Red Hat Quay 2.9

Deploy Red Hat Quay - High Availability

Deploy Red Hat Quay HA

Last Updated: 2019-04-20

Red Hat Quay 2.9 Deploy Red Hat Quay - High Availability

Deploy Red Hat Quay HA

Legal Notice

Abstract

Deploy Red Hat Quay in a HA environment

Table of Contents

PREFACE

Red Hat Quay is an enterprise-quality container registry. Use Quay to build and store containers, then deploy them to the servers across your enterprise.

This procedure describes how to deploy a high availability, enterprise-quality Red Hat Quay setup.

CHAPTER 1. OVERVIEW

Features of Quay include:

- High availability
- Geo-replication
- Continuous integration
- Security scanning
- Custom log rotation
- 24/7 support

Quay provides support for multiple:

- Authentication and access methods
- Storage backends
- SSL certificates
- Application registries

CHAPTER 2. ARCHITECTURE

Quay is made up of three core components for a basic setup. In highly available setups, an additional object storage component is needed. The three core components are:

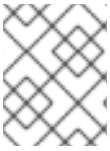
- **Database (MySQL or PostgreSQL)**: Used by Quay as its primary metadata storage (not for image storage).
- **Redis (key, value store)**: Used for providing real time events and during Quay setup and installation.
- **Quay (container registry)**: Runs Quay as a service, consisting of several components in the pod.

For the high availability installation, you need to use one of the following types of storage:

- **Public cloud storage**: In public cloud environments, you should use the cloud provider's object storage, such as Amazon S3 (for AWS) or Google Cloud Storage (for Google Cloud).
- **Private cloud storage**: In private clouds, an S3 or Swift compliant Object Store is needed, such as Ceph RADOS, or OpenStack Swift.

Local storage is supported for the Red Hat Quay test-only installation, but not for high-availability.

CHAPTER 3. INSTALLING RED HAT QUAY (HIGH AVAILABILITY)



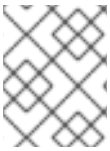
NOTE

This procedure presents guidance on how to set up a highly available, production-quality deployment of Red Hat Quay.

3.1. PREREQUISITES

Here are are a few things you need to know before you begin the Red Hat Quay high availability deployment:

- Either Postgres or MySQL can be used to provide the database service. Postgres was chosen here as the database because it includes the features needed to support Clair security scanning.



NOTE

You can substitute your own enterprise-quality database if you choose. The Postgres database illustrated here is not, itself, configured for high availability.

- Ceph Object Gateway (also called RADOS Gateway) provides the object storage needed by Red Hat Quay. If you want your Red Hat Quay setup to do geo-replication, Ceph Object Gateway or other supported object storage is required. For cloud installations, you can use any of the following cloud object storage:
  - Amazon S3 (see S3 IAM Bucket Policy for details on configuring an S3 bucket policy for Quay)
  - Azure Blob Storage
  - Google Cloud Storage
  - Ceph Object Gateway
  - OpenStack Swift
  - CloudFront + S3
- The haproxy server is used in this example, although you can use any proxy service that works for your environment.
- Number of systems: This procedure uses seven systems (physical or virtual) that are assigned with the following tasks:
  - **db01: Load balancer and database**: Runs the haproxy load balancer and a Postgres database. Note that thes components are not themselves highly available, but are used to indicate how you might set up your own load balancer or production database.
  - **quay01, quay02, quay03: Quay and Redis**: Three (or more) systems are assigned to run the Quay and Redis services.
  - **ceph01, ceph02, ceph03, ceph04, ceph05: Ceph**: Three (or more) systems provide the

Ceph service, for storage. If you are deploying to a cloud, you can use the cloud storage features described earlier. This procedure employs an additional system for Ansible (ceph05) and one for a Ceph Object Gateway (ceph04).

Each system should have the following attributes:

- **Red Hat Enterprise Linux (RHEL)**: Obtain the latest Red Hat Enterprise Linux server media from the Downloads page and follow instructions from the Red Hat Enterprise Linux 7 Installation Guide to install RHEL on each system.
  - **Valid Red Hat Subscription**: Obtain Red Hat Enterprise Linux server subscriptions and apply one to each system.
  - **CPUs**: Two or more virtual CPUs
  - **RAM**: 4GB for each A and B system; 8GB for each C system
  - **Disk space**: About 20GB of disk space for each A and B system (10GB for the operating system and 10GB for docker storage). At least 30GB of disk space for C systems (or more depending on required container storage).

3.2. SET UP LOAD BALANCER AND DATABASE

On the first two systems (q01 and q02), install the haproxy load balancer and postgresql database. Haproxy will be configured as the access point and load balancer for the following services running on other systems:

- Quay (ports 80 and 443 on B systems)
- Redis (port 6379 on B systems)
- RADOS (port 7480 on C systems)

Because the services on the two systems run as containers, you also need the docker service running. Here's how to set up the A systems:

1. **Install and start docker service**: Install, start, and enable the docker service.
2. **Open ports for haproxy service**: Open all haproxy ports in SELinux and selected haproxy ports in the firewall:

```
# setsebool -P haproxy_connect_any=on
# firewall-cmd --permanent --zone=public --add-port=6379/tcp --add-
port=7480/tcp
success
# firewall-cmd --reload
success
```

3. **Set up haproxy service**: Configure the **`/etc/haproxy/haproxy.cfg`** to point to the systems and ports providing the Quay, Redis, and Ceph RADOS services. Here are examples of defaults and added frontend and backend settings:

```
#---------------------------------------------------------------------
------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
```

```
#---------------------------------------------------------------------
------
defaults
    mode                    tcp
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

#---------------------------------------------------------------------
------
# main frontend which proxys to the backends
#---------------------------------------------------------------------
------

frontend  fe_http *:80
    default_backend             be_http
frontend  fe_https *:443
    default_backend             be_https
frontend fe_redis *:6379
   default_backend              be_redis
frontend  fe_rdgw *:7480
    default_backend             be_rdgw
backend be_http
    balance     roundrobin
    server quay01 quay01:80 check
    server quay02 quay02:80 check
    server quay03 quay03:80 check
backend be_https
    balance     roundrobin
    server quay01 quay01:443 check
    server quay02 quay02:443 check
    server quay03 quay03:443 check
backend be_rdgw
    balance     roundrobin
    server ceph01 ceph01:7480 check
    server ceph02 ceph02:7480 check
    server ceph03 ceph03:7480 check
backend be_redis
server quay01 quay01:6380 check inter 1s
server quay02 quay02:6380 check inter 1s
server quay03 quay03:6380 check inter 1s
```

Once the new haproxy.cfg file is in place, restart the haproxy service.

```
# systemctl restart haproxy
```

4. **Install** / **Deploy a Database**: Install, enable and start the PostgreSQL database container. The following commands will:

   - Start the PostgreSQL database with the user, password and database all set. Data from the container will be stored on the host system in the **`/var/lib/pgsql/data`** directory.
   - List available extensions.
   - Create the pg_trgm extension.
   - Confirm the extension is installed

```
$ mkdir -p /var/lib/pgsql/data
$ chmod 777 /var/lib/pgsql/data
$ sudo docker run -d --name postgresql_database \
    -v /var/lib/pgsql/data:/var/lib/pgsql/data:Z  \
    -e POSTGRESQL_USER=quayuser -e POSTGRESQL_PASSWORD=quaypass \
    -e POSTGRESQL_DATABASE=quaydb -p 5432:5432 \
    rhscl/postgresql-96-rhel7

$ sudo docker exec -it postgresql_database /bin/bash -c 'echo
"SELECT * FROM pg_available_extensions" | /opt/rh/rh-
postgresql96/root/usr/bin/psql'
   name    | default_version | installed_version |
comment
-----------+-----------------+-------------------+-----------
---------------------------
 adminpack | 1.0             |                   |
administrative functions for PostgreSQL
...


$ sudo docker exec -it postgresql_database /bin/bash -c 'echo
"CREATE EXTENSION pg_trgm" | /opt/rh/rh-
postgresql96/root/usr/bin/psql'
CREATE EXTENSION

$ sudo docker exec -it postgresql_database /bin/bash -c 'echo
"SELECT * FROM pg_extension" | /opt/rh/rh-
postgresql96/root/usr/bin/psql'
 extname | extowner | extnamespace | extrelocatable | extversion
| extconfig | extcondition
---------+----------+--------------+----------------+---------
---+-----------+--------------
 plpgsql |       10 |           11 | f              | 1.0
|           |
 pg_trgm |       10 |         2200 | t              | 1.3
|           |
(2 rows)

$ sudo docker exec -it postgresql_database /bin/bash -c 'echo
"ALTER USER quayuser WITH SUPERUSER;" | /opt/rh/rh-
postgresql96/root/usr/bin/psql'
ALTER ROLE
```

-

5. **Open the firewall**: If you have a firewalld service active on your system, run the following commands to make the PostgreSQL port available through the firewall:

```
# firewall-cmd --permanent --zone=trusted --add-port=5432/tcp
success
# firewall-cmd --reload
success
```

6. **Test PostgreSQL Connectivity**: Use the **psql** command to test connectivity to the PostgreSQL database. Try this on a remote system as well, to make sure you can access the service remotely:

```
# yum install postgresql -y

# psql -h localhost quaydb quayuser
Password for user test:
psql (9.2.23, server 9.6.5)
WARNING: psql version 9.2, server version 9.6.
         Some psql features might not work.
Type "help" for help.

test=> \q
```

3.3. SET UP CEPH

For this Red Hat Quay configuration, we create a three-node Ceph cluster, with several other supporting nodes, as follows:

- ceph01, ceph02, and ceph03 - Ceph Monitor, Ceph Manager and Ceph OSD nodes
- ceph04 - Ceph RGW node
- ceph05 - Ceph Ansible administration node

For details on installing Ceph nodes, see Installing Red Hat Ceph Storage on Red Hat Enterprise Linux.

Once you have set up the Ceph storage cluster, create a Ceph Object Gateway (also referred to as a RADOS gateway). See Installing the Ceph Object Gateway for details.

3.3.1. Install each Ceph node

On ceph01, ceph02, ceph03, ceph04, and ceph05, do the following:

1. Review prerequisites for setting up Ceph nodes in Requirements for Installing Red Hat Ceph Storage. In particular:
   - Decide if you want to use RAID controllers on OSD nodes.
   - Decide if you want a separate cluster network for your Ceph Network Configuration.
2. Prepare OSD storage (ceph01, ceph02, and ceph03 only). Set up the OSD storage on the three OSD nodes (ceph01, ceph02, and ceph03). See OSD Ansible Settings in Table 3.2 for details on supported storage types that you will enter into your Ansible configuration later. For this

example, a single, unformatted block device (**/dev/sdb**), that is separate from the operating system, is configured on each of the OSD nodes. If you are installing on metal, you might want to add an extra hard drive to the machine for this purpose.

3. Install Red Hat Enterprise Linux Server edition, as described in the RHEL 7 Installation Guide.

4. Register and subscribe each Ceph node as described in the Registering Red Hat Ceph Storage Nodes. Here is how to subscribe to the necessary repos:

```
# subscription-manager repos --disable=*
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-extras-rpms
# subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-
rpms
# subscription-manager repos --enable=rhel-7-server-rhceph-3-osd-
rpms
# subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-
rpms
```

5. Create an ansible user with root privilege on each node. Choose any name you like. For example:

```
# USER_NAME=ansibleadmin
# useradd $USER_NAME -c "Ansible administrator"
# passwd $USER_NAME
New password: *********
Retype new password: *********
# cat << EOF >/etc/sudoers.d/admin
admin ALL = (root) NOPASSWD:ALL
EOF
# chmod 0440 /etc/sudoers.d/$USER_NAME
```

3.3.2. Configure the Ceph Ansible node (ceph05)

Log into the Ceph Ansible node (ceph05) and configure it as follows. You will need the ceph01, ceph02, and ceph03 nodes to be running to complete these steps.

1. In the Ansible user's home directory create a directory to store temporary values created from the ceph-ansible playbook

```
# USER_NAME=ansibleadmin
# sudo su - $USER_NAME
[ansibleadmin@ceph05 ~]$ mkdir ~/ceph-ansible-keys
```

2. Enable password-less ssh for the ansible user. Run ssh-keygen on ceph05 (leave passphrase empty), then run and repeat ssh-copy-id to copy the public key to the Ansible user on ceph01, ceph02, and ceph03 systems:

```
# USER_NAME=ansibleadmin
# sudo su - $USER_NAME
[ansibleadmin@ceph05 ~]$ ssh-keygen
[ansibleadmin@ceph05 ~]$ ssh-copy-id $USER_NAME@ceph01
[ansibleadmin@ceph05 ~]$ ssh-copy-id $USER_NAME@ceph02
[ansibleadmin@ceph05 ~]$ ssh-copy-id $USER_NAME@ceph03
[ansibleadmin@ceph05 ~]$ exit
```

```
#
```

3. Install the ceph-ansible package:

```
# yum install ceph-ansible
```

4. Create a symbolic between these two directories:

```
# ln -s /usr/share/ceph-ansible/group_vars \
    /etc/ansible/group_vars
```

5. Create copies of Ceph sample yml files to modify:

```
# cd /usr/share/ceph-ansible
# cp group_vars/all.yml.sample group_vars/all.yml
# cp group_vars/osds.yml.sample group_vars/osds.yml
# cp site.yml.sample site.yml
```

6. Edit the copied group_vars/all.yml file. See General Ansible Settings in Table 3.1 for details. For example:

```
ceph_origin: repository
ceph_repository: rhcs
ceph_repository_type: cdn
ceph_rhcs_version: 3
monitor_interface: eth0
public_network: 192.168.122.0/24
```

Note that your network device and address range may differ.

7. Edit the copied **`group_vars/osds.yml`** file. See the OSD Ansible Settings in Table 3.2 for details. In this example, the second disk device (**`/dev/sdb`**) on each OSD node is used for both data and journal storage:

```
osd_scenario: collocated
devices:
  - /dev/sdb
dmcrypt: true
osd_auto_discovery: false
```

8. Edit the **`/etc/ansible/hosts`** inventory file to identify the Ceph nodes as Ceph monitor, OSD and manager nodes. In this example, the storage devices are identified on each node as well:

```
[mons]
ceph01
ceph02
ceph03

[osds]
ceph01 devices="[ '/dev/sdb' ]"
ceph02 devices="[ '/dev/sdb' ]"
ceph03 devices="[ '/dev/sdb' ]"

[mgrs]
```

```
ceph01 devices="[ '/dev/sdb' ]"
ceph02 devices="[ '/dev/sdb' ]"
ceph03 devices="[ '/dev/sdb' ]"
```

9. Add this line to the **/etc/ansible/ansible.cfg** file, to save the output from each Ansible playbook run into your Ansible user's home directory:

```
retry_files_save_path = ~/
```

10. Check that Ansible can reach all the Ceph nodes you configured as your Ansible user:

```
# USER_NAME=ansibleadmin
# sudo su - $USER_NAME
[ansibleadmin@ceph05 ~]$ ansible all -m ping
ceph01 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ceph02 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ceph03 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ansibleadmin@ceph05 ~]$
```

11. Run the ceph-ansible playbook (as your Ansible user):

```
[ansibleadmin@ceph05 ~]$ cd /usr/share/ceph-ansible/
[ansibleadmin@ceph05 ~]$ ansible-playbook site.yml
```

At this point, the Ansible playbook will check your Ceph nodes and configure them for the services you requested. If anything fails, make needed corrections and rerun the command.

12. Log into one of the three Ceph nodes (ceph01, ceph02, or ceph03) and check the health of the Ceph cluster:

```
# ceph health
HEALTH_OK
```

13. On the same node, verify that monitoring is working using rados:

```
# ceph osd pool create test 8
# echo 'Hello World!' > hello-world.txt
# rados --pool test put hello-world hello-world.txt
# rados --pool test get hello-world fetch.txt
# cat fetch.txt
Hello World!
```

3.3.3. Install the Ceph Object Gateway

On the Ansible system (ceph05), configure a Ceph Object Gateway to your Ceph Storage cluster (which will ultimately run on ceph04). See Installing the Ceph Object Gateway for details.

3.4. SET UP QUAY AND REDIS

With Red Hat Enterprise Linux server installed on each of the three Quay systems (quay01, quay02, and quay03), install and start the Red Hat Quay and Redis services.

IMPORTANT

When you go to configure Red Hat Quay, only do so on one of the three quay0* systems. Once that is done, the procedure will have you copy that configuration to the other two systems running the Quay service.

1. **Setup Docker**: Install, enable, and start the docker service as shown here (see Getting Docker in RHEL 7 for details):

2. **Install** / **Deploy Redis**: Run Redis as a container on each of the three quay0* systems:

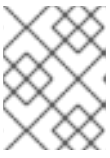
```
# mkdir -p /mnt/hostredis
# chmod 777 /mnt/hostredis
# docker run -d --restart=always -p 6379:6379 \
    -v /mnt/hostredis:/var/lib/redis/data:Z \
    registry.access.redhat.com/rhscl/redis-32-rhel7
```

3. **Check redis connectivity**: You can use the **`telnet`** command to test connectivity to the redis service. Type MONITOR (to begin monitoring the service) and QUIT to exit:

```
# yum install telnet -y
# telnet 192.168.122.99 6379
Trying 192.168.122.99...
Connected to 192.168.122.99.
Escape character is '^]'.
MONITOR
+OK
+1525703165.754099 [0 172.17.0.1:43848] "PING"
QUIT
+OK
Connection closed by foreign host.
```

4. **Add Quay authentication**: Set up authentication to Quay.io, so you can pull the Quay container, as described in Accessing Red Hat Quay without a CoreOS login

5. **Install** / **Deploy Quay**: Start and set up Red Hat Quay on quay01, then start that same service on quay02 and quay03 (using the shared configuration file).



NOTE

Add **`-e DEBUGLOG=true`** to the **`docker run`** command line for the quay container to enable debug level logging.

On each of the three quay0* systems, run Red Hat Quay as a container, as follows:

```
# mkdir -p /mnt/quay/config /mnt/quay/storage
# firewall-cmd --permanent --zone=trusted --add-port=80/tcp
# firewall-cmd --permanent --zone=trusted --add-port=443/tcp
# firewall-cmd --reload

# docker run --restart=always -p 443:443 -p 80:80 \
   --privileged=true \
   -v /mnt/quay/config:/conf/stack \
   -v /mnt/quay/storage:/datastorage \
   -d quay.io/coreos/quay:v2.9.5
```

Wait for the Quay service to come up, then proceed to Completing the Guided Setup.

NOTE

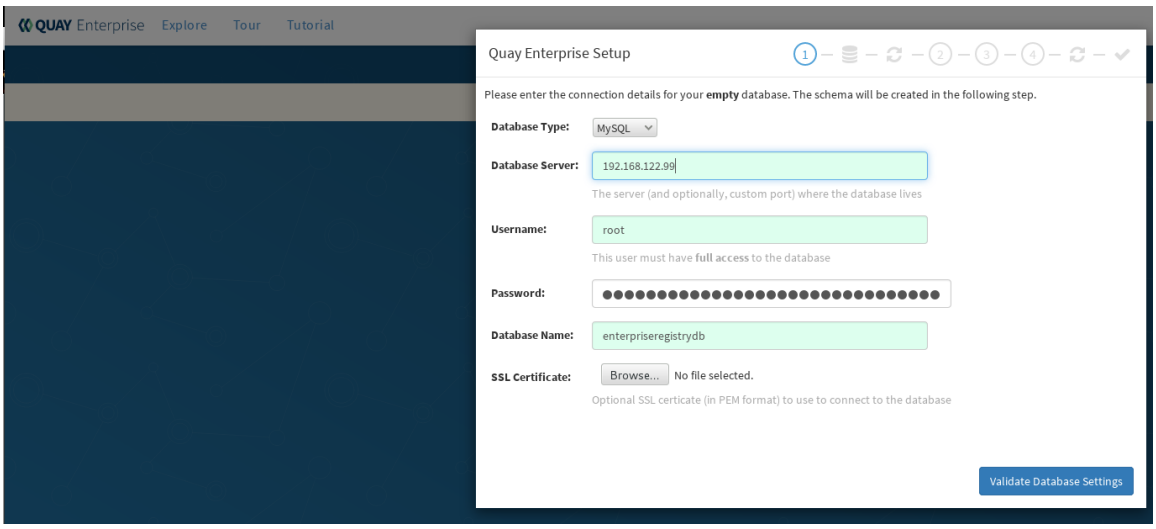
The quay container startup can take several minutes. Type, **`docker ps`** to see the container id and **`docker logs -f <containerid>`** if you want to watch the progress. It's getting near completion when you see the container open the /etc/hosts file. When attempting to access the Guided Setup you might receive a "502 Bad Gateway" nginx message. If you do, wait a while longer and try again.

CHAPTER 4. COMPLETING THE GUIDED SETUP

Open a browser to the setup page on the system where you just started quay (for example http://hostname/setup) and complete the following steps:

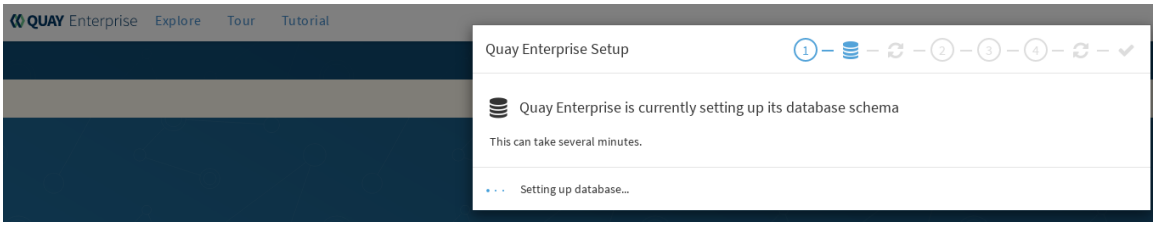
1. **Identify the database**: Add the following information about the type and location of the database to be used by Quay:

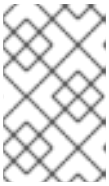
   - **Database Type**: Choose MySQL or PostgreSQL. (MySQL is used in the basic example; PostgreSQL is used with the high availability example.)
   - **Database Server**: Identify the IP address or hostname of the database, along with the port number if it is different from 3306.
   - **Username**: Identify a user with full access to the database.
   - **Password**: Enter the password you assigned to the selected user.
   - **Database Name**: Enter the database name you assigned when you started the database server.
   - **SSL Certificate**: For production environments, you should provide an SSL certificate to connect to the database.
     Figure 1 shows an example of the screen for identifying the database used by Red Hat Quay.



Select "Validate Database Settings", and proceed to the next section.

Figure 2 shows an example of the Red Hat Quay Setup screen as the database schema is set up.

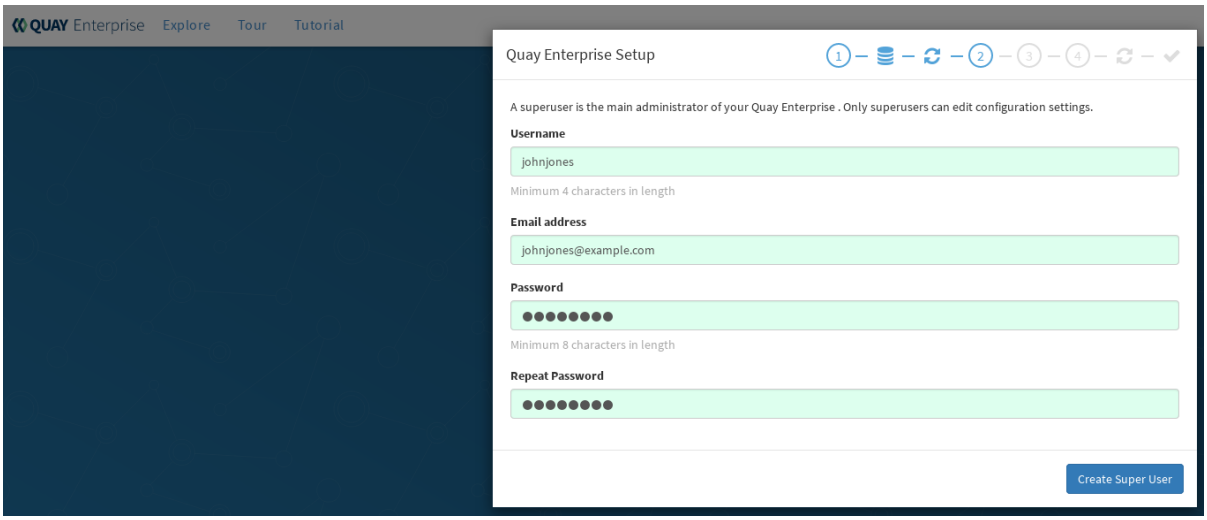




NOTE

At this point a restart of the Quay container should happen. If the container does not restart, the docker restart policy may not be working properly, and a manual restart of the container may be required.

2. **Create Quay superuser**: You need to set up an account with superuser privileges to Quay, to use for editing Quay configuration settings. That information includes a Username, Email address, and Password (entered twice).
   Figure 3 shows an example of the Red Hat Quay Setup screen for setting up a Quay superuser account:



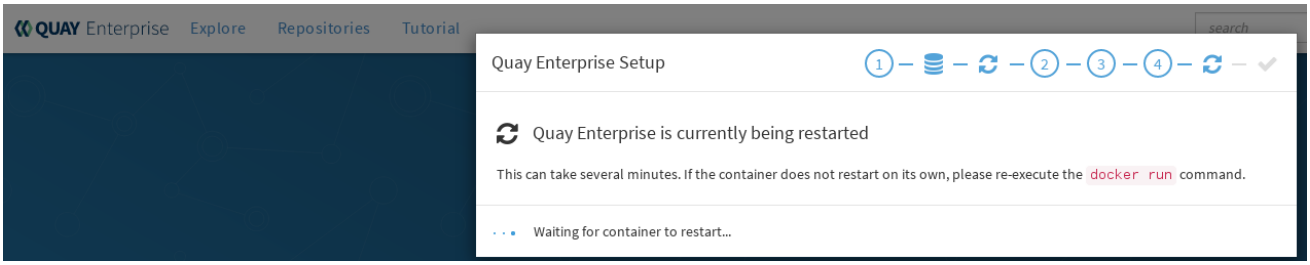
   Select "Create Super User", and proceed to the next section.

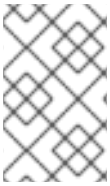
3. **Identify the Redis hostname and add other desired settings**: Other setting you can add to complete the setup are as follows. More settings for high availability Quay deployment that for the basic deployment:

   - For the basic, test configuration, identifying the Redis Hostname should be all you need to do.
   - For the high availability configuration, more settings are needed (as noted below) to allow for shared storage, secure communications between systems, and other features.
     Here are the settings you need to consider:
   - **Custom SSL Certificates**: Upload custom or self-signed SSL certificates for use by Quay. See Using SSL to protect connections to Red Hat Quay for details. Recommended for high availability.
   - **Basic Configuration**: Upload a company logo to rebrand your Quay registry.
   - **Server Configuration**: Hostname or IP address to reach the Quay service, along with TLS indication (recommended for production installations). Recommended for high availability.
   - **Data Consistency Settings**: Select to relax logging consistency guarantees to improve performance and availability.
   - **Time Machine**: Allow older image tags to remain in the repository for set periods of time and allow users to select their own tag expiration times.

- **redis**: Identify the hostname or IP address (and optional password) to connect to the redis service used by Quay.
- **Registry Storage**: Identify the location of storage. A variety of cloud and local storage options are available. Remote storage is required for high availability.
- **Action Log Rotation and Archiving**: Select to enable log rotation, which moves logs older than 30 days into storage, then indicate storage area.
- **Security Scanner**: Enable security scanning by selecting a security scanner endpoint and authentication key. To setup Clair to do image scanning, refer to Clair Setup and Configuring Clair. Recommended for high availability.
- **Application Registry**: Enable an additional application registry that includes things like Kubernetes manifests or Helm charts (see the App Registry specification).
- **BitTorrent-based download**: Allow all registry images to be downloaded using BitTorrent protocol (using quayctl tool).
- **rkt Conversion**: Allow `rkt fetch` to be used to fetch images from Quay registry. Public and private GPG2 keys are needed (see Generating signing keys for ACI conversion for details.
- **E-mail**: Enable e-mail to use for notifications and user password resets.
- **Internal Authentication**: Change default authentication for the registry from Local Database to LDAP, Keystone (OpenStack), JWT Custom Authentication, or External Application Token.
- **External Authorization (OAuth)**: Enable to allow GitHub or GitHub Enterprise to authenticate to the registry.
- **Google Authentication**: Enable to allow Google to authenticate to the registry.
- **Access settings**: Basic username/password authentication is enabled by default. Other authentication types that can be enabled include: external application tokens (user-generated tokens used with docker or rkt commands), anonymous access (enable for public access to anyone who can get to the registry), user creation (let users create their own accounts), encrypted client password (require command-line user access to include encrypted passwords), and prefix username autocompletion (disable to require exact username matches on autocompletion).
- **Dockerfile Build Support**: Enable to allow users to submit Dockerfiles to be built and pushed to Quay.
  Select "Save Configuration Changes", then "Save Configuration.

4. **Restart Quay**: When prompted, select "Restart Container" to restart Quay. Figure 4 shows that screen that appears as you wait for Quay to restart.





NOTE

At this point a restart of the Quay container should happen. If the container does not restart, the docker restart policy may not be working properly, and a manual restart of the container may be required.

CHAPTER 5. ADD OTHER RED HAT QUAY NODES (HIGH AVAILABILITY ONLY)

If you are doing a high availability setup, follow these steps to set up the other Quay nodes:

1. Install Red Hat Enterprise Linux on the other Quay nodes (quay02 and quay03, in this example) and create the same configuration and storage directories.

2. Copy the config.yaml file from the initial Quay systems to the others. For example:

```
# scp /mnt/quay/config/config.yaml quay02:/mnt/quay/config/
# scp /mnt/quay/config/config.yaml quay03:/mnt/quay/config/
```

3. Run the quay and redis containers on the other nodes as you did on the first node. The configuration information will be used from the config.yaml file.

CHAPTER 6. START USING RED HAT QUAY

With Red Hat Quay now running, you can:

- Select Tutorial from the Quay home page to try the 15-minute tutorial. In the tutorial, you learn to log into Quay, start a container, create images, push repositories, view repositories, and change repository permissions with Quay.
- Refer to the Use Red Hat Quay for information on working with Red Hat Quay repositories.

ADDITIONAL RESOURCES