters.

If you run this command before the Image Registry Operator initializes its components, the **oc patch** command fails with the following error:

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

Wait a few minutes and run the command again.

2. Ensure that your registry is set to managed to enable building and pushing of images.

- Run:

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

Then, change the line

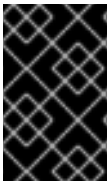
```
managementState: Removed
```

to

```
managementState: Managed
```

3.5.5. Configuring block registry storage for VMware vSphere

To allow the image registry to use block storage types such as vSphere Virtual Machine Disk (VMDK) during upgrades as a cluster administrator, you can use the **Recreate** rollout strategy.



重要

Block storage volumes are supported but not recommended for use with image registry on production clusters. An installation where the registry is configured on block storage is not highly available because the registry cannot have more than one replica.

Procedure

1. To set the image registry storage as a block storage type, patch the registry so that it uses the **Recreate** rollout strategy and runs with only **1** replica:

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

2. Provision the PV for the block storage device, and create a PVC for that volume. The requested block volume uses the ReadWriteOnce (RWO) access mode.

   a. Create a **pvc.yaml** file with the following contents to define a VMware vSphere **PersistentVolumeClaim** object:

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

   (1) A unique name that represents the **PersistentVolumeClaim** object.

   (2) The access mode of the PersistentVolumeClaim. With **ReadWriteOnce**, the volume can be mounted with read and write permissions by a single node.

   (3) The size of the PersistentVolumeClaim.

   b. Create the **PersistentVolumeClaim** object from the file:

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

3. Edit the registry configuration so that it references the correct PVC:

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

Example output

```
storage:
  pvc:
    claim: (1)
```

(1) Creating a custom PVC allows you to leave the **claim** field blank for the default automatic creation of an **image-registry-storage** PVC.

For instructions about configuring registry storage so that it references the correct PVC, see Configuring the registry for vSphere.

3.5.6. Additional resources

For more details about configuring registry storage for vSphere, see Recommended configurable storage technology.

第 4 章 REGISTRY OPTIONS

OpenShift Container Platform can build images from your source code, deploy them, and manage their lifecycle. To enable this, OpenShift Container Platform provides an internal, integrated container image registry that can be deployed in your OpenShift Container Platform environment to locally manage images.

4.1. INTEGRATED OPENSHIFT CONTAINER PLATFORM REGISTRY

OpenShift Container Platform provides a built-in container image registry that runs as a standard workload on the cluster. The registry is configured and managed by an infrastructure Operator. It provides an out-of-the-box solution for users to manage the images that run their workloads, and runs on top of the existing cluster infrastructure. This registry can be scaled up or down like any other cluster workload and does not require specific infrastructure provisioning. In addition, it is integrated into the cluster user authentication and authorization system, which means that access to create and retrieve images is controlled by defining user permissions on the image resources.

The registry is typically used as a publication target for images built on the cluster, as well as being a source of images for workloads running on the cluster. When a new image is pushed to the registry, the cluster is notified of the new image and other components can react to and consume the updated image.

Image data is stored in two locations. The actual image data is stored in a configurable storage location, such as cloud storage or a filesystem volume. The image metadata, which is exposed by the standard cluster APIs and is used to perform access control, is stored as standard API resources, specifically images and imagestreams.

4.2. THIRD-PARTY REGISTRIES

OpenShift Container Platform can create containers using images from third-party registries, but it is unlikely that these registries offer the same image notification support as the integrated OpenShift Container Platform registry. In this situation, OpenShift Container Platform will fetch tags from the remote registry upon imagestream creation.

Refreshing the fetched tags is as simple as running **oc import-image <stream>**. When new images are detected, the previously described build and deployment reactions occur.

4.2.1. Authentication

OpenShift Container Platform can communicate with registries to access private image repositories using credentials supplied by the user. This allows OpenShift Container Platform to push and pull images to and from private repositories.

4.3. RED HAT QUAY REGISTRIES

If you need an enterprise-quality container image registry, Red Hat Quay is available both as a hosted service and as software you can install in your own data center or cloud environment. Advanced registry features in Red Hat Quay include geo-replication, image scanning, and the ability to roll back images.

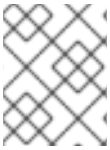
Visit the Quay.io site to set up your own hosted Quay registry account. After that, follow the Quay Tutorial to log in to the Quay registry and start managing your images.

You can access your Red Hat Quay registry from OpenShift Container Platform like any remote container image registry.

4.4. AUTHENTICATION ENABLED RED HAT REGISTRY

All container images available through the Container images section of the Red Hat Ecosystem Catalog are hosted on an image registry, **registry.redhat.io**.

The registry, **registry.redhat.io**, requires authentication for access to images and hosted content on OpenShift Container Platform. Following the move to the new registry, the existing registry will be available for a period of time.



注意

OpenShift Container Platform pulls images from **registry.redhat.io**, so you must configure your cluster to use it.

The new registry uses standard OAuth mechanisms for authentication, with the following methods:

- **Authentication token.** Tokens, which are generated by administrators, are service accounts that give systems the ability to authenticate against the container image registry. Service accounts are not affected by changes in user accounts, so the token authentication method is reliable and resilient. This is the only supported authentication option for production clusters.
- **Web username and password.** This is the standard set of credentials you use to log in to resources such as **access.redhat.com**. While it is possible to use this authentication method with OpenShift Container Platform, it is not supported for production deployments. Restrict this authentication method to stand-alone projects outside OpenShift Container Platform.

You can use **podman login** with your credentials, either username and password or authentication token, to access content on the new registry.

All imagestreams point to the new registry, which uses the installation pull secret to authenticate.

You must place your credentials in either of the following places:

- **openshift namespace**. Your credentials must exist in the OpenShift namespace so that the imagestreams in the **openshift** namespace can import.
- **Your host**. Your credentials must exist on your host because Kubernetes uses the credentials from your host when it goes to pull images.

第 5 章 ACCESSING THE REGISTRY

Use the following sections for instructions on accessing the registry, including viewing logs and metrics, as well as securing and exposing the registry.

You can access the registry directly to invoke **podman** commands. This allows you to push images to or pull them from the integrated registry directly using operations like **podman push** or **podman pull**. To do so, you must be logged in to the registry using the **oc login** command. The operations you can perform depend on your user permissions, as described in the following sections.

5.1. PREREQUISITES

- You must have configured an identity provider (IDP).
- For pulling images, for example when using the **podman pull** command, the user must have the **registry-viewer** role. To add this role:

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

- For writing or pushing images, for example when using the **podman push** command, the user must have the **registry-editor** role. To add this role:

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

5.2. ACCESSING REGISTRY DIRECTLY FROM THE CLUSTER

You can access the registry from inside the cluster.

Procedure

Access the registry from the cluster by using internal routes:

1. Access the node by getting the node's address:

```
$ oc get nodes
```

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

2. To enable access to tools such as **oc** and **podman** on the node, run the following command:

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

3. Log in to the container image registry by using your access token:

```
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
```

You should see a message confirming login, such as:

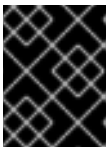
```
Login Succeeded!
```

注意

You can pass any value for the user name; the token contains all necessary information. Passing a user name that contains colons will result in a login failure.

Since the Image Registry Operator creates the route, it will likely be similar to **default-route-openshift-image-registry.<cluster_name>**.

4. Perform **podman pull** and **podman push** operations against your registry:



重要

You can pull arbitrary images, but if you have the **system:registry** role added, you can only push images to the registry in your project.

In the following examples, use:

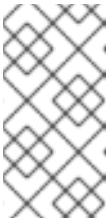
Component	Value
<registry_ip>	**172.30.124.220**
<port>	**5000**
<project>	**openshift**
<image>	**image**
<tag>	omitted (defaults to **latest**)

a. Pull an arbitrary image:

```
$ podman pull name.io/image
```

b. Tag the new image with the form **<registry_ip>:<port>/<project>/<image>**. The project name must appear in this pull specification for OpenShift Container Platform to correctly place and later access the image in the registry:

```
$ podman tag name.io/image image-registry.openshift-image-
registry.svc:5000/openshift/image
```



注意

You must have the **system:image-builder** role for the specified project, which allows the user to write or push an image. Otherwise, the **podman push** in the next step will fail. To test, you can create a new project to push the image.

c. Push the newly tagged image to your registry:

```
$ podman push image-registry.openshift-image-registry.svc:5000/openshift/image
```

5.3. CHECKING THE STATUS OF THE REGISTRY PODS

As a cluster administrator, you can list the image registry pods running in the **openshift-image-registry** project and check their status.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).

Procedure

1. List the pods in the **openshift-image-registry** project and view their status:

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

Example output

```
NAME READY STATUS RESTARTS AGE
cluster-image-registry-operator-764bd7f846-qqtpb 1/1 Running 0 78m
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
```

5.4. VIEWING REGISTRY LOGS

You can view the logs for the registry by using the **oc logs** command.

Procedure

1. Use the **oc logs** command with deployments to view the logs for the container image registry:

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

Example output

```
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
```

5.5. ACCESSING REGISTRY METRICS

The OpenShift Container Registry provides an endpoint for Prometheus metrics. Prometheus is a stand-alone, open source systems monitoring and alerting toolkit.

The metrics are exposed at the ***/extensions/v2/metrics*** path of the registry endpoint.

Procedure

There are two ways in which you can access the metrics, running a metrics query or using the cluster role.

Metrics query

1. Run a metrics query, for example:

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

Example output

```
# 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>** can be arbitrary, but **<secret>** must match the value specified in the registry configuration.

Cluster role

1. Create a cluster role if you do not already have one to access the metrics:

```
$ 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. Add this role to a user, run the following command:

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

3. Access the metrics using cluster role. The part of the configuration file responsible for metrics should look like this:

```
openshift:
  version: 1.0
  metrics:
    enabled: true
...
```

5.6. ADDITIONAL RESOURCES

- For more information on allowing pods in a project to reference images in another project, see Allowing pods to reference images across projects .
- A **kubeadmin** can access the registry until deleted. See Removing the kubeadmin user for more information.
- For more information on configuring an identity provider, see Understanding identity provider configuration.

第 6 章 EXPOSING THE REGISTRY

By default, the OpenShift Container Platform registry is secured during cluster installation so that it serves traffic through TLS. Unlike previous versions of OpenShift Container Platform, the registry is not exposed outside of the cluster at the time of installation.

6.1. EXPOSING A SECURE REGISTRY MANUALLY

Instead of logging in to the OpenShift Container Platform registry from within the cluster, you can gain external access to it by exposing it with a route. This allows you to log in to the registry from outside the cluster using the route address, and to tag and push images using the route host.

Prerequisites:

- The following prerequisites are automatically performed:
  - Deploy the Registry Operator.
  - Deploy the Ingress Operator.

Procedure

You can expose the route by using **DefaultRoute** parameter in the **configs.imageregistry.operator.openshift.io** resource or by using custom routes.

To expose the registry using **DefaultRoute**:

1. Set **DefaultRoute** to **True**:

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

2. Log in with **podman**:

```
$ 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** is needed if the cluster's default certificate for routes is untrusted. You can set a custom, trusted certificate as the default certificate with the Ingress Operator.

To expose the registry using custom routes:

1. Create a secret with your route's TLS keys:

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

This step is optional. If you do not create a secret, the route uses the default TLS configuration from the Ingress Operator.

2. On the Registry Operator:

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

注意

Only set **secretName** if you are providing a custom TLS configuration for the registry's route.