Red Hat Quay 3.7

Upgrade Red Hat Quay

Upgrade Red Hat Quay

Last Updated: 2022-08-11

Red Hat Quay 3.7 Upgrade Red Hat Quay

Upgrade Red Hat Quay

Legal Notice

Abstract

Upgrade Red Hat Quay

Table of Contents

CHAPTER 1. UPGRADE OVERVIEW

The upgrade procedure for Red Hat Quay depends on the type of installation you are using.

The Red Hat Quay Operator provides a simple method to deploy and manage a Red Hat Quay cluster. This is the preferred procedure for deploying Red Hat Quay on OpenShift.

The Red Hat Quay Operator should be upgraded using the Operator Lifecycle Manager (OLM) as described in the section "Upgrading Quay using the Quay Operator".

The procedure for upgrading a proof-of-concept or highly available installation of Red Hat Quay and Clair is documented in the section "Standalone upgrade".

CHAPTER 2. UPGRADING THE QUAY OPERATOR OVERVIEW

The Quay Operator follows a *synchronized versioning* scheme, which means that each version of the Operator is tied to the version of Quay and the components that it manages. There is no field on the **QuayRegistry** custom resource which sets the version of Quay to deploy; the Operator only knows how to deploy a single version of all components. This scheme was chosen to ensure that all components work well together and to reduce the complexity of the Operator needing to know how to manage the lifecycles of many different versions of Quay on Kubernetes.

2.1. OPERATOR LIFECYCLE MANAGER

The Quay Operator should be installed and upgraded using the Operator Lifecycle Manager (OLM). When creating a **Subscription** with the default **approvalStrategy: Automatic**, OLM will automatically upgrade the Quay Operator whenever a new version becomes available.

WARNING

When the Quay Operator is installed via Operator Lifecycle Manager, it may be configured to support automatic or manual upgrades. This option is shown on the **Operator Hub** page for the Quay Operator during installation. It can also be found in the Quay Operator **Subscription** object via the **approvalStrategy** field. Choosing **Automatic** means that your Quay Operator will automatically be upgraded whenever a new Operator version is released. If this is not desirable, then the **Manual** approval strategy should be selected.

2.2. UPGRADING THE QUAY OPERATOR

The standard approach for upgrading installed Operators on OpenShift is documented at Upgrading installed Operators.

In general, Red Hat Quay supports upgrades from a prior (N-1) minor version only. For example, upgrading directly from Red Hat Quay 3.0.5 to the latest version of 3.5 is not supported. Instead, users would have to upgrade as follows:

1. 3.0.5 → 3.1.3
2. 3.1.3 → 3.2.2
3. 3.2.2 → 3.3.4
4. 3.3.4 → 3.4.z
5. 3.4.z → 3.5.z

This is required to ensure that any necessary database migrations are done correctly and in the right order during the upgrade.

In some cases, Red Hat Quay supports direct, single-step upgrades from prior (N-2, N-3) minor versions. This exception to the normal, prior minor version-only, upgrade simplifies the upgrade procedure for customers on older releases. The following upgrade paths are supported:

1. 3.3.z → 3.6.z
2. 3.4.z → 3.6.z
3. 3.4.z → 3.7.z
4. 3.5.z → 3.7.z

For users on standalone deployments of Quay wanting to upgrade to 3.7, see the Standalone upgrade guide.

2.2.1. Upgrading Quay

To update Quay from one minor version to the next, for example, 3.4 → 3.5, you need to change the update channel for the Quay Operator.

For **z** stream upgrades, for example, 3.4.2 → 3.4.3, updates are released in the major-minor channel that the user initially selected during install. The procedure to perform a **z** stream upgrade depends on the **approvalStrategy** as outlined above. If the approval strategy is set to **Automatic**, the Quay Operator will upgrade automatically to the newest **z** stream. This results in automatic, rolling Quay updates to newer **z** streams with little to no downtime. Otherwise, the update must be manually approved before installation can begin.

2.2.2. Notes on upgrading directly from 3.3.z or 3.4.z to 3.6

2.2.2.1. Upgrading with edge routing enabled

- Previously, when running a 3.3.z version of Red Hat Quay with edge routing enabled, users were unable to upgrade to 3.4.z versions of Red Hat Quay. This has been resolved with the release of Red Hat Quay 3.6.
- When upgrading from 3.3.z to 3.6, if **tls.termination** is set to **none** in your Red Hat Quay 3.3.z deployment, it will change to HTTPS with TLS edge termination and use the default cluster wildcard certificate. For example:

```
apiVersion: redhatcop.redhat.io/v1alpha1
kind: QuayEcosystem
metadata:
  name: quay33
spec:
  quay:
    imagePullSecretName: redhat-pull-secret
    enableRepoMirroring: true
    image: quay.io/quay/quay:v3.3.4-2
    ...
    externalAccess:
      hostname: quayv33.apps.devcluster.openshift.com
      tls:
        termination: none
    database:
...
```

2.2.2.2. Upgrading with custom TLS certificate/key pairs without Subject Alternative Names

There is an issue for customers using their own TLS certificate/key pairs without Subject Alternative Names (SANs) when upgrading from Red Hat Quay 3.3.4 to Red Hat Quay 3.6 directly. During the upgrade to Red Hat Quay 3.6, the deployment is blocked, with the error message from the Quay Operator pod logs indicating that the Quay TLS certificate must have SANs.

If possible, you should regenerate your TLS certificates with the correct hostname in the SANs. A possible workaround involves defining an environment variable in the **quay-app**, **quay-upgrade** and **quay-config-editor** pods after upgrade to enable CommonName matching:

```
GODEBUG=x509ignoreCN=0
```

The **GODEBUG=x509ignoreCN=0** flag enables the legacy behavior of treating the CommonName field on X.509 certificates as a host name when no SANs are present. However, this workaround is not recommended, as it will not persist across a redeployment.

2.2.2.3. Configuring Clair v4 when upgrading from 3.3.z or 3.4.z to 3.6 using the Quay Operator

To set up Clair v4 on a new Red Hat Quay deployment on OpenShift, it is highly recommended to use the Quay Operator. By default, the Quay Operator will install or upgrade a Clair deployment along with your Red Hat Quay deployment and configure Clair security scanning automatically.

For instructions on setting up Clair v4 on OpenShift, see Setting Up Clair on a Red Hat Quay OpenShift deployment.

2.2.3. Swift configuration when upgrading from 3.3.z to 3.6

When upgrading from Red Hat Quay 3.3.z to 3.6.z, some users might receive the following error: **Switch auth v3 requires tenant_id (string) in os_options**. As a workaround, you can manually update your **DISTRIBUTED_STORAGE_CONFIG** to add the **os_options** and **tenant_id** parameters:

```
DISTRIBUTED_STORAGE_CONFIG:
  brscale:
  - SwiftStorage
  - auth_url: http://****/v3
    auth_version: "3"
    os_options:
      tenant_id: ****
      project_name: ocp-base
      user_domain_name: Default
    storage_path: /datastorage/registry
    swift_container: ocp-svc-quay-ha
    swift_password: *****
    swift_user: *****
```

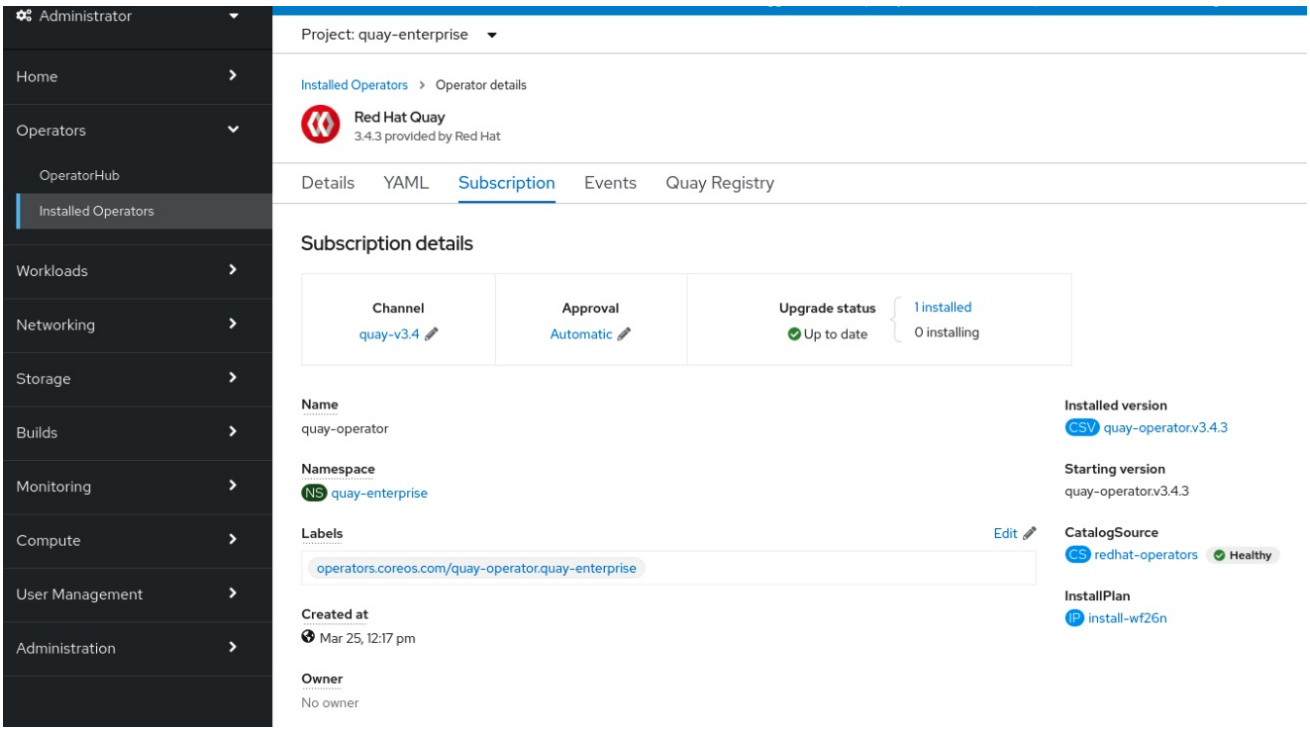
2.2.4. Changing the update channel for an Operator

The subscription of an installed Operator specifies an update channel, which is used to track and receive updates for the Operator. To upgrade the Quay Operator to start tracking and receiving updates from a newer channel, change the update channel in the **Subscription** tab for the installed Quay Operator. For subscriptions with an **Automatic** approval strategy, the upgrade begins automatically and can be monitored on the page that lists the Installed Operators.

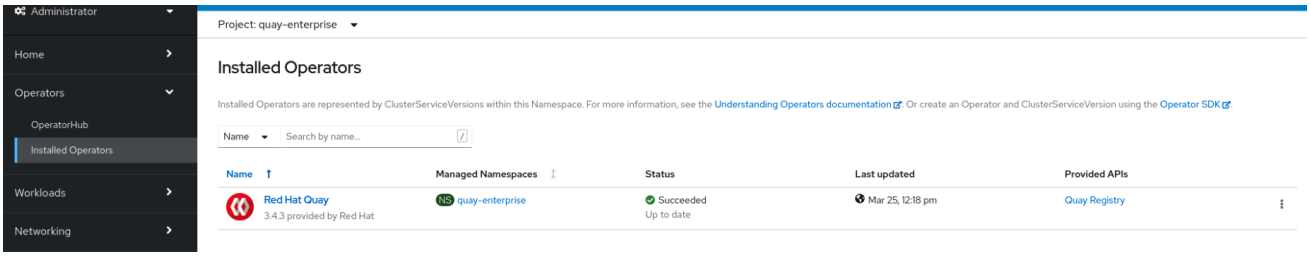
2.2.5. Manually approving a pending Operator upgrade

If an installed Operator has the approval strategy in its subscription set to **Manual**, when new updates are released in its current update channel, the update must be manually approved before installation can begin. If the Quay Operator has a pending upgrade, this status will be displayed in the list of Installed Operators. In the **Subscription** tab for the Quay Operator, you can preview the install plan and review the resources that are listed as available for upgrade. If satisfied, click **Approve** and return to the page that lists Installed Operators to monitor the progress of the upgrade.

The following image shows the **Subscription** tab in the UI, including the update **Channel**, the **Approval** strategy, the **Upgrade status** and the **InstallPlan**:



The list of Installed Operators provides a high-level summary of the current Quay installation:



2.3. UPGRADING A QUAYREGISTRY

When the Quay Operator starts, it immediately looks for any **QuayRegistries** it can find in the namespace(s) it is configured to watch. When it finds one, the following logic is used:

- If **status.currentVersion** is unset, reconcile as normal.
- If **status.currentVersion** equals the Operator version, reconcile as normal.
- If **status.currentVersion** does not equal the Operator version, check if it can be upgraded. If it can, perform upgrade tasks and set the **status.currentVersion** to the Operator's version once complete. If it cannot be upgraded, return an error and leave the **QuayRegistry** and its deployed Kubernetes objects alone.

2.4. ENABLING FEATURES IN QUAY 3.7

2.4.1. Quota management configuration

Quota management is now supported under the **FEATURE_QUOTA_MANAGEMENT** property and is turned off by default. To enable quota management, set the feature flag in your **config.yaml** to **true**:

```
FEATURE_QUOTA_MANAGEMENT: true
```

2.4.2. Using Red Hat Quay to proxy a remote organization configuration

Using Red Hat Quay to proxy a remote organization is now supported under the **FEATURE_PROXY_CACHE** property. To enable proxy cache, set the feature flag in your **confg.yaml** to **true**:

```
FEATURE_PROXY_CACHE: true
```

2.4.3. Red Hat Quay build enhancements

Builds can be run on virtualized platforms. Backwards compatibility to run previous build configurations are also available. To enable virtual builds, set the feature flag in your **config.yaml** to **true**:

```
FEATURE_BUILD_SUPPORT: true
```

2.4.4. Geo-replication using the Red Hat Quay Operator

Deployments of Red Hat Quay with geo-replication is now supported by Operator deployments. To enable geo-replication, set the feature flag in your **config.yaml** to **true**:

```
FEATURE_STORAGE_REPLICATION: true
```

2.5. ENABLING FEATURES IN QUAY 3.6

2.5.1. Console monitoring and alerting

The support for monitoring Quay 3.6 in the OpenShift console requires that the Operator is installed in all namespaces. If you previously installed the Operator in a specific namespace, delete the Operator itself and reinstall it for all namespaces once the upgrade has taken place.

2.5.2. OCI and Helm support

Support for Helm and some OCI artifacts is now enabled by default in Red Hat Quay 3.6. If you want to explicitly enable the feature, for example, if you are upgrading from a version where it is not enabled by default, you need to reconfigure your Quay deployment to enable the use of OCI artifacts using the following properties:

```
FEATURE_GENERAL_OCI_SUPPORT: true
```

2.6. UPGRADING A QUAYECOSYSTEM

Upgrades are supported from previous versions of the Operator which used the **QuayEcosystem** API for a limited set of configurations. To ensure that migrations do not happen unexpectedly, a special label needs to be applied to the **QuayEcosystem** for it to be migrated. A new **QuayRegistry** will be created for the Operator to manage, but the old **QuayEcosystem** will remain until manually deleted to ensure that you can roll back and still access Quay in case anything goes wrong. To migrate an existing **QuayEcosystem** to a new **QuayRegistry**, follow these steps:

1. Add **"quay-operator/migrate": "true"** to the **metadata.labels** of the **QuayEcosystem**.

```
$ oc edit quayecosystem <quayecosystemname>
```

```
metadata:
  labels:
    quay-operator/migrate: "true"
```

2. Wait for a **QuayRegistry** to be created with the same **metadata.name** as your **QuayEcosystem**. The **QuayEcosystem** will be marked with the label **"quay-operator/migration-complete": "true"**.

3. Once the **status.registryEndpoint** of the new **QuayRegistry** is set, access Quay and confirm all data and settings were migrated successfully.

4. When you are confident everything worked correctly, you may delete the **QuayEcosystem** and Kubernetes garbage collection will clean up all old resources.

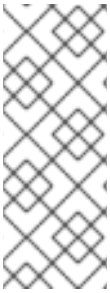
2.6.1. Reverting QuayEcosystem Upgrade

If something goes wrong during the automatic upgrade from **QuayEcosystem** to **QuayRegistry**, follow these steps to revert back to using the **QuayEcosystem**:

1. Delete the **QuayRegistry** using either the UI or **kubectl**:

```
$ kubectl delete -n <namespace> quayregistry <quayecosystem-name>
```

2. If external access was provided using a **Route**, change the **Route** to point back to the original **Service** using the UI or **kubectl**.



NOTE

If your **QuayEcosystem** was managing the Postgres database, the upgrade process will migrate your data to a new Postgres database managed by the upgraded Operator. Your old database will not be changed or removed but Quay will no longer use it once the migration is complete. If there are issues during the data migration, the upgrade process will exit and it is recommended that you continue with your database as an unmanaged component.

2.6.2. Supported QuayEcosystem Configurations for Upgrades

The Quay Operator will report errors in its logs and in **status.conditions** if migrating a **QuayEcosystem** component fails or is unsupported. All unmanaged components should migrate successfully because no Kubernetes resources need to be adopted and all the necessary values are already provided in Quay's **config.yaml**.

Database

Ephemeral database not supported (**volumeSize** field must be set).

Redis

Nothing special needed.

External Access

Only passthrough **Route** access is supported for automatic migration. Manual migration required for other methods.

- **LoadBalancer** without custom hostname: After the **QuayEcosystem** is marked with label **"quay-operator/migration-complete": "true"**, delete the **metadata.ownerReferences** field from existing **Service** *before* deleting the **QuayEcosystem** to prevent Kubernetes from garbage collecting the **Service** and removing the load balancer. A new **Service** will be created with **metadata.name** format **<QuayEcosystem-name>-quay-app**. Edit the **spec.selector** of the existing **Service** to match the **spec.selector** of the new **Service** so traffic to the old load balancer endpoint will now be directed to the new pods. You are now responsible for the old **Service**; the Quay Operator will not manage it.
- **LoadBalancer**/**NodePort**/**Ingress** with custom hostname: A new **Service** of type **LoadBalancer** will be created with **metadata.name** format **<QuayEcosystem-name>-quay-app**. Change your DNS settings to point to the **status.loadBalancer** endpoint provided by the new **Service**.

Clair

Nothing special needed.

Object Storage

QuayEcosystem did not have a managed object storage component, so object storage will always be marked as unmanaged. Local storage is not supported.

Repository Mirroring

Nothing special needed.

CHAPTER 3. STANDALONE UPGRADE

In general, Red Hat Quay supports upgrades from a prior (N-1) minor version only. For example, upgrading directly from Red Hat Quay 3.0.5 to the latest version of 3.5 is not supported. Instead, users would have to upgrade as follows:

1. 3.0.5 → 3.1.3
2. 3.1.3 → 3.2.2
3. 3.2.2 → 3.3.4
4. 3.3.4 → 3.4.z
5. 3.4.z → 3.5.z

This is required to ensure that any necessary database migrations are done correctly and in the right order during the upgrade.

In some cases, Red Hat Quay supports direct, single-step upgrades from prior (N-2, N-3) minor versions. This exception to the normal, prior minor version-only, upgrade simplifies the upgrade procedure for customers on older releases. The following upgrade paths are supported:

1. 3.3.z → 3.6.z
2. 3.4.z → 3.6.z
3. 3.4.z → 3.7.z
4. 3.5.z → 3.7.z

For users wanting to upgrade via the Quay Operator, see Upgrading Quay by upgrading the Quay Operator.

This document describes the steps needed to perform each individual upgrade. Determine your current version and then follow the steps in sequential order, starting with your current version and working up to your desired target version.

- Upgrade to 3.7.z from 3.6.z
- Upgrade to 3.7.z from 3.5.z
- Upgrade to 3.7.z from 3.4.z
- Upgrade to 3.7.z from 3.3.z
- Upgrade to 3.6.z from 3.5.z
- Upgrade to 3.6.z from 3.4.z
- Upgrade to 3.6.z from 3.3.z
- Upgrade to 3.5.z from 3.4.z
- Upgrade to 3.4.z from 3.3.4
- Upgrade to 3.3.4 from 3.2.2

- Upgrade to 3.2.2 from 3.1.3
- Upgrade to 3.1.3 from 3.0.5
- Upgrade to 3.0.5 from 2.9.5

See the Red Hat Quay Release Notes for information on features for individual releases.

The general procedure for a manual upgrade consists of the following steps:

1. Stop the Quay and Clair containers.
2. Backup the database and image storage (optional but recommended).
3. Start Clair using the new version of the image.
4. Wait until Clair is ready to accept connections before starting the new version of Quay.

3.1. ACCESSING IMAGES

Images for Quay 3.4.0 and later are available from registry.redhat.io and registry.access.redhat.com, with authentication set up as described in Red Hat Container Registry Authentication .

Images for Quay 3.3.4 and earlier are available from quay.io, with authentication set up as described in Accessing Red Hat Quay without a CoreOS login .

3.2. UPGRADE TO 3.7.Z FROM 3.6.Z

3.2.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.7.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.3. UPGRADE TO 3.7.Z FROM 3.5.Z

3.3.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.7.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.4. UPGRADE TO 3.7.Z FROM 3.4.Z

3.4.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.7.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.5. UPGRADE TO 3.7.Z FROM 3.3.Z

Upgrading to Red Hat Quay 3.7 from 3.3. is unsupported. Users must first upgrade to 3.6 from 3.3, and then upgrade to 3.7. For more information, see Upgrade to 3.6.z from 3.3.z .

3.6. UPGRADE TO 3.6.Z FROM 3.5.Z

3.6.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.7.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.7. UPGRADE TO 3.6.Z FROM 3.4.Z

+

NOTE

Red Hat Quay 3.6 supports direct, single-step upgrade from 3.4.z. This exception to the normal, prior minor version-only, upgrade simplifies the upgrade procedure for customers on older releases.

Upgrading to Red Hat Quay 3.6 from 3.4.z requires a database migration which does not support downgrading back to a prior version of Red Hat Quay. Please back up your database before performing this migration.

Users will also need to configure a completely new Clair v4 instance to replace the old Clair v2 when upgrading from 3.4.z. For instructions on configuring Clair v4, see Setting up Clair on a non-OpenShift Red Hat Quay deployment.

3.7.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.7.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.8. UPGRADE TO 3.6.Z FROM 3.3.Z

+

NOTE

Red Hat Quay 3.6 supports direct, single-step upgrade from 3.3.z. This exception to the normal, prior minor version-only, upgrade simplifies the upgrade procedure for customers on older releases.

Upgrading to Red Hat Quay 3.6.z from 3.3.z requires a database migration which does not support downgrading back to a prior version of Red Hat Quay. Please back up your database before performing this migration.

Users will also need to configure a completely new Clair v4 instance to replace the old Clair v2 when upgrading from 3.3.z. For instructions on configuring Clair v4, see Setting up Clair on a non-OpenShift Red Hat Quay deployment.

3.8.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.7.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.8.2. Swift configuration when upgrading from 3.3.z to 3.6

When upgrading from Red Hat Quay 3.3.z to 3.6.z, some users might receive the following error: **Switch auth v3 requires tenant_id (string) in os_options**. As a workaround, you can manually update your **DISTRIBUTED_STORAGE_CONFIG** to add the **os_options** and **tenant_id** parameters:

```
DISTRIBUTED_STORAGE_CONFIG:
  brscale:
  - SwiftStorage
  - auth_url: http://****/v3
    auth_version: "3"
    os_options:
      tenant_id: ****
      project_name: ocp-base
      user_domain_name: Default
    storage_path: /datastorage/registry
    swift_container: ocp-svc-quay-ha
    swift_password: *****
    swift_user: *****
```

3.9. UPGRADE TO 3.5.7 FROM 3.4.Z

3.9.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.5.7

- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.10. UPGRADE TO 3.4.6 FROM 3.3.Z

Upgrading to Quay 3.4 requires a database migration which does not support downgrading back to a prior version of Quay. Please back up your database before performing this migration.

3.10.1. Target images

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.4.6
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5:1

3.11. UPGRADE TO 3.3.4 FROM 3.2.Z

3.11.1. Target images

- **Quay:** quay.io/redhat/quay:v3.3.4
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7

3.12. UPGRADE TO 3.2.2 FROM 3.1.Z

Once your cluster is running any Red Hat Quay 3.1.z version, to upgrade your cluster to 3.2.2 you must bring down your entire cluster and make a small change to the configuration before bringing it back up with the 3.2.2 version.

WARNING

Once you set the value of DATABASE_SECRET_KEY in this procedure, do not ever change it. If you do so, then existing robot accounts, API tokens, etc. cannot be used anymore. You would have to create a new robot account and API tokens to use with Quay.

1. Take all hosts in the Red Hat Quay cluster out of service.
2. Generate some random data to use as a database secret key. For example:

```
$ openssl rand -hex 48
2d023adb9c477305348490aa0fd9c
```

3. Add a new DATABASE_SECRET_KEY field to your **config.yaml** file. For example:

```
DATABASE_SECRET_KEY: "2d023adb9c477305348490aa0fd9c"
```

NOTE

For an OpenShift installation, the **config.yaml** file is stored as a secret.

4. Bring up one **Quay** container to complete the migration to 3.2.2.
5. Once the migration is done, make sure the same **config.yaml** is available on all nodes and bring up the new quay 3.2.2 service on those nodes.
6. Start 3.0.z versions of quay-builder and Clair to replace any instances of those containers you want to return to your cluster.

3.12.1. Target images

- **Quay:** quay.io/redhat/quay:v3.2.2
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7

3.13. UPGRADE TO 3.1.3 FROM 3.0.Z

3.13.1. Target images

- **Quay:** quay.io/redhat/quay:v3.1.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.7.3
- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7

3.14. UPGRADE TO 3.0.5 FROM 2.9.5

For the 2.9.5 to 3.0.5 upgrade, you can either do the whole upgrade with Red Hat Quay down (synchronous upgrade) or only bring down Red Hat Quay for a few minutes and have the bulk of the upgrade continue with Red Hat Quay running (background upgrade).

A background upgrade could take longer to run the upgrade depending on how many tags need to be processed. However, there is less total downtime. The downside of a background upgrade is that you will not have access to the latest features until the upgrade completes. The cluster runs from the Quay v3 container in v2 compatibility mode until the upgrade is complete.

3.14.1. Overview of upgrade

Follow the procedure below if you are starting with a Red Hat Quay 2.y.z cluster. Before upgrading to the latest Red Hat Quay 3.x version, you must first migrate that cluster to 3.0.5, as described here. Once your cluster is running 3.0.5, you can then upgrade to the latest 3.x version by sequentially upgrading to each minor version in turn. For example:

1. 3.0.5 → 3.1.3
2. 3.1.3 → 3.2.2
3. 3.2.2 → 3.3.4
4. 3.3.4 → 3.4.z

Before beginning your Red Hat Quay 2.y.z to 3.0 upgrade, please note the following:

- **Synchronous upgrade**: For a synchronous upgrade, expect less than one hour of total downtime for small installations. Consider a small installation to contain a few thousand container image tags or fewer. For that size installation, you could probably get by with just a couple hours of scheduled downtime. The entire Red Hat Quay service is down for the duration, so if you were to try a synchronous upgrade on a registry with millions of tags, you could potentially be down for several days.
- **Background upgrade**: For a background upgrade (also called a compatibility mode upgrade), after a short shutdown your Red Hat Quay cluster upgrade runs in the background. For large Red Hat Quay registries, this could take weeks to complete, but the cluster continues to operate in v2 mode for the duration of the upgrade. As a point of reference, one Red Hat Quay v3 upgrade took four days to process approximately 30 million tags across six machines.
- **Full features on completion**: Before you have access to features associated with Docker version 2, schema 2 changes (such as support for containers of different architectures), the entire migration must complete. Other v3 features are immediately available when you switch over.
- **Upgrade complete**: When the upgrade is complete, you need to set **V3_UPGRADE_MODE: complete** in the Red Hat Quay **config.yaml** file for the new features to be available. All new Red Hat Quay v3 installations automatically have that set.

3.14.2. Prerequisites

To assure the best results, we recommend the following prerequisites:

- Back up your Red Hat Quay database before starting the upgrade (doing regular backups is a general best practice). A good time to do this is right after you have taken down the Red Hat Quay cluster to do the upgrade.
- Back up your storage (also a general best practice).
- Upgrade your current Red Hat Quay 2.y.z setup to the latest 2.9.z version (currently 2.9.5) before starting the v3 upgrade. To do that:
    - While the Red Hat Quay cluster is still running, take one node and change the **Quay** container on that system to a **Quay** container that is running the latest 2.9.z version.
    - Wait for all the database migrations to run, bringing the database up to the latest 2.9.z version. This should only take a few minutes to a half an hour.

- Once that is done, replace the **Quay** container on all the existing nodes with the same latest 2.9.z version. With the entire Red Hat Quay cluster on the new version, you can proceed to the v3 upgrade.

3.14.3. Choosing upgrade type

Choose between a synchronous upgrade (complete the upgrade in downtime) and a background upgrade (complete the upgrade while Red Hat Quay is still running). Both of these major-release upgrades require that the Red Hat Quay cluster be down for at least a short period of time.

Regardless of which upgrade type you choose, during the time that the Red Hat Quay cluster is down, if you are using builder and Clair images, you need to also upgrade to those new images:

- **Builder**: quay.io/redhat/quay-builder:v3.0.5
- **Clair**: quay.io/redhat/clair-jwt:v3.0.5

Both of those images are available from the registry.redhat.io/quay repository.

3.14.4. Running a synchronous upgrade

To run a synchronous upgrade, where your whole cluster is down for the entire upgrade, do the following:

1. Take down your entire Red Hat Quay cluster, including any quay-builder and Clair containers.
2. Add the following setting to the **config.yaml** file on all nodes:
   V3_UPGRADE_MODE: complete
3. Pull and start up the v3 container on a single node and wait for however long it takes to do the upgrade (it will take a few minutes). Use the following container or later:
   - **Quay**: quay.io/redhat/quay:v3.0.5
     Note that the **Quay** container comes up on ports 8080 and 8443 for Red Hat Quay 3, instead of 80 and 443, as they did for Red Hat Quay 2. Therefore, we recommend remapping 8080 and 8443 into 80 and 443, respectively, as shown in this example:

```
# docker run --restart=always -p 80:8080 -p 443:8443 \
  --sysctl net.core.somaxconn=4096 \
  --privileged=true \
  -v /mnt/quay/config:/conf/stack:Z \
  -v /mnt/quay/storage:/datastorage:Z \
  -d quay.io/redhat/quay:v3.0.5
```

4. After the upgrade completes, bring the Red Hat Quay 3 container up on all other nodes.
5. Start 3.0.z versions of quay-builder and Clair to replace any instances of those containers you want to return to your cluster.
6. Verify that Red Hat Quay is working, including pushes and pulls of containers compatible with Docker version 2, schema 2. This can include windows container images and images of different computer architectures (arm, ppc, etc.).

3.14.5. Running a background upgrade

To run a background upgrade, you need only bring down your cluster for a short period of time on two

occasions. When you bring the cluster back up after the first downtime, the quay v3 container runs in v2 compatibility mode as it backfills the database. This background process can take hours or even days to complete. Background upgrades are recommended for large installations where downtime of more than a few hours would be a problem.

For this type of upgrade, you put Red Hat Quay into a compatibility mode, where you have a **Quay** 3 container running, but it is running on the old data model while the upgrade completes. Here's what you do:

1. Pull the Red Hat Quay 3 container to all the nodes. Use the following container or later: quay.io/redhat/quay:v3.0.5

2. Take down your entire Red Hat Quay cluster, including any quay-builder and Clair containers.

3. Edit the **config.yaml** file on each node and set the upgrade mode to background as follows: V3_UPGRADE_MODE: background

4. Bring the Red Hat Quay 3 container up on a single node and wait for the migrations to complete (should take a few minutes maximum). Here is an example of that command:
   Note that the **Quay** container comes up on ports 8080 and 8443 for Red Hat Quay 3, instead of 80 and 443, as they did for Red Hat Quay 2. Therefore, we recommend remapping 8080 and 8443 into 80 and 443, respectively, as shown in this example:

   ```
   # docker run --restart=always -p 80:8080 -p 443:8443 \
     --sysctl net.core.somaxconn=4096 \
     --privileged=true \
     -v /mnt/quay/config:/conf/stack:Z \
     -v /mnt/quay/storage:/datastorage:Z \
     -d quay.io/redhat/quay:v3.0.5
   ```

5. Bring the Red Hat Quay 3 container up on all the other nodes.

6. Monitor the /**upgradeprogress** API endpoint until it reports done enough to move to the next step (the status reaches 99%). For example, view **https://myquay.example.com/upgradeprogress** or use some other tool to query the API.

7. Once the background process is far enough along you have to schedule another maintenance window.

8. During your scheduled maintenance, take the entire Red Hat Quay cluster down.

9. Edit the **config.yaml** file on each node and set the upgrade mode to **complete** as follows:

   ```
   V3_UPGRADE_MODE: complete
   ```

10. Bring Red Hat Quay back up on one node to have it do a final check.

11. Once the final check is done, bring Red Hat Quay v3 back up on all the other nodes.

12. Start 3.0.z versions of quay-builder and Clair to replace any instances of those containers you want to return to your cluster.

13. Verify Quay is working, including pushes and pulls of containers compatible with Docker version 2, schema 2. This can include windows container images and images of different computer architectures (arm, ppc, etc.).

3.14.6. Target images

- **Quay:** quay.io/redhat/quay:v3.0.5
- **Clair:** quay.io/redhat/clair-jwt:v3.0.5
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7
- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Builder:** quay.io/redhat/quay-builder:v3.0.5

CHAPTER 4. UPGRADE QUAY BRIDGE OPERATOR

To upgrade the Quay Bridge Operator (QBO), change the Channel Subscription update channel in the Subscription tab to the desired channel.

When upgrading QBO from version 3.5 to 3.7, a number of extra steps are required:

1. You need to create a new **QuayIntegration** custom resource. This can be completed in the Web Console or from the command line.

   upgrade-quay-integration.yaml

   ```
   - apiVersion: quay.redhat.com/v1
     kind: QuayIntegration
     metadata:
       name: example-quayintegration-new
     spec:
       clusterID: openshift 1
       credentialsSecret:
         name: quay-integration
         namespace: openshift-operators
       insecureRegistry: false
       quayHostname: https://registry-quay-quay35.router-default.apps.cluster.openshift.com
   ```

   1 Make sure that the **clusterID** matches the value for the existing **QuayIntegration** resource.

2. Create the new **QuayIntegration** custom resource:

   ```
   $ oc create -f upgrade-quay-integration.yaml
   ```

3. Delete the old **QuayIntegration** custom resource.

4. Delete the old **mutatingwebhookconfigurations**:

   ```
   $ oc delete mutatingwebhookconfigurations.admissionregistration.k8s.io quay-bridge-operator
   ```