Red Hat Quay 2.9

Use Red Hat Quay

Use Red Hat Quay

Last Updated: 2018-09-14

Red Hat Quay 2.9 Use Red Hat Quay

Use Red Hat Quay

Legal Notice

Abstract

Learn to use Red Hat Quay

Table of Contents

PREFACE

Whether you deployed your own Red Hat Quay service or are using the Quay.io registry, follow descriptions here to start using your Quay repository to store and work with images.

CHAPTER 1. CREATING A REPOSITORY

There are two ways to create a repository in Quay.io: via a **`docker push`** and via the Quay.io UI.

1.1. CREATING AN IMAGE REPOSITORY VIA THE UI

To create a repository in the Quay.io UI, click the **+** icon in the top right of the header on any Quay.io page and choose **`New Repository`**. Select **`Container Image Repository`** on the next page, choose a namespace (only applies to organizations), enter a repository name and then click the **`Create Repository`** button. The repository will start out empty unless a **`Dockerfile`** is uploaded as well.

1.2. CREATING AN IMAGE REPOSITORY VIA DOCKER

First, tag the repository:

```
# docker tag 0u123imageid quay.io/namespace/repo_name
```

Then push to Quay.io:

```
# docker push quay.io/namespace/repo_name
```

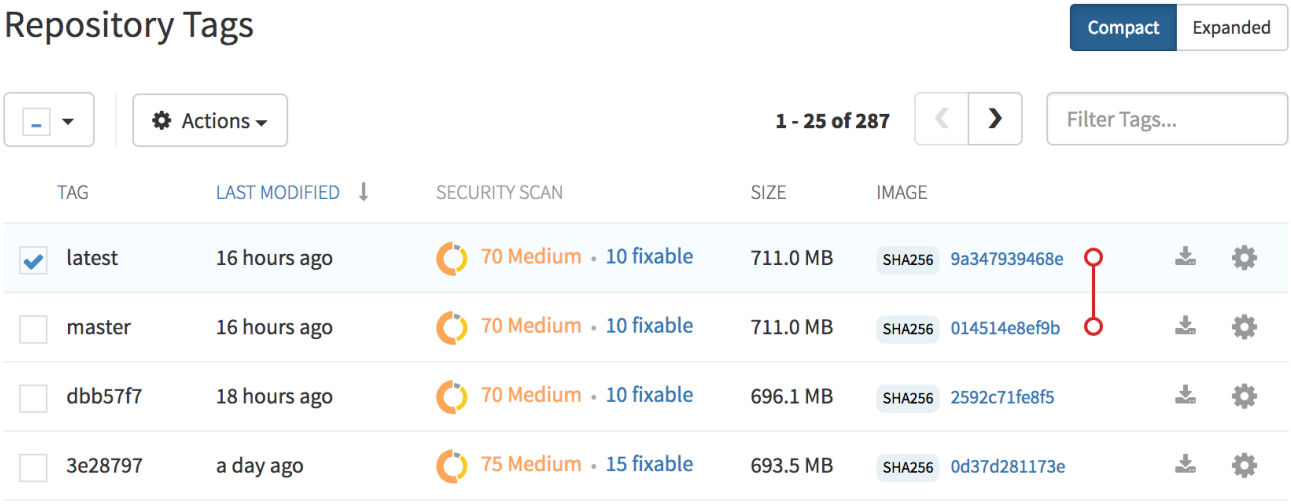
1.3. CREATING AN APPLICATION REPOSITORY VIA THE UI

To create a repository in the Quay.io UI, click the **+** icon in the top right of the header on any Quay.io page and choose **`New Repository`**. Select **`Application Repository`** on the next page, choose a namespace (only applies to organizations), enter a repository name and then click the **`Create Repository`** button. The repository will start out empty.

CHAPTER 2. WORKING WITH TAGS

2.1. VIEWING AND MODIFYING TAGS

The tags of a repository can be viewed and modified in the tags panel of the repository page, found by clicking on the **Tags** tab.



2.1.1. Adding a new tag to a tagged image

A new tag can be added to a tagged image by clicking on the gear icon next to the tag and choosing **Add New Tag**. Quay.io will confirm the addition of the new tag to the image.

2.1.2. Moving a tag

Moving a tag to a different image is accomplished by performing the same operation as adding a new tag, but giving an existing tag name. Quay.io will confirm that you want the tag moved, rather than added.

2.1.3. Deleting a tag

A specific tag and all its images can be deleted by clicking on the tag's gear icon and choosing **Delete Tag**. This will delete the tag and any images unique to it. Images will not be deleted until no tag references them either directly or indirectly through a parent child relationship.

2.1.4. Viewing tag history and going back in time

2.1.4.1. Viewing tag history

To view the image history for a tag, click on the **View Tags History** menu item located under the **Actions** menu. The page shown will display each image to which the tag pointed in the past and when it pointed to that image.

2.1.4.2. Going back in time

To revert the tag to a previous image, find the history line where your desired image was overwritten, and click on the Restore link.

2.2. SECURITY SCANNING

By clicking the on the vulnerability or fixable count next to a tab you can jump into the security scanning information for that tag. There you can find which CVEs your image is susceptible to, and what remediation options you may have available.

CHAPTER 3. SETTING UP A CUSTOM GIT TRIGGER

A Custom Git Trigger is a generic way for any git server to act as a build trigger. It relies solely on SSH keys and webhook endpoints; everything else is left to the user to implement.

3.1. CREATING A TRIGGER

Creating a Custom Git Trigger is similar to the creation of any other trigger with a few subtle differences:

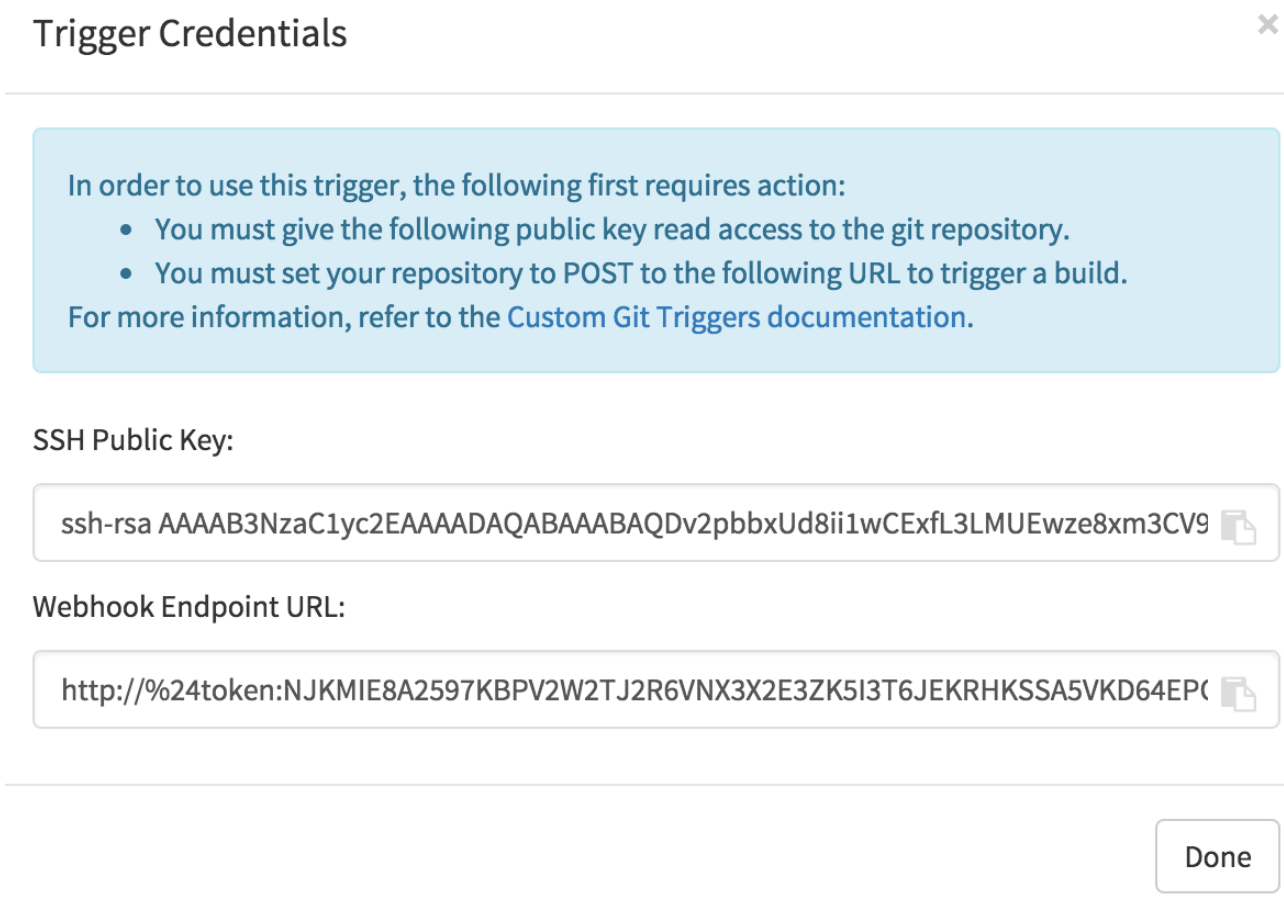
- It is not possible for Quay.io to automatically detect the proper robot account to use with the trigger. This must be done manually in the creation process.
- There are extra steps after the creation of the trigger that must be done in order to use the trigger. These steps are detailed below.

3.2. POST TRIGGER-CREATION SETUP

Once a trigger has been created, **there are 2 additional steps required** before the trigger can be used:

- Provide read access to the *SSH public key* generated when creating the trigger.
- Setup a *webhook* that POSTs to the Quay.io endpoint to trigger a build.

The key and the URL are both available at all times by selecting **`View Credentials`** from the gear located in the trigger listing.



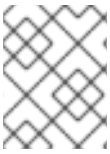
3.2.1. SSH public key access

Depending on the Git server setup, there are various ways to install the SSH public key that Quay.io generates for a custom git trigger. For example, Git documentation describes a small server setup in which simply adding the key to **`$HOME/.ssh/authorize_keys`** would provide access for builders to clone the repository. For any git repository management software that isn't officially supported, there is usually a location to input the key often labeled as **`Deploy Keys`**.

3.2.2. Webhook

In order to automatically trigger a build, one must POST a JSON payload to the webhook URL with the following format:

```
{
  "commit": "1c002dd",                                    // required
  "ref": "refs/heads/master",                             // required
  "default_branch": "master",                             // required
  "commit_info": {                                        // optional
    "url": "gitsoftware.com/repository/commits/1234567", // required
    "message": "initial commit",                          // required
    "date": "timestamp",                                  // required
    "author": {                                           // optional
      "username": "user",                                 // required
      "avatar_url": "gravatar.com/user.png",              // required
      "url": "gitsoftware.com/users/user"                 // required
    },
    "committer": {                                        // optional
      "username": "user",                                 // required
      "avatar_url": "gravatar.com/user.png",              // required
      "url": "gitsoftware.com/users/user"                 // required
    }
  }
}
```



NOTE

This request requires a **`Content-Type`** header containing **`application/json`** in order to be valid.

Once again, this can be accomplished in various ways depending on the server setup, but for most cases can be done via a post-receive git hook.

CHAPTER 4. SKIPPING A SOURCE CONTROL-TRIGGERED BUILD

To specify that a commit should be ignored by the Quay build system, add the text **`[skip build]`** or **`[build skip]`** anywhere in the commit message.

CHAPTER 5. REPOSITORY NOTIFICATIONS

Quay supports adding *notifications* to a repository for various events that occur in the repository's lifecycle. To add notifications, click the **`Settings`** tab in the Repository View.

NOTE

Adding notifications requires *repository admin permission*.

5.1. REPOSITORY EVENTS

5.1.1. Repository Push

A successful push of one or more images was made to the repository:

```
{
  "name": "repository",
  "repository": "mynamespace/repository",
  "namespace": "mynamespace",
  "docker_url": "quay.io/mynamespace/repository",
  "homepage": "https://quay.io/repository/mynamespace/repository",
  "updated_tags": [
    "latest"
  ]
}
```

5.1.2. Dockerfile Build Queued

A Dockerfile build has been queued into the build system:

```
{
  "repository": "mynamespace/repository",
  "namespace": "mynamespace",
  "name": "repository",
  "docker_url": "quay.io/mynamespace/repository",
  "homepage": "https://quay.io/repository/mynamespace/repository/build?
current=some-fake-build",

  "is_manual": false,
  "build_id": "build_uuid",
  "docker_tags": ["latest", "foo", "bar"],

  "trigger_kind": "github",                                    //
Optional
  "trigger_id": "some-id-here",                                //
Optional
  "trigger_metadata": {                                        //
Optional
    "default_branch": "master",
    "ref": "refs/heads/somebranch",
    "commit": "42d4a62c53350993ea41069e9f2cfdefb0df097d",
    "commit_info": {                                           //
Optional
```

```
      "url": "http://path/to/the/commit",
      "message": "Some commit message",
      "date": 2395748365,
      "author": {                                                    //
Optional
        "username": "fakeauthor",
        "url": "http://path/to/fake/author/in/scm",                  //
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"      //
Optional
      },
      "committer": {                                                 //
Optional
        "username": "fakecommitter",
        "url": "http://path/to/fake/comitter/in/scm",                //
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"      //
Optional
      }
    }
  }
}
```

5.1.3. Dockerfile Build Started

A Dockerfile build has been started by the build system

```
{
  "repository": "mynamespace/repository",
  "namespace": "mynamespace",
  "name": "repository",
  "docker_url": "quay.io/mynamespace/repository",
  "homepage": "https://quay.io/repository/mynamespace/repository/build?
current=some-fake-build",

  "build_id": "build_uuid",
  "docker_tags": ["latest", "foo", "bar"],

  "trigger_kind": "github",                                          //
Optional
  "trigger_id": "some-id-here",                                      //
Optional
  "trigger_metadata": {                                              //
Optional
    "default_branch": "master",
    "ref": "refs/heads/somebranch",
    "commit": "42d4a62c53350993ea41069e9f2cfdefb0df097d",
    "commit_info": {                                                 //
Optional
      "url": "http://path/to/the/commit",
      "message": "Some commit message",
      "date": 2395748365,
      "author": {                                                    //
Optional
        "username": "fakeauthor",
```

```
        "url": "http://path/to/fake/author/in/scm",                //
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"    //
Optional
      },
      "committer": {                                               //
Optional
        "username": "fakecommitter",
        "url": "http://path/to/fake/comitter/in/scm",              //
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"    //
Optional
      }
    }
  }
}
```

5.1.4. Dockerfile Build Successfully Completed

A Dockerfile build has been successfully completed by the build system

NOTE

This event will occur **simultaneously** with a *Repository Push* event for the built image(s)

```
{
  "repository": "mynamespace/repository",
  "namespace": "mynamespace",
  "name": "repository",
  "docker_url": "quay.io/mynamespace/repository",
  "homepage": "https://quay.io/repository/mynamespace/repository/build?
current=some-fake-build",
  "visibility": "public",

  "build_id": "build_uuid",
  "docker_tags": ["latest", "foo", "bar"],

  "trigger_kind": "github",                                        //
Optional
  "trigger_id": "some-id-here",                                    //
Optional
  "trigger_metadata": {                                            //
Optional
    "default_branch": "master",
    "ref": "refs/heads/somebranch",
    "commit": "42d4a62c53350993ea41069e9f2cfdefb0df097d",
    "commit_info": {                                               //
Optional
      "url": "http://path/to/the/commit",
      "message": "Some commit message",
      "date": 2395748365,
      "author": {                                                  //
Optional
        "username": "fakeauthor",
        "url": "http://path/to/fake/author/in/scm",                //
```

```
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"   //
Optional
      },
      "committer": {                                              //
Optional
        "username": "fakecommitter",
        "url": "http://path/to/fake/comitter/in/scm",              //
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"   //
Optional
      }
    }
  }
}
```

5.1.5. Dockerfile Build Failed

A Dockerfile build has failed

```
{
  "repository": "mynamespace/repository",
  "namespace": "mynamespace",
  "name": "repository",
  "docker_url": "quay.io/mynamespace/repository",
  "homepage": "https://quay.io/repository/mynamespace/repository/build?
current=some-fake-build",

  "build_id": "build_uuid",
  "docker_tags": ["latest", "foo", "bar"],

  "error_message": "This is the reason the build failed",

  "trigger_kind": "github",                                       //
Optional
  "trigger_id": "some-id-here",                                   //
Optional
  "trigger_metadata": {                                           //
Optional
    "default_branch": "master",
    "ref": "refs/heads/somebranch",
    "commit": "42d4a62c53350993ea41069e9f2cfdefb0df097d",
    "commit_info": {                                              //
Optional
      "url": "http://path/to/the/commit",
      "message": "Some commit message",
      "date": 2395748365,
      "author": {                                                 //
Optional
        "username": "fakeauthor",
        "url": "http://path/to/fake/author/in/scm",               //
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"   //
Optional
      },
```

```
      "committer": {                                                  //
Optional
        "username": "fakecommitter",
        "url": "http://path/to/fake/comitter/in/scm",                 //
Optional
        "avatar_url": "http://www.gravatar.com/avatar/fakehash"       //
Optional
      }
    }
  }
}
```

5.1.6. Vulnerability Detected

A vulnerability was detected in the repository

```
{
  "repository": "mynamespace/repository",
  "namespace": "mynamespace",
  "name": "repository",
  "docker_url": "quay.io/mynamespace/repository",
  "homepage": "https://quay.io/repository/mynamespace/repository",

  "tags": ["latest", "othertag"],

  "vulnerability": {
    "id": "CVE-1234-5678",
    "description": "This is a bad vulnerability",
    "link": "http://url/to/vuln/info",
    "priority": "Critical",
    "has_fix": true
  }
}
```

5.2. NOTIFICATION ACTIONS

5.2.1. Quay Notification

A notification will be added to the Quay.io notification area. The notification area can be found by clicking on the bell icon in the top right of any Quay.io page.

Quay.io notifications can be setup to be sent to a *User*, *Team*, or the *organization* as a whole.

5.2.2. E-mail

An e-mail will be sent to the specified address describing the event that occurred.

NOTE

All e-mail addresses will have to be verified on a *per-repository* basis

5.2.3. Webhook POST

An HTTP POST call will be made to the specified URL with the event's data (see above for each event's data format).

When the URL is HTTPS, the call will have an SSL client certificate set from Quay.io. Verification of this certificate will prove the call originated from Quay.io. Responses with status codes in the 2xx range are considered successful. Responses with any other status codes will be considered failures and result in a retry of the webhook notification.

5.2.4. Hipchat Notification

Posts a message to HipChat.

5.2.5. Slack Notification

Posts a message to Slack.

5.2.6. Flowdock Notification

Posts a message to Flowdock.

CHAPTER 6. BUILDING DOCKERFILES

Quay.io supports the ability to build Dockerfiles on our build fleet and push the resulting image to the repository.

6.1. VIEWING AND MANAGING BUILDS

Repository Builds can be viewed and managed by clicking the Builds tab in the **`Repository View`**.

6.2. MANUALLY STARTING A BUILD

To manually start a repository build, click the **+** icon in the top right of the header on any repository page and choose **`New Dockerfile Build`**. An uploaded **`Dockerfile`**, **`.tar.gz`**, or an HTTP URL to either can be used for the build.

NOTE

You will not be able to specify the Docker build context when manually starting a build.

6.3. BUILD TRIGGERS

Repository builds can also be automatically triggered by events such as a push to an SCM (GitHub, BitBucket or GitLab) or via a call to a webhook.

6.3.1. Creating a new build trigger

To setup a build trigger, click the **`Create Build Trigger`** button on the Builds view page and follow the instructions of the dialog. You will need to grant Quay.io access to your repositories in order to setup the trigger and your account *requires admin access on the SCM repository*.

6.3.2. Manually triggering a build trigger

To trigger a build trigger manually, click the icon next to the build trigger and choose **`Run Now`**.

6.3.3. Build Contexts

When building an image with Docker, a directory is specified to become the build context. This holds true for both manual builds and build triggers because the builds conducted by Quay.io are no different from running **`docker build`** on your own machine.

Quay.io build contexts are always the specified *subdirectory* from the build setup and fallback to the root of the build source if none is specified. When a build is triggered, Quay.io build workers clone the git repository to the worker machine and enter the build context before conducting a build.

For builds based on tar archives, build workers extract the archive and enter the build context. For example:

```
example
├── .git
├── Dockerfile
```

```
├── file
└── subdir
    └── Dockerfile
```

Imagine the example above is the directory structure for a GitHub repository called "example". If no subdirectory is specified in the build trigger setup or while manually starting a build, the build will operate in the example directory.

If **`subdir`** is specified to be the subdirectory in the build trigger setup, only the Dockerfile within it is visible to the build. This means that you cannot use the **`ADD`** command in the Dockerfile to add **`file`**, because it is outside of the build context.

Unlike the Docker Hub, the Dockerfile is part of the build context on Quay. Thus, it must not appear in the **`.dockerignore`** file.

CHAPTER 7. DOWNLOADING SQUASHED DOCKER IMAGES

Docker images are composed of image layers which include all of the intermediary data used to reach their current state. When iterating on a solution locally on a developer's machine, layers provide an efficient workflow.

There are scenarios, however, in which the layers cease to be efficient. For example, when deploying software to an ephemeral machine, that machine doesn't care about the whole layer history, it just needs the end state of the image. This is why Quay.io supports *Squashed Images*.

7.1. DOWNLOADING A SQUASHED IMAGE

To download a squashed image:

1. Navigate to the **`Tags`** tab of a Quay **`Repository View`** (**`https://quay.io/repository/YOURORG/YOURREPO?tab=tags`**).
2. On the left side of the table, click on the *Fetch Tag* icon for the tag you want to download. A modal dialog appears with a dropdown for specifying the desired format of the download.
3. Select **`Squashed Docker Image`** from the dropdown and then select a robot that has *read* permission to be able to pull the repository.
4. Click on the **`Copy Command`** button and paste this command into the shell of the machine you want to download.

7.2. CAVEATS & WARNINGS

7.2.1. Prime the cache!

When the first pull of a squashed image occurs, the registry streams the image as it is being flattened in real time. Afterwards, the end result is cached and served directly. Thus, it is recommended to pull the first squashed image on a developer machine before deploying, so that all of the production machines can pull the cached result.

7.2.2. Isn't piping curl insecure?

You may be familiar with installers that pipe curl into bash (**`curl website.com/installer | /bin/bash`**). These scripts are insecure because they allow arbitrary code execution. The Quay script to download squashed images uses **`curl`** to download a tarball that is streamed into **`docker load`**. This is just as secure as running **`docker pull`** because it never executes anything we've downloaded from the internet.

ADDITIONAL RESOURCES