RED HAT®
CONSULTING

redhat.

Creating a Local Quay Registry

Travis Michette

Version 1.0

Table of Contents

1. Quay Setup Playbooks for Home Lab

This guide is meant to provide instructions on how to use the playbooks in this project. The playbooks will deploy a local instance of Quay Container registry as containers running on the specified Ansible managed hosts.

The process automates the https://access.redhat.com/documentation/en-us/red_hat_quay/3.5/html/deploy_red_hat_quay_for_proof-of-concept_non-production_purposes (Quay Proof of Concept) process from Red Hat.

1.1. Preparing the System and Deploying Quay

1.1.1. Preparing a System for Quay

These instructions have been tested on a RHEL8 system that has already been subscripted to a content repository and has the ability to download and install packages. The playbooks will attempt to install **podman** and any other dependencies needed for deployment of Quay and its supporting containers.

Storage Space and Considerations

If you are using this for a **production** or operational workload, you will want to give considerations to the system running and hosting the containers for back-end storage.

For these instructions, we are mounting the **/quay** directory from the host filesystem and other directories under there to be used as persistent storage for the running containers. This is also where the container images will be stored. It is recommended to have this space as a separate mount point (LVM) so it can be easily expanded as your registry grows.

For the test and demonstration environment, I've used a local virtual machine named **quay.local** and gave the system the following resources:

- 64GB Virtual Disk
- 6 vCPUs
- 16GB RAM

Another important requirement before beginning the lab and deployment is to ensure that SSH keys for the root user have been copied to your system that will be hosting the Quay containers and that you've modified the **/etc/hosts** file on you Ansible control node so that it can properly deployed the containers based on the playbooks.

1. Copy SSH key from Ansible control node to the Quay host
2. Modify the **/etc/hosts** file so that you can reach the Quay server by FQDN.

Listing 1. **/etc/host** *Entry*

```
... output omitted ...

10.211.55.50    quay.local

... output omitted ...
```

3. Clone github repository

Listing 2. Cloning **github repo** *via SSH*

```
git clone git@github.com:tmichett/quay_lab_poc.git
```

4. Copy the **registry_login.yml_example** to **registry_login.yml** and update with your login credentials.

Listing 3. Create a local variable file

```
cp registry_login.yml_example registry_login.yml
```

File is Copied and in **.gitignore** *to prevent credential leaks*

Listing 4. registry_login.yml

```
registry_un: UN_Goes_Here①
registry_pass: Password_Goes_Here②
registry_url: registry.redhat.io
```

① Replace with your **registry.redhat.io** Username

② Replace with your **registry.redhat.io** Password

1.1.2. Quay Deployment Preparation

Run a set of Ansible playbooks to setup the environment with the needed containers and container images to provide support to Quay. This will also allow the Quay container image to be downloaded and enter a configuration mode to create the **quay-config.tar.gz** file.

Demonstration and Instructions **skip** *the Quay Configuration File*

For time purposes, the configuration of the Quay environment is being skipped. It is possible to use these same playbooks, but for the deployment if you choose to use the configuration container, you would modify the **ansible-playbook Quay_Config_Deploy_Files.yml** command to be **ansible-playbook Quay_Config_Deploy_Tar.yml**. This will also require that you have placed the **quay-config.tar.gz** file in the ***files*** directory relative to the Ansible playbook.

1. Run the Quay_Prepare.yml playbook to prepare the system for deploying Quay.

Listing 5. Preparing the System with Correct Packages

```
travis@Traviss-MacBook-Pro quay_lab % ansible-playbook Quay_Prepare.yml


PLAY [Installation of Packages and Preparing the System] ****************************

TASK [Gathering Facts] ***************************************************************
ok: [quay.local]

TASK [Install Podman Packages] *******************************************************
changed: [quay.local]

TASK [Enable Firewall Ports] *********************************************************
changed: [quay.local] => (item=8443/tcp)
changed: [quay.local] => (item=8080/tcp)
changed: [quay.local] => (item=443/tcp)
changed: [quay.local] => (item=5432/tcp)
changed: [quay.local] => (item=6379/tcp)
changed: [quay.local] => (item=5433/tcp)

... output omitted ...

TASK [Stop and Remove the Quay Config Container] *************************************
changed: [quay.local]

PLAY RECAP ***************************************************************************
quay.local                 : ok=13   changed=10   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

The Quay Configuration Container

The playbook will pause to allow you to update or create a new Quay configuration TGZ file. You will be accessing a specialized Quay configuration container at **http://FQDN:8080** to complete a web form. You will be logging in with the passwords that were setup for the playbook. In this instance, it is:

- **Username**: quayconfig
- **Password**: secret

2. Deploy QUAY Configuration Files

Listing 6. Deploy Quay Configuration Files

```
travis@Traviss-MacBook-Pro quay_lab_poc % ansible-playbook Quay_Config_Deploy_Files.yml

PLAY [Deploy Quay after Quay_Prepare.yml Playbook] *********************************

TASK [Gathering Facts] *************************************************************
ok: [quay.local]

TASK [Prepare Config Folder] *******************************************************
changed: [quay.local]

TASK [Deploy Config File] **********************************************************
changed: [quay.local]

TASK [Create "/quay/storage" Directory] ********************************************
changed: [quay.local]

TASK [Set ACL on "/quay/storage"] **************************************************
changed: [quay.local]

PLAY RECAP *************************************************************************
quay.local                 : ok=5    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Direct Config File Manipulation

This allows manual modification of the **config.yaml** file. There is another playbook that will deploy the actual quay-config.tar.gz file. That playbook is **Quay_Config_Deploy_Tar.yml**.

3. Deploy the Clair Scanning Container

Listing 7. Deploy Clair

```
travis@Traviss-MacBook-Pro quay_lab % ansible-playbook Quay_Clair_Deploy.yml

PLAY [Deploy Quay Claire Image Scanning Service] ***********************************

TASK [Gathering Facts] *************************************************************
ok: [quay.local]

... output omitted ...

TASK [Start the Clair Container] ***************************************************
changed: [quay.local]

PLAY RECAP *************************************************************************
quay.local                 : ok=10   changed=8    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Wait for about three (3) minutes before Clair is up

Sometimes it takes a while for Clair to come up. If Clair isn't fully up and operational before you attempt deploying the **Quay** container or the **Quay-Mirror** container, they will both fail because of failure to communicate with the security scanner container.

Listing 8. **podman logs quay** *Snippet*

```
+-----------------------+-------------------------------------------------------------+--------+
| SecurityScanner       | dial tcp 10.211.55.50:8081: connect: connection refused |        |
+-----------------------+-------------------------------------------------------------+--------+
```

If Quay container fails on deployment, it is possible it is taking longer for ClairV4 container to come up and be operational. I've seen it take as long as 15 minutes and not sure why!

4. Deploy the QUAY Container

Listing 9. Deploy Quay

```
travis@Traviss-MacBook-Pro quay_lab % ansible-playbook Quay_Deploy.yml

PLAY [Deploy Quay after Quay_Prepare.yml Playbook] *********************************

TASK [Gathering Facts] *************************************************************
ok: [quay.local]

TASK [Prepare Config Folder] *******************************************************
changed: [quay.local]

... output omitted ...

TASK [Start the Quay Container] ****************************************************
changed: [quay.local]

PLAY RECAP *************************************************************************
quay.local                 : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

5. Deploy the QUAY Mirror Container

Listing 10. Deploy Quay Mirror

```
travis@Traviss-MacBook-Pro quay_lab % ansible-playbook Quay_Mirror_Deploy.yml

PLAY [Deploy Quay Mirror] **********************************************************

TASK [Gathering Facts] *************************************************************
ok: [quay.local]

TASK [Login to Container Registry] *************************************************
changed: [quay.local]

TASK [Start the Quay Container] ****************************************************
changed: [quay.local]

PLAY RECAP *************************************************************************
quay.local                 : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

1.2. Setting Up Quay

After all Quay containers have been configured and installed, it is necessary to setup the Admin (Superuser) for Quay as well as test out the system for both image scanning and the ability to mirror container images from upstream repositories.

1.2.1. Configuring the Quay Super User

After the Quay registry has been deployed, it is important to finish configuring the super users (admins) that were defined as part of the setup and configuration file (**config.yaml**) that was created during the Quay preparation section.

It is necessary to look at the **config.yaml** file and configure these users with a password and create the accounts officially before moving forward with utilizing the Quay container registry and the lab environment.

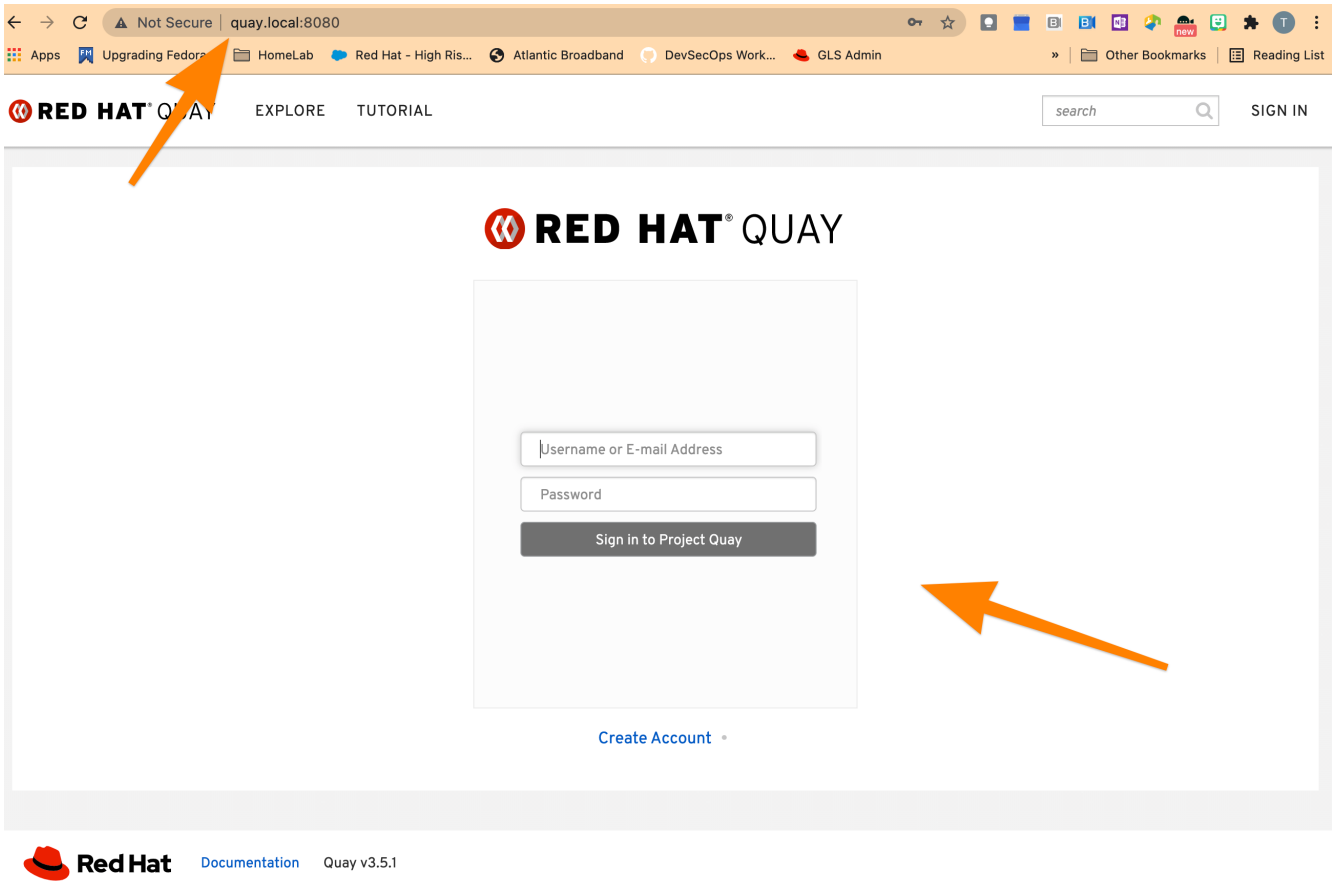
Configure Quay Super Users

It is possible to either look in the configuration file of the **quay-config.tar.gz** or the actual **config.yaml** file for the ***SUPER_USERS*** section. This is where the usernames are defined that will function as Quay super users.

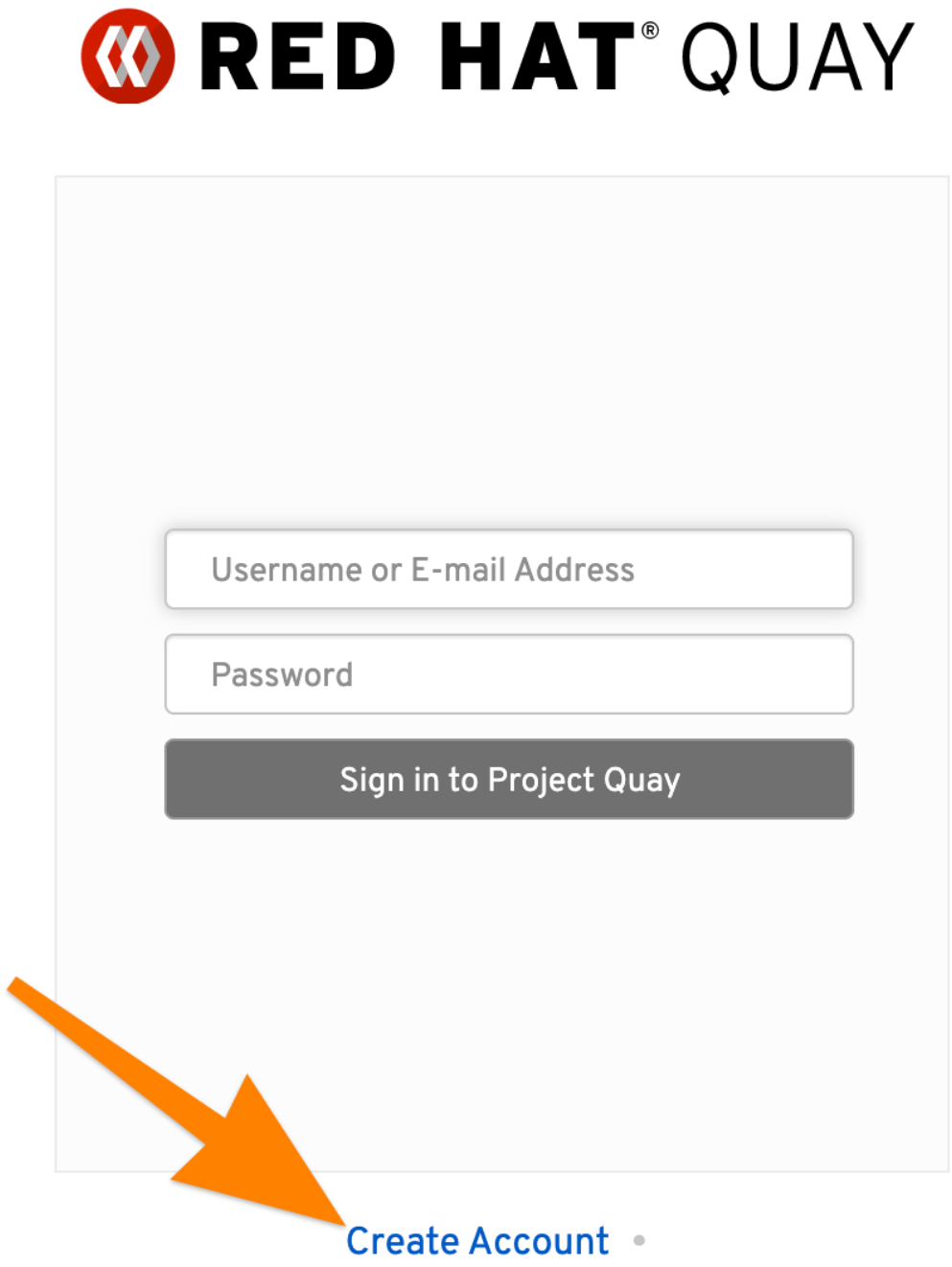
Listing 11. Quay Super Users

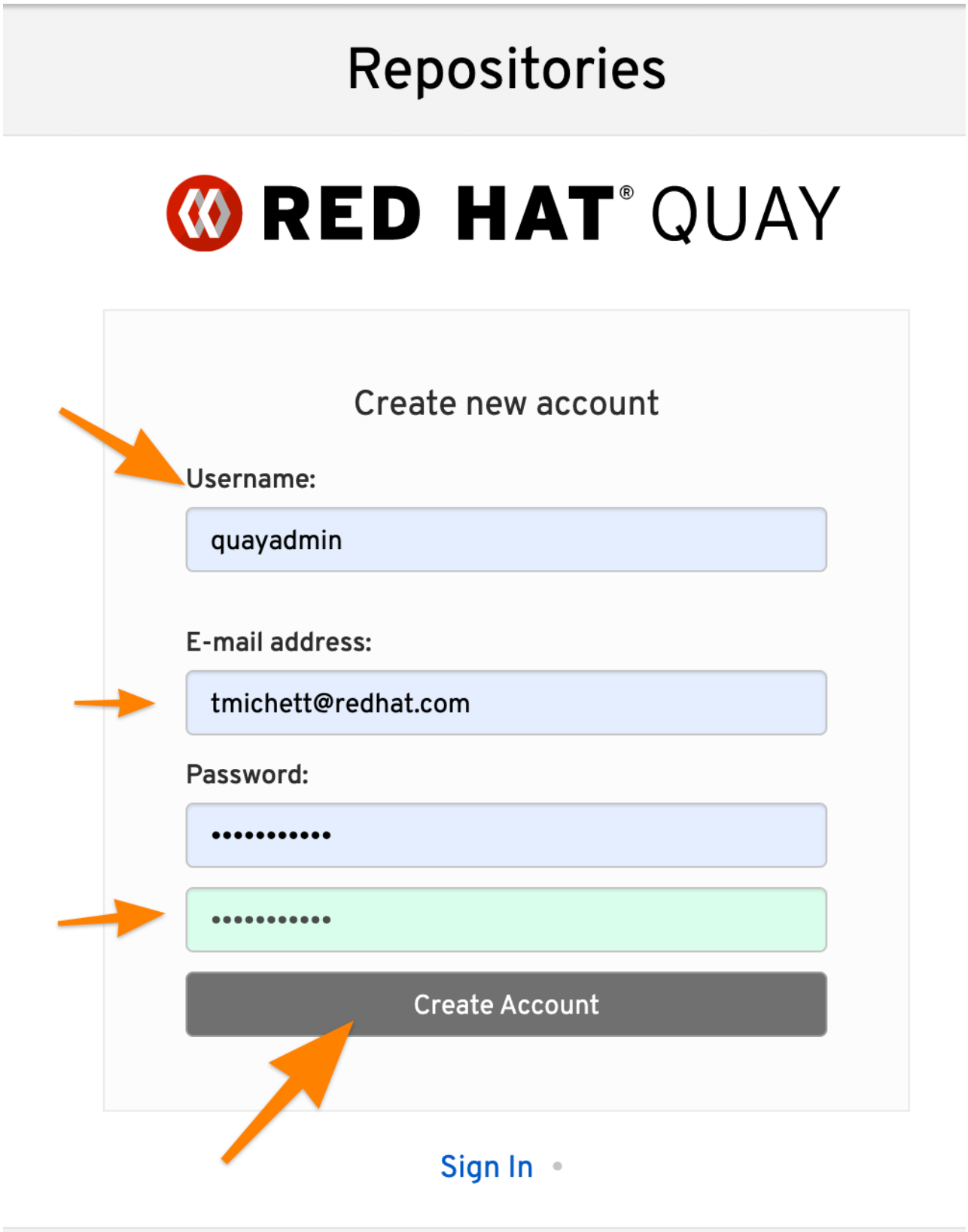
```
SUPER_USERS:
    - quayadmin
    - travis
```

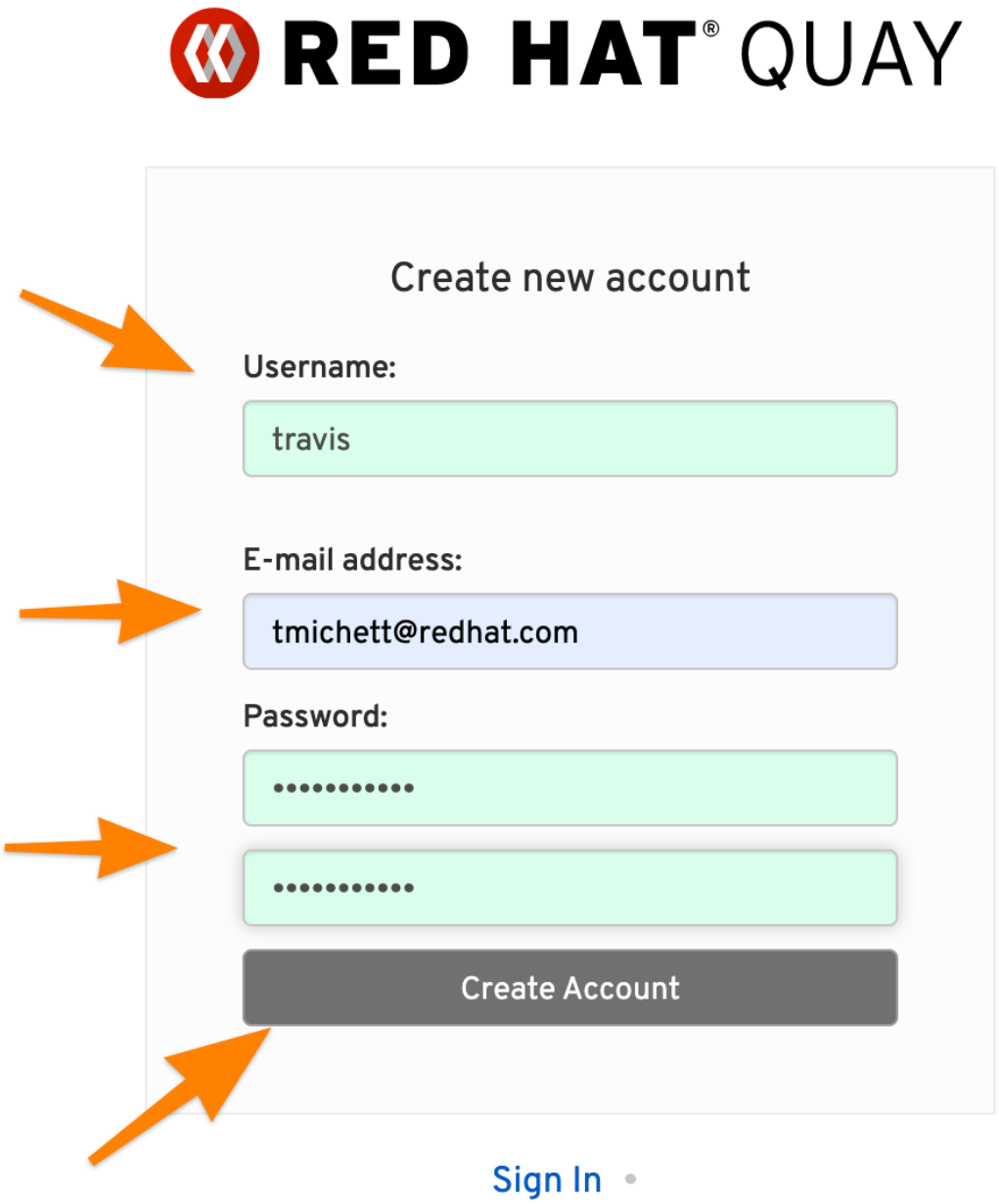
1. Open the Quay web console by navigating to it in your favorite browser using **http://Quay-FQDN:8080**



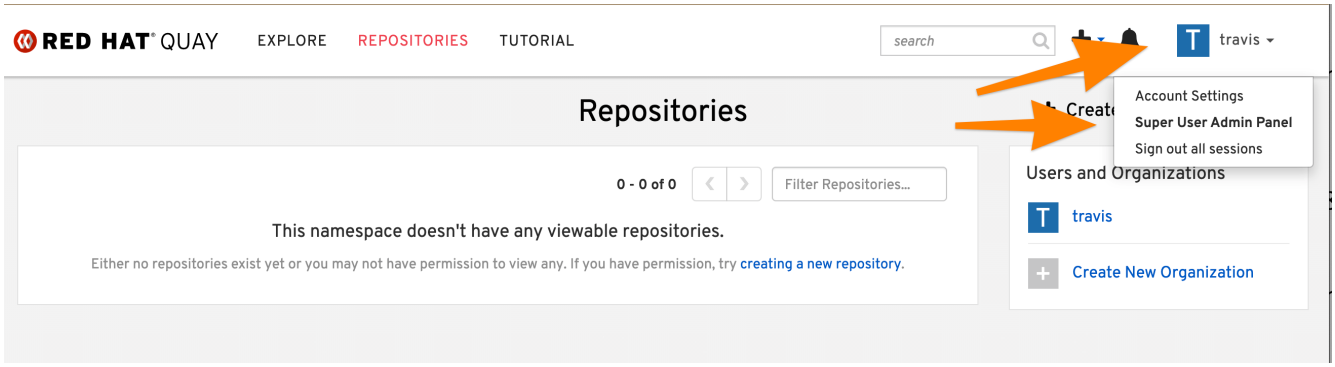
2. Click **Create Account** to create the administrator/superuser accounts for Quay as defined in the **config.yaml** file.
    - Repeat this step for all super users in the **config.yaml** file.


RED HAT® QUAY
Username or E-mail Address
Password
Sign in to Project Quay
Create Account


Repositories
RED HAT® QUAY
Create new account
Username:
quayadmin
E-mail address:
tmichett@redhat.com
Password:
Create Account
Sign In



3. Verify the account was setup properly and you have **Super User** rights by clicking your Username and looking for **Super User Admin Panel**.



1.2.2. Testing Clair Image Scanner

In order to test the scanning capabilities and ensure that things function properly, update a basic image into the Quay Repository

1. Login to Quay Repository

Listing 12. **podman** *Authentication*

```
[root@quay ~]# podman login --tls-verify=false quay.local:8080
Username: travis
Password:
Login Succeeded!
```

2. Pull and Download an Image, Tag it, then upload to repository

Listing 13. Downloading image

```
[root@quay ~]# podman pull ubuntu:20.04
Resolved "ubuntu" as an alias (/etc/containers/registries.conf.d/000-shortnames.conf)
Trying to pull docker.io/library/ubuntu:20.04...
Getting image source signatures
Copying blob 16ec32c2132b done
Copying config 1318b700e4 done
Writing manifest to image destination
Storing signatures
1318b700e415001198d1bf66d260b07f67ca8a552b61b0da02b3832c778f221b
```

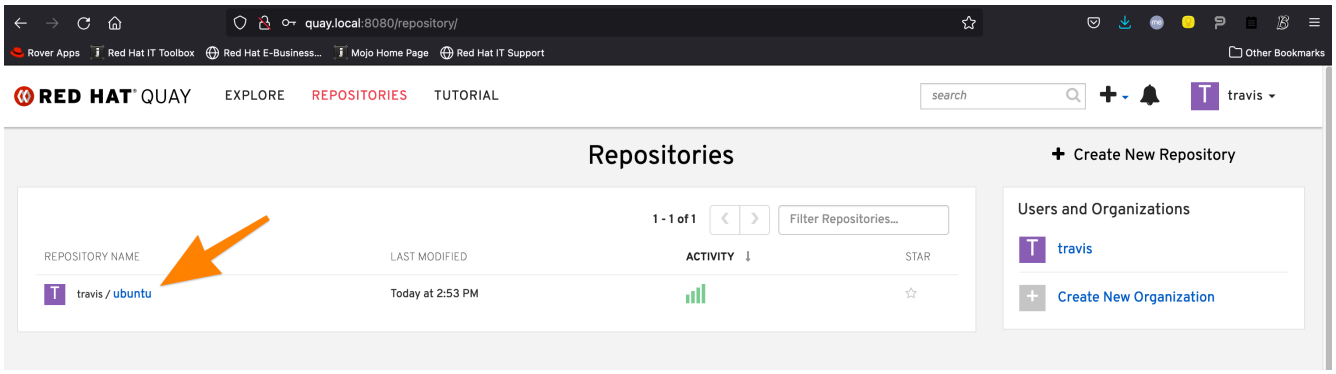
Listing 14. Tagging image

```
[root@quay ~]# podman tag docker.io/library/ubuntu:20.04 quay.local:8080/travis/ubuntu:20.04
```

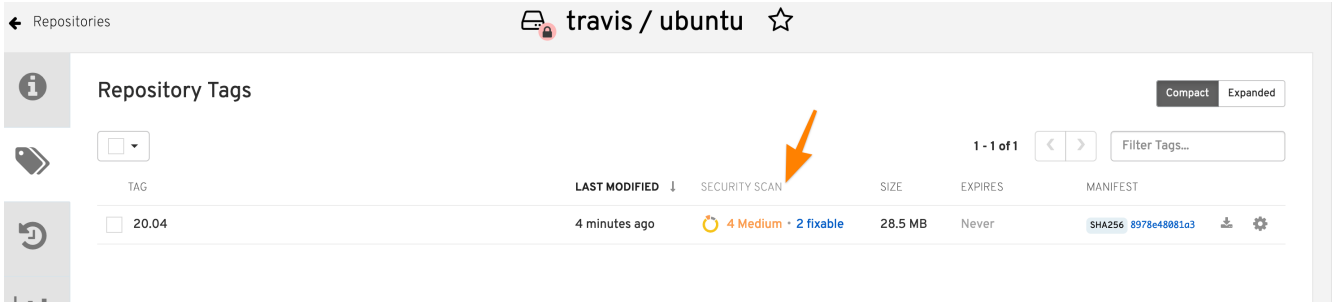
Listing 15. Push image

```
[root@quay ~]# podman push --tls-verify=false quay.local:8080/travis/ubuntu:20.04
Getting image source signatures
Copying blob 7555a8182c42 done
Copying config 1318b700e4 done
Writing manifest to image destination
Storing signatures
```

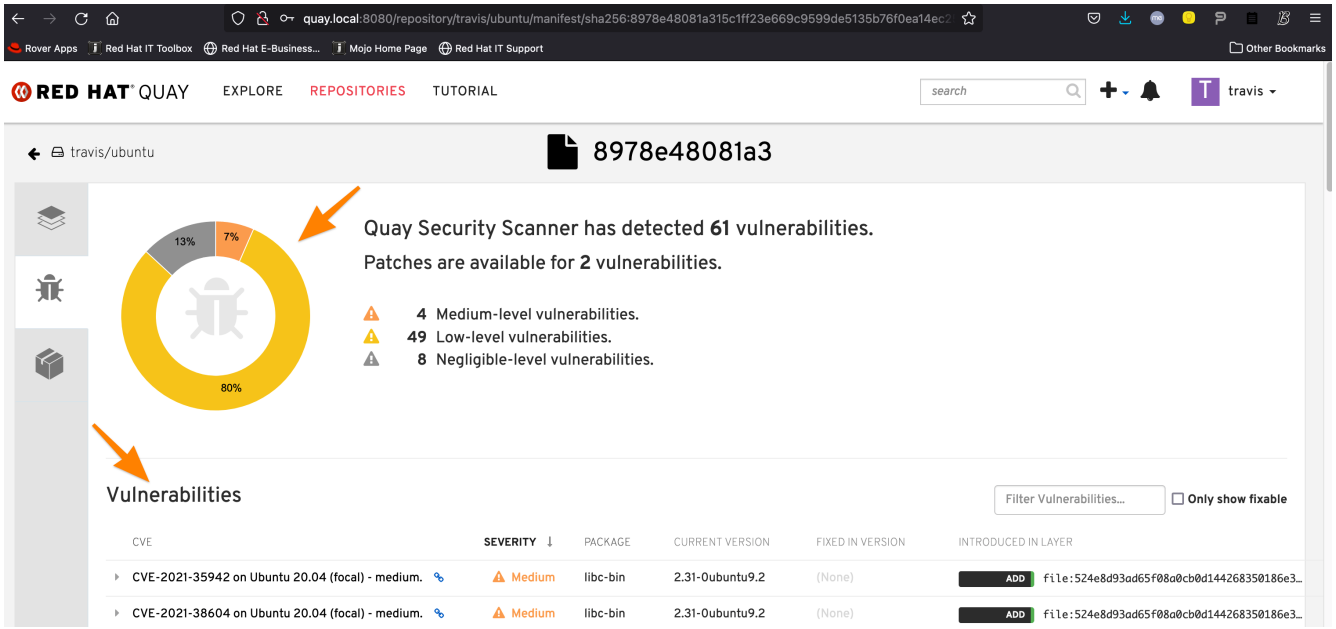
3. Verify image exists in Quay



4. Navigate to image tags and see if the security scan has completed



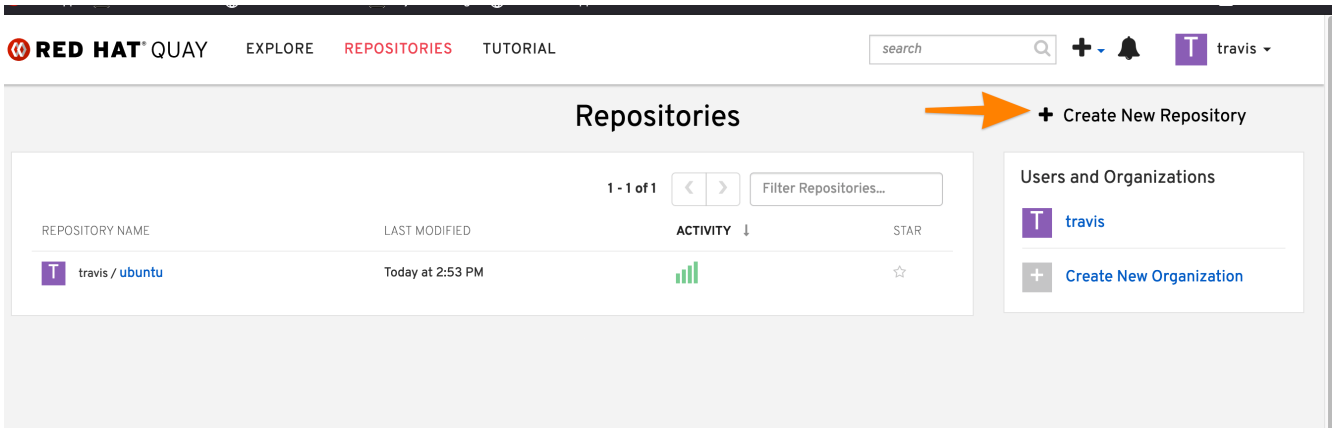
5. Click on Security scan to view the vulnerabilities



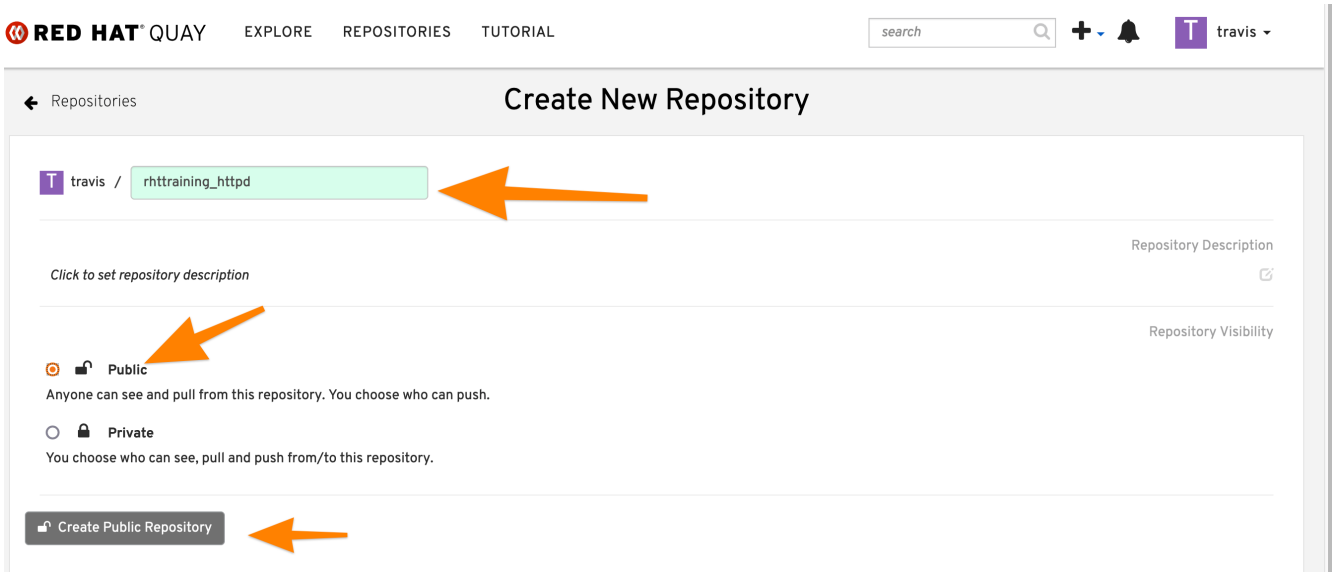
1.2.3. Testing Quay Image Mirroring

The next step is to ensure that the QUAY Image mirroring container is working and that you can successfully mirror container images from upstream repositories.

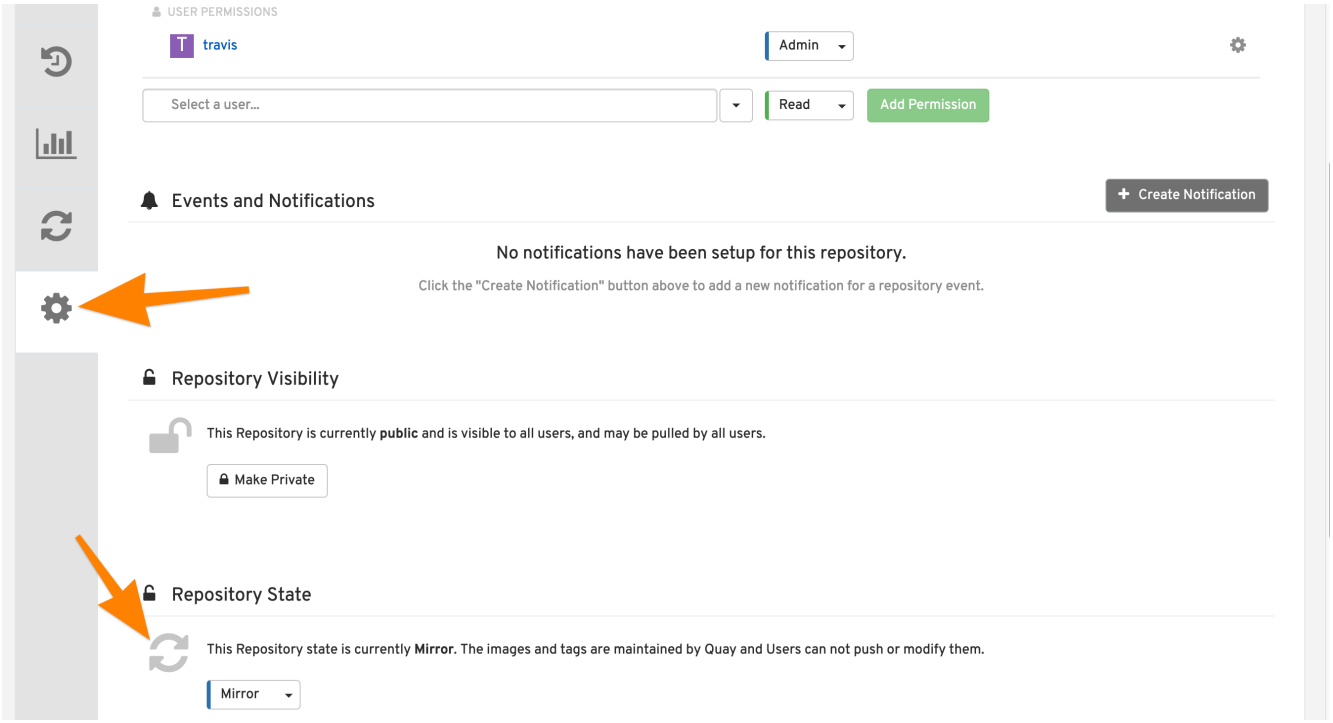
1. Create a new repository in Quay by clicking **Create New Repository**



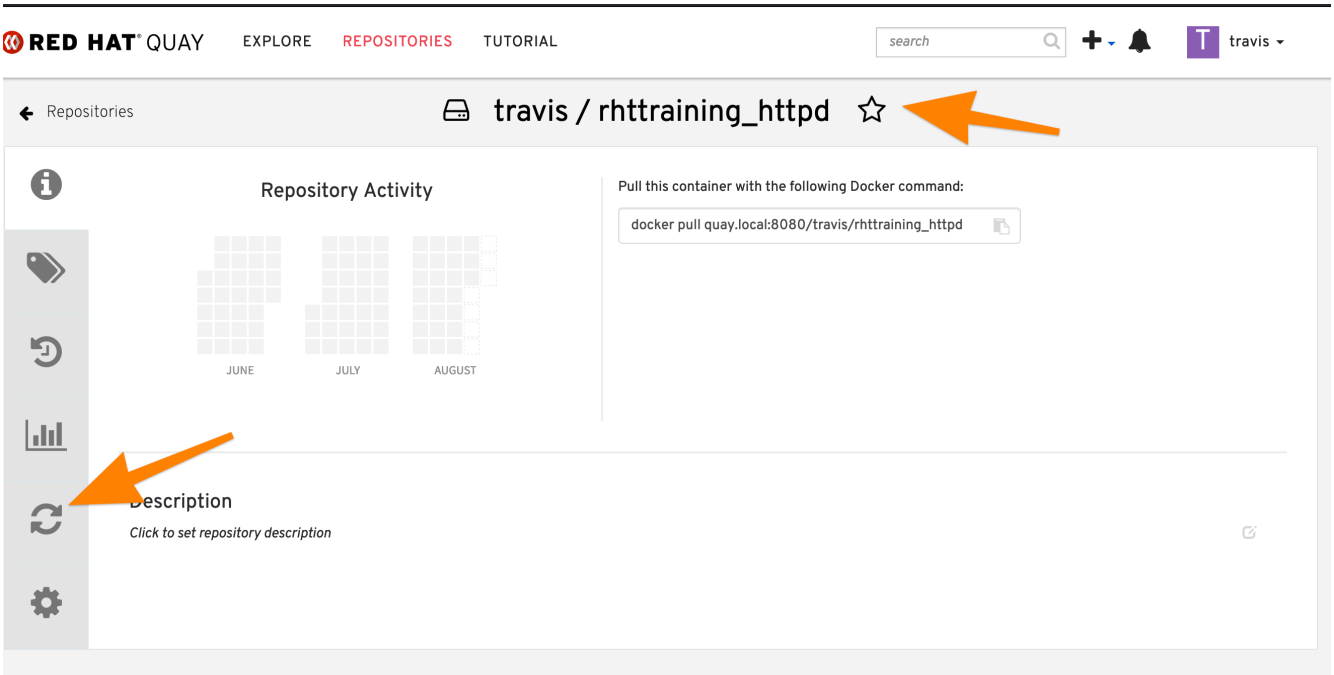
2. Give repository a name and setup the repository visibility



3. In the newly created repository, click the **Settings** option from the left-side navigation menu. Set the **Repository State** to ***Mirror***.



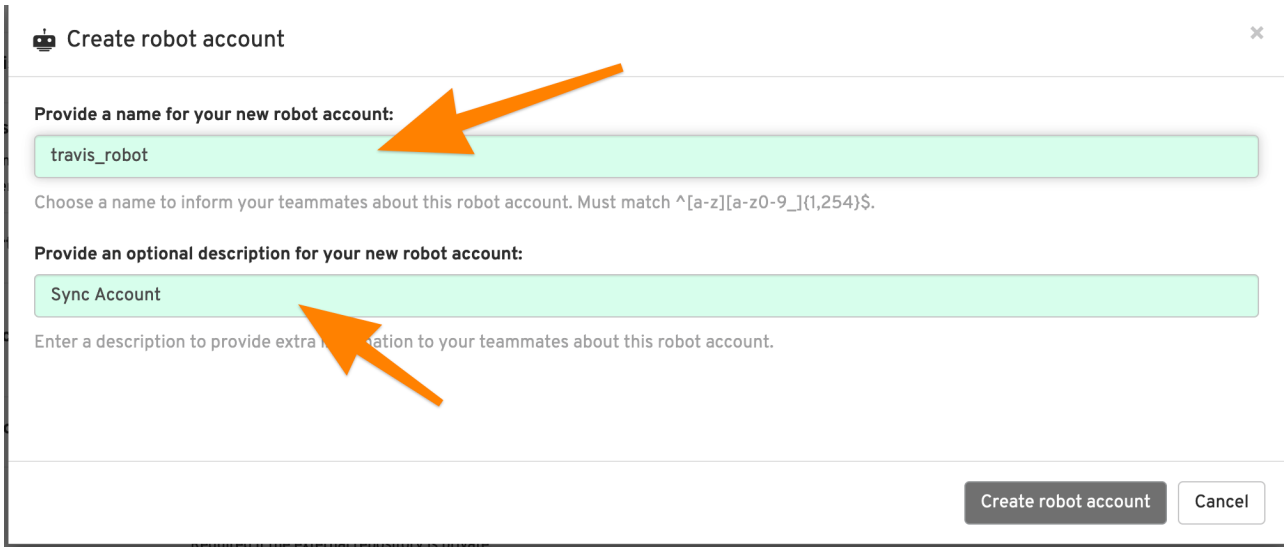
4. In the newly created repository, click the **Mirroring** option from the left-side navigation menu.

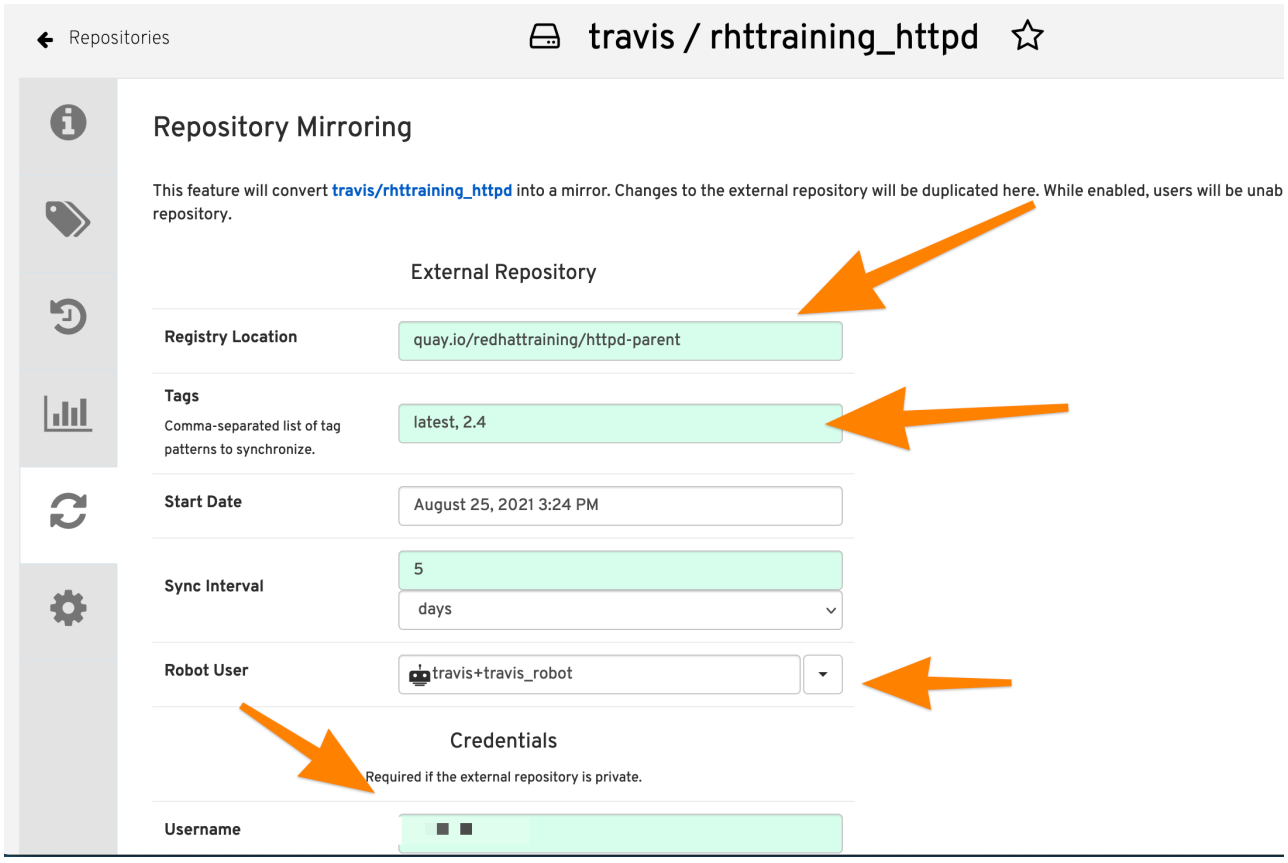


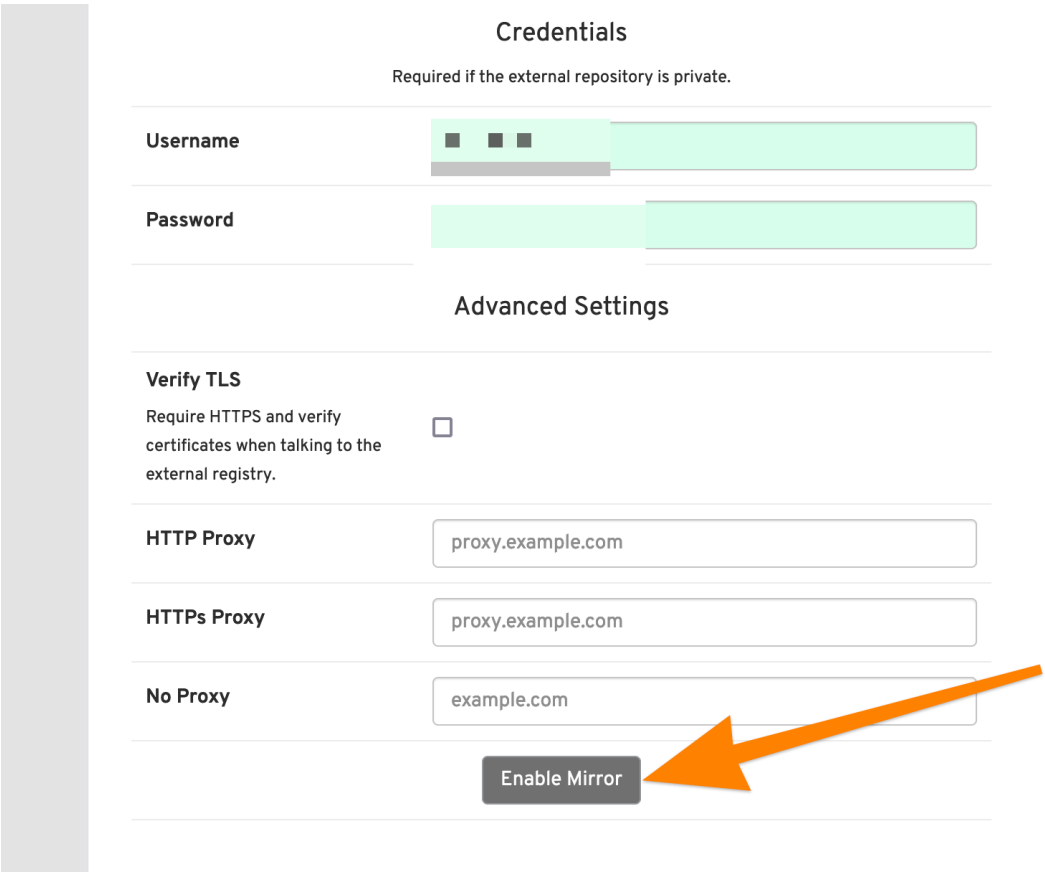
5. In the **Mirroring** tab, complete the required information for the repository and create a **Robot User**. Click **Enable Mirror**

   a. Registry Location - quay.io/redhattraining/httpd-parent

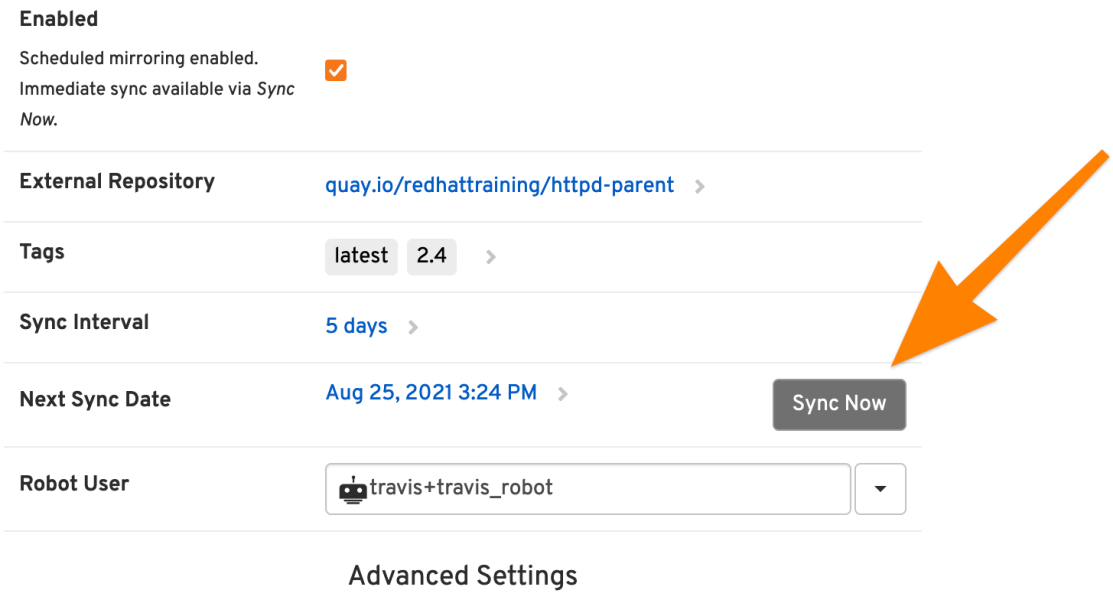
b. Tags: latest and 2.4







6. Click "**Sync Now**" to perform immediate synchronization



7. Verify synchronization completed on the **Mirroring** tab as well as the **Tag History**