OpenShift Container Platform 4.1

Registry

Configuring registries for OpenShift Container Platform 4.1

Last Updated: 2020-02-28

OpenShift Container Platform 4.1 Registry

Configuring registries for OpenShift Container Platform 4.1

Legal Notice

Abstract

This document provides instructions for configuring and managing the internal registry for OpenShift Container Platform 4.1. It also provides a general overview of registries associated with OpenShift Container Platform 4.1.

Table of Contents

CHAPTER 1. IMAGE REGISTRY

1.1. INTEGRATED OPENSHIFT CONTAINER PLATFORM REGISTRY

OpenShift Container Platform provides a built in container image registry which 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 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.

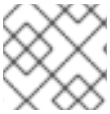
Image data is stored in two locations. The actual image data is stored in a configurable storage location such as cloud storage or a filesystem volume. The image metadata, which is exposed by the standard cluster APIs and is used to perform access control, is stored as standard API resources, specifically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

Additional resources

- Image Registry Operator in OpenShift Container Platform

CHAPTER 2. IMAGE REGISTRY OPERATOR IN OPENSHIFT CONTAINER PLATFORM

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



NOTE

Storage is only automatically configured when you install on Amazon Web Services.

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.

Prerequisites

- Deploy an OpenShift Container Platform cluster.

2.1. 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.

Parameter	Description
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.

2.2. IMAGE REGISTRY OPERATOR CONFIGURATION RESOURCES

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

Prerequisites

- The CAs must be PEM-encoded.

Procedure

You can create a ConfigMap in the **openshift-config** namespace and use its name in **AdditionalTrustedCA** in the **image.config.openshift.io** resource to provide additional CAs that should be trusted when contacting external registries. The key is the host name of a registry with the port for which this CA is to be trusted. The **image-registry-private-configuration**(Secret) provides credentials needed for storage access and management. It overrides the default credentials used by the Operator, if default credentials were found.

Image registry CA 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 **..**.

For S3 storage the ConfigMap is expected to contain two keys:

- REGISTRY_STORAGE_S3_ACCESSKEY
- REGISTRY_STORAGE_S3_SECRETKEY

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. Check your image inside the **image-registry** pod:

```
$ oc rsh image-registry-xxxxx
sh-4.2
$ ls /etc/pki/ca-trust/source/anchors
<external_registry_address> image-registry.openshift-image-registry.svc..5000 image-
registry.openshift-image-registry.svc.cluster.local..5000
```

2.3. 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}}'
```

2.4. CONFIGURING IMAGE REGISTRY STORAGE

2.4.1. Configuring registry storage for AWS with user-provisioned infrastructure

During installation, your cloud credentials are sufficient to create an 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 a S3 bucket and configure storage with the following procedure.

Prerequisites

- A cluster on AWS with user-provisioned infrastructure.

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

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

WARNING

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

2.4.2. 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.
- A provisioned persistent volume (PV) with **ReadWriteMany** access mode, such as **NFS**.
- Must have "100Gi" capacity.

Procedure

1. To configure your registry to use storage, change the **spec.storage.pvc** in the **configs.imageregistry/cluster** resource.
2. Verify you do not have a registry pod:

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

NOTE

If the storage type is **emptyDIR**, the replica number can not be greater than **1**. If the storage type is **NFS**, and you want to scale up the registry Pod by setting **replica>1** you must enable the **no_wdelay** mount option. For example:

```
# cat /etc/exports
/mnt/data *(rw,sync,no_wdelay,no_root_squash,insecure,fsid=0)
sh-4.3# exportfs -rv
exporting *:/mnt/data
```

1. Check the registry configuration:

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

storage:
  pvc:
    claim:
```

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

2. Check the **clusteroperator** status:

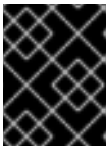
```
$ oc get clusteroperator image-registry
```

2.4.3. 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.
- A provisioned persistent volume (PV) with **ReadWriteMany** access mode, such as **NFS**.



IMPORTANT

vSphere volumes do not support the **ReadWriteMany** access mode. You must use a different storage backend, such as **NFS**, to configure the registry storage.

- Must have "100Gi" capacity.

Procedure

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

2. Verify you do not have a registry pod:

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

NOTE

If the storage type is **emptyDIR**, the replica number can not be greater than **1**. If the storage type is **NFS**, and you want to scale up the registry Pod by setting **replica>1** you must enable the **no_wdelay** mount option. For example:

```
# cat /etc/exports
/mnt/data *(rw,sync,no_wdelay,no_root_squash,insecure,fsid=0)
sh-4.3# exportfs -rv
exporting *:/mnt/data
```

1. Check the registry configuration:

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

storage:
  pvc:
    claim:
```

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

2. Check the **clusteroperator** status:

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

CHAPTER 3. 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.

3.1. INTEGRATED OPENSHIFT CONTAINER PLATFORM REGISTRY

OpenShift Container Platform provides a built in container image registry which 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 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.

3.2. 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.

3.2.1. 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.

3.3. 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.

3.4. AUTHENTICATION ENABLED RED HAT REGISTRY

All container images available through the Red Hat Container 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.

NOTE

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. Because the registry requires authentication for access, the Samples Operator creates the **samples-registry-credentials** secret.

You must place your credentials in two 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.

CHAPTER 4. 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.

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>
```

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

```
$ oc login -u kubeadmin -p <password_from_install_log>
$ podman login -u kubeadmin -p $(oc whoami -t) image-registry.openshift-image-registry.svc:5000
```

You should see a message confirming login, such as:

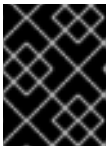
```
Login Succeeded!
```

NOTE

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>**.

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



IMPORTANT

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:

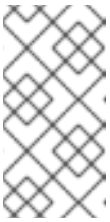
Component	Value
<registry_ip>	**172.30.124.220**
<port>	**5000**
<project>	**openshift**
<image>	**image**
<tag>	omitted (defaults to **latest**)

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
```



NOTE

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
```

4.2. VIEWING THE REGISTRY'S CONTENTS

As an administrator, you can view your registry's contents.

Prerequisites

- Log in as administrator.

Procedure

1. Check the pods under project **openshift-image-registry**:

```
# oc get pods
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
```

4.3. 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
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
```

4.4. 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
# 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 |
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-scraper
rules:
- apiGroups:
  - image.openshift.io
  resources:
  - registry/metrics
  verbs:
  - get
EOF
$ oc create -f -
```

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
...
```

Additional resources

- 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.

CHAPTER 5. 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.

5.1. 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 $(oc whoami) -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 with your route's TLS keys:

```
$ oc create secret tls public-route-tls \
    -n 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
...
```

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