Red Hat
Training and
Certification

Managing and Building Container Images and Containers

Travis Michette

Version 1.0

Table of Contents

Introduction

This guide with cover additional and supplemental materials for the DO180 custom container course. As part of this guide, students will be learning the following concepts:

Course Objectives

- Exploring RHEL 8.x Differences
- Exploring SystemD and Creating SystemD Services
- Exploring FirewallD and FirewallD Custom Services
- Exploring Cockpit and the Red Hat Web Console
- Rootless Podman
- Leveraging Podman's Pod Capabilities
- Managing Containers with the Red Hat Web Console (Cockpit)
- Running Containers as a Service (SystemD)
- Building Container Images with Buildah (from scratch)
- Building Container Images from Containerfile/Dockerfile Files
- Installing/Configuring/Using Quay Container Registry
- Using ClairV4 and Quay Mirroring with Quay
- Exploring Skopeo to Interact with Container Images

Courses for Reference

- DO180
- RH134
- RH354

This course will use a DO180 course for the hands-on lab environment. The environment has been modified to have an additional machine to perform custom exercises and have the Quay registry installed locally.

Lab Machine Requirements

In addition to the DO180 lab environment, a new VM has been added to that environment. This machine can be setup and configured locally with the following requirements:

VM Requirements

- RHEL 8.4+
- 6 vCPU (8 vCPU Recommended)
- 12GB RAM (16GB Recommended)
- 60GB Storage (Image and Database storage)

Using the Lab Guide

This lab guide and contents within the guide are meant to supplement the course and materials delivered as part of a custom DO180 delivery. It is advisable to download the RH354 course manual and the RH134 course manual prior to the class delivery. The DO180 course guide can be downloaded as part of the course.

Course Materials

All course materials and lab materials for the custom portion of the course can be found here: https://github.com/tmichett/OCP_Demos

The lab guide for this course can be downloaded from here: https://github.com/tmichett/OCP_Demos/blob/main/Containers/Containers.pdf

1. RHEL 8 Changes

RHEL 8.x brought several significant changes and expanded upon other changes that were introduced as part of the **SystemD** switch in RHEL 7.x. This course will highlight some of the most significant changes and enhancements to RHEL 8.x.

RHEL 8.x Enhancements

- TMUX
- SystemD
- FirewallD

Before beginning the course, there is a quick set of steps that will be completed as an exercise so that the environment can be setup and prepared. The **Workstation** machine will be our GUI machine used throughout the week and is running RHEL 8.2. We will also be using a custom RHEL 8.4 system called **server**.

1.1. Setup Exercise

Before you begin using the customized DO180 Workshop Lab Environment a couple things must be accomplished.

Before you Begin

1. Prepare Workstation
2. SSH Keys must be Generated and Distributed for the root user on **server**
3. Github Projects Must be Downloaded
4. Ansible Playbooks Must be Executed

There will be two Github projects used in this course for the custom exercises and demonstrations.

- **OCP Demos**: https://github.com/tmichett/OCP_Demos - The ***/*Containers** directory contains most of the resources being used for the course.
- **Quay_Lab_PoC**: https://github.com/tmichett/quay_lab_poc - This repository contains playbooks that will setup and deploy Quay locally on the **server** system.

The lab environment will need **server** configured to be our **quay.local** system and will need to have access to all packages and keys.

Example 1. Lab Installation and Setup Exercise

1. Create Directories and Download Github Repos

Listing 1. Creating Directory and Switch to Directory

```
[student@workstation ~]$ mkdir github ; cd github
```

Listing 2. Cloning Repositories - Cloning **quay_lab_poc**

```
[student@workstation github]$ git clone https://github.com/tmichett/quay_lab_poc.git
Cloning into 'quay_lab_poc'...
remote: Enumerating objects: 266, done.
remote: Counting objects: 100% (266/266), done.
remote: Compressing objects: 100% (167/167), done.
remote: Total 266 (delta 67), reused 253 (delta 54), pack-reused 0
Receiving objects: 100% (266/266), 5.90 MiB | 22.90 MiB/s, done.
Resolving deltas: 100% (67/67), done
```

Listing 3. Cloning Repositories - Cloning **OCP_Demos**

```
[student@workstation github]$ git clone https://github.com/tmichett/OCP_Demos.git
Cloning into 'OCP_Demos'...
remote: Enumerating objects: 763, done.
remote: Counting objects: 100% (763/763), done.
remote: Compressing objects: 100% (527/527), done.
remote: Total 763 (delta 329), reused 595 (delta 181), pack-reused 0
Receiving objects: 100% (763/763), 37.16 MiB | 29.22 MiB/s, done.
Resolving deltas: 100% (329/329), done.
```

2. Switch to the **OCP_Demos/Containers/labs/server_prep** Folder

```
[student@workstation github]$ cd OCP_Demos/Containers/labs/server_prep
```

3. Execute the **Setup_Workstation.yml** playbook

```
[student@workstation server_prep]$ ansible-playbook Setup_Workstation.yml
```

4. Create SSH keys

```
[student@workstation server_prep]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:/FXYb+9b7dGD9CJpwTdsE0MqJkitLBuHYJDofp4Jz3w student@workstation.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
|+.   ..      .   |
|oo  . ..    oo   |
|o . o... o ..oo  |
| . + + .o o ..o. |
|.   =   S  o.B  o|
| o o     . .* =.+|
|  B o     .+ o +=|
|   B E    . . .o+|
|    .          .+|
+----[SHA256]-----+
```

5. Copy SSH key for the **root** user to **server**

```
[student@workstation server_prep]$ ssh-copy-id root@server
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/student/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@server's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@server'"
and check to make sure that only the key(s) you wanted were added.
```

6. Execute the **Setup_Server.yml** playbook

```
[student@workstation server_prep]$ ansible-playbook Setup_Server.yml
```

1.2. TMUX Usage

As part of the upgrade process and package replacement process in RHEL8, several packages have not only been deprecated, they've been completely removed. The **screen** package is one package that is no longer available for installation. Instead, a new terminal program **tmux** has been introduced to provide the **screen** functionality as well as other enhancements.

TMUX Usage

TMUX References

Red Hat Learning Community: https://learn.redhat.com/t5/Platform-Linux/Using-tmux-to-execute-commands-on-servers-in-parallel/m-p/2200

Tactical TMUX: https://danielmiessler.com/study/tmux/

In the example below, we will explore the **screen** functionalities of TMUX with respect to attaching and detaching of sessions.

Installing TMUX

1. Install **tmux** with YUM

Listing 4. Installation of TMUX

```
[root@workstation ~]# yum install tmux
Red Hat Enterprise Linux 8.0 AppStream (dvd)     21 MB/s | 5.3 MB     00:00
Red Hat Enterprise Linux 8.0 BaseOS (dvd)        23 MB/s | 2.2 MB     00:00
Last metadata expiration check: 0:00:01 ago on Mon 13 Apr 2020 10:55:47 AM EDT.
Package tmux-2.7-1.el8.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

2. Launch **tmux**

Listing 5. Using **tmux**

```
[student@workstation ~]$ tmux
```

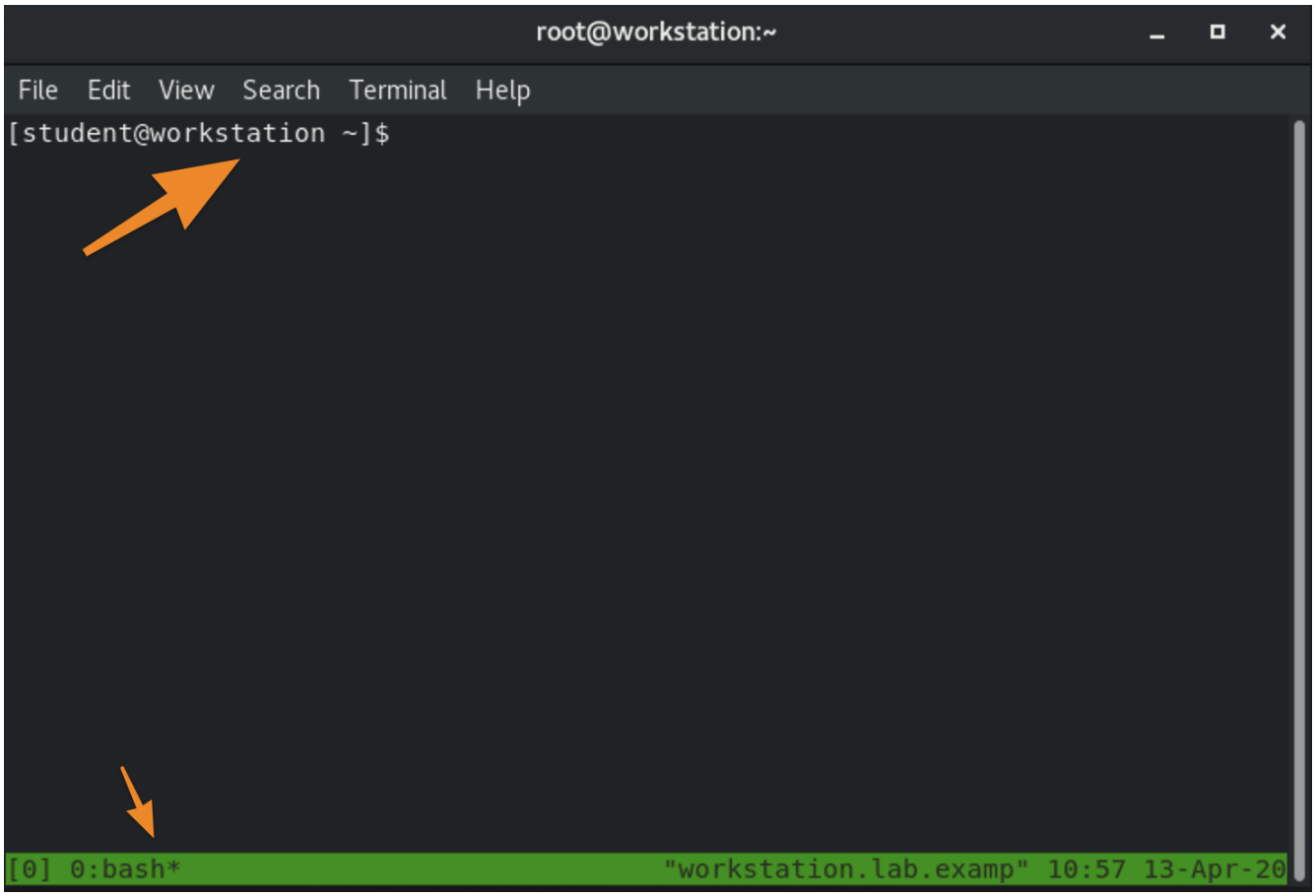


Figure 1. **tmux** *Terminal Window*

3. Placing **tmux** Window in Background (*CTRL+B+D*)

Listing 6. Detaching a **tmux** *Session*

```
[student@workstation ~]$ tmux
[detached (from session 0)]
```

4. Re-attaching to **tmux**

Listing 7. Source Description

```
[student@workstation ~]$ tmux list-sessions
0: 1 windows (created Mon Apr 13 11:01:53 2020) [80x23]

[student@workstation ~]$ tmux attach-session -t 0
```

Naming or Renaming Sessions

It is possible that once you are in a **tmux** session to rename the session.

Listing 8. Renaming a TMUX Session

```
[student@workstation ~]$ tmux rename-session --help
usage: rename-session [-t target-session] new-name

[student@workstation ~]$ tmux rename-session -t 0 travis-demo
```

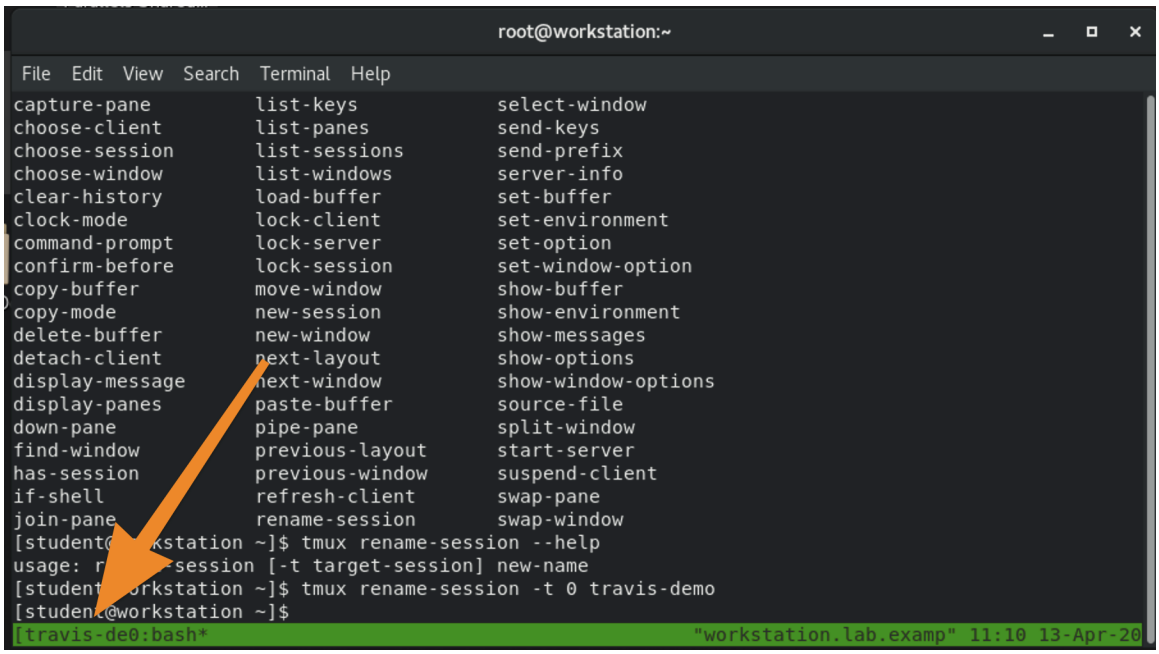


Figure 2. Renamed **tmux** *Session*

```
[student@workstation ~]$ tmux new-session
[detached (from session travis-demo)]

[student@workstation ~]$ tmux list-sessions
travis-demo: 1 windows (created Mon Apr 13 11:08:28 2020) [98x23]

[student@workstation ~]$ tmux attach-session -t travis-demo
```

1.3. SystemD Overview

1.3.1. Understanding SystemD Unit Files and Creating a Service

Starting in RHEL 7, **SystemD** replaced the older style Linux SystemV (sysvinit daemon). This change continued through with RHEL 8 and more systemd tools replaced the traditional legacy tools. This small section will provide references and an overview of how **systemd** can be used to replace the older *init* scripts that were placed in **/etc/init.d/** or some of the other directories.

SystemD References

Linus Torvalds and others on Linux's systemd: https://www.zdnet.com/article/linus-torvalds-and-others-on-linuxs-systemd/

SysVinit Vs systemd Cheatsheet: https://www.2daygeek.com/sysvinit-vs-systemd-cheatsheet-systemctl-command-usage/

1. Create the Init Script

Listing 9. Init Script to Run at Startup

```
[root@server ~]# vim /usr/bin/ups_test_service.sh
#!/usr/bin/bash

DATE=`date '+%Y-%m-%d %H:%M:%S'`
echo "This is a sample service started at ${DATE} for the UPS RH354 course." | systemd-cat -p info

while :
do
echo "Looping...";
sleep 30;
done
```

2. Make script executable

Listing 10. Running **chmod** *on script*

```
[root@server ~]# chmod +x /usr/bin/ups_test_service.sh
```

3. Edit SystemD Service (Unit File)

Listing 11. Editing the .service File

```
[root@server ~]# vim /etc/systemd/system/ups_test_service.service

[Unit]
Description=UPS example systemd service.

[Service]
Type=simple
ExecStart=/bin/bash /usr/bin/ups_test_service.sh

[Install]
WantedBy=multi-user.target
```

4. Fix Permissions on **.service** File

Listing 12. Running **chmod** *on .service File*

```
[root@server ~]# chmod 644 /etc/systemd/system/ups_test_service.service
```

SystemD Services

Once a script has been created and made executable and a service file has properly been created and placed in **/etc/systemd/system/** directory it is possible to use the **systemctl** command to interact with the service file and make it active on boot.

Default SystemD Directories

It is important to note that there are several default SystemD directories. When defining your own files, the proper location is to place them in **/etc/systemd**. There is more information available in other Red Hat courses, specifically the RH442 Performance Tuning course.

Default Location: **/lib/systemd/**

1. Starting and Enabling a Custom Service

Listing 13. Controlling a Custom Service with **systemctl**

```
[root@server ~]# systemctl enable ups_test_service.service --now
Created symlink /etc/systemd/system/multi-user.target.wants/ups_test_service.service → /etc/systemd/system/ups_test_service.service.
```

2. Checking Status of a Custom Service

Listing 14. Using **systemctl** *to Check Service Status*

```
[root@server ~]# systemctl status ups_test_service.service
● ups_test_service.service - UPS example systemd service.
   Loaded: loaded (/etc/systemd/system/ups_test_service.service; enab>
   Active: active (running) since Wed 2020-05-20 18:08:22 EDT; 19s ago
 Main PID: 2136 (bash)
    Tasks: 2 (limit: 23896)
   Memory: 940.0K
   CGroup: /system.slice/ups_test_service.service
           ├─2136 /bin/bash /usr/bin/ups_test_service.sh
           └─2140 sleep 30

May 20 18:08:22 server.lab.example.com systemd[1]: Started UPS examp>
May 20 18:08:22 server.lab.example.com bash[2136]: Looping..
```

3. Checking **/var/log/messages** for Custom Service

Listing 15. Seaching Log file for Custom Service

```
[root@server ~]# grep -i ups /var/log/messages
May 20 18:08:22 jegui journal[2139]: This is a sample service started at 2020-05-20 18:08:22 for the UPS RH354 course.
```

SystemD References

Creating a Service at Boot: https://www.linode.com/docs/quick-answers/linux/start-service-at-boot/

Overview of SystemD for RHEL7: https://access.redhat.com/articles/754933

Converting traditional sysV init scripts to Red Hat Enterprise Linux 7 systemd unit files: https://www.redhat.com/en/blog/converting-traditional-sysv-init-scripts-red-hat-enterprise-linux-7-systemd-unit-files

Creating and Modifying SystemD Unit Files: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files

Creating a Linux service with systemd: https://medium.com/@benmorel/creating-a-linux-service-with-systemd-611b5c8b91d6

How to create systemd service unit in Linux: https://linuxconfig.org/how-to-create-systemd-service-unit-in-linux

1.4. FirewallD

Beginning in RHEL 8.0, Red Hat moved away from **iptables** as the back-end firewall implementation and instead moved to NFTables. However, the introduction of FirewallD and the **firewall-cmd** management commands implemented in RHEL 7.x have evolved and leveraging FirewallD is still the preferred firewall management solution in RHEL 8.x.

1.4.1. FirewallD Service Definitions

FirewallD was introduced in RHEL7 as part of the SystemD transition and a new way to manage firewalls without using the underlying firewall implementation (**iptables**). FirewallD with **firewall-cmd** continues to be used in RHEL8 as the preferred method of managing and maintaining firewall rules.

FirewallD Resources

https://firewalld.org/

https://www.liquidweb.com/kb/an-introduction-to-firewalld/

https://cheatography.com/mikael-leberre/cheat-sheets/firewall-cmd/

1.4.2. The firewall-cmd Utility

The **firewall-cmd** utility is the primary method to manage and interact with firewall rules on RHEL7/8 systems. The **firewall-cmd** utility supports BASH completion and allows firewall rules to be added based on defined services or by specifying ports/protocols.

Listing 16. Allowing HTTP through the Firewall by Port/Protocol

```
[root@server ~]# firewall-cmd --add-port=80/tcp
success

[root@server ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: cockpit dhcpv6-client ssh
  ports: 80/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

[root@server ~]# firewall-cmd --remove-port=80/tcp
success
```

Listing 17. Allowing HTTP through the Firewall by Service

```
[root@server ~]# firewall-cmd --add-service=
Display all 154 possibilities? (y or n)

[root@server ~]# firewall-cmd --add-service=http
success

[root@server ~]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp1s0
  sources:
  services: cockpit dhcpv6-client http ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

firewall-cmd *Usage Warning*

The **firewall-cmd** utility can be used to make changes to the running firewall as shown in the examples above. This does not make changes to the firewall config file. In order to make the changes to the configuration file, it is necessary to use the **--permanent** options to have the changes written to a file.

When using **--permanent**, and you are not making changes to the current firewall runtime, it is also necessary to use: **firewall-cmd --reload** to reload or load new firewall rules from the firewall configuration file.

Important Header

It is important to note that presently the Red Hat Web Console (cockpit) only supports management of **firewalld** using defined services.

1.4.3. FirewallD Files and Locations

FirewallD has a few locations for files both for configuration and usage. As with most configuration files on a Linux system, those rely in **/etc/**. The user configurable files for FirewallD also reside in **/etc/**. Default configuration files for services, zones, and other FirewallD functionality resides in **/usr/lib/firewalld**. This location contains all defined services files and default configuration files for FirewallD and used by the **firewall-cmd** utility.

Listing 18. FirewallD Configuration Files

```
[root@server ~]# tree /etc/firewalld/
/etc/firewalld/
├── firewalld.conf
├── helpers
├── icmptypes
├── ipsets
├── lockdown-whitelist.xml
├── services
└── zones
    ├── public.xml
    └── public.xml.old

5 directories, 4 files
```

Listing 19. FirewallD Default Configuration Files

```
[root@server ~]# tree /usr/lib/firewalld/
/usr/lib/firewalld/
├── helpers
│   ├── amanda.xml
│   ├── ftp.xml
│   ├── h323.xml
│   ├── irc.xml
│   ├── netbios-ns.xml
│   ├── pptp.xml
│   ├── proto-gre.xml
... output omitted ...
└── zones
    ├── block.xml
    ├── dmz.xml
    ├── drop.xml
    ├── external.xml
    ├── home.xml
    ├── internal.xml
    ├── public.xml
    ├── trusted.xml
    └── work.xml

5 directories, 222 files
```

1.4.4. Defining a Custom Service File

It is possible to define custom **firewalld** service files. These files can be used for your environments for custom applications or custom firewall rules. These service files can also be checked into a version control system such as **git**.

Creating a Custom Service File using Existing File as Base

It is easiest to take an existing service file, copy it to the **/etc/firewalld/services** directory, rename and edit the file.

1. Copy existing FirewallD service file to **/etc/firewalld/services**

Listing 20. Using SSH Service File as a Starting Point

```
[root@server ~]# cp /usr/lib/firewalld/services/ssh.xml /etc/firewalld/services/custom_ssh.xml
```

2. Edit the file and specify the new options

Listing 21. Edit the Custom SSH Service Definition

```
[root@server ~]# vim /etc/firewalld/services/custom_ssh.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>SSH_Custom</short>
  <description>This is a custom SSH service definition for paranoid people that don't want to run SSH on the default port of 22. This will
allow SSH to run on the port 8022 to meet our defined security guidance.</description>
  <port protocol="tcp" port="8022"/>
</service>
```

3. Listing FirewallD Services to Verify New Service

Listing 22. Getting Listing of FirewallD Services

```
[root@server ~]# firewall-cmd --get-services | grep custom_ssh
```

FirewallD Delays in Discovering a Service

Depending on system speed and refreshing of FirewallD daemon, the new service might not be picked up immediately. It will be discovered or if you are in a hurry, you can run **firewall-cmd --reload** command to immediately have the service discovered and available.

1.4.5. FirewallD Configuration Files

As stated above, the main FirewallD configuration files are located in **/etc/firewalld**. To specifically change or view the configuration file on the system, you generally want to look at the firewalls in the **Firewall Zone**. This is found by opening the corresponding zone file in **/etc/firewalld/zones** directory.

1. Viewing Firewall configuration for the Public Zone.

Listing 23. FirewallD Configuration for Public

```
[root@server ~]# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks to not harm your computer. Only selected incoming
connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
</zone>
```

2. Adding a custom service

Listing 24. Adding our new Service

```
[root@server ~]# firewall-cmd --add-service=custom_ssh --permanent
success
```

3. Verifying Firewall configuration for the Public Zone.

Listing 25. FirewallD Configuration for Public with Custom Service

```
[root@server ~]# cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas. You do not trust the other computers on networks to not harm your computer. Only selected incoming
connections are accepted.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <service name="cockpit"/>
  <service name="custom_ssh"/>
</zone>
```

1.5. Cockpit

Another change with RHEL 8.x was with the graphical server and the desktop manager. The X11 project and XWindows has been replaced largely with Wayland/Gnome3 as the graphical rendering environment of choice. These changes introduced some dependencies on the graphical management of several system services and components. The Wayland change no longer supports the X11 forwarding, so it was necessary to build tools or extend existing tools to be managed differently. The **cockpit** project was utilized and implemented in RHEL 8 as the new Red Hat Web Management Console which brought in numerous plugins allowing these services to be managed graphically within a standard web browser.

Red Hat Web Management Console

Leverage portions of the RH354 text around Cockpit and the Red Hat Web Management Console before looking at the additional Cockpit pacakges.

1.5.1. Installing Additional Cockpit Packages

Listing 26. Install all Base Cockpit Packages

```
[root@server ~]# yum install cockpit*
```

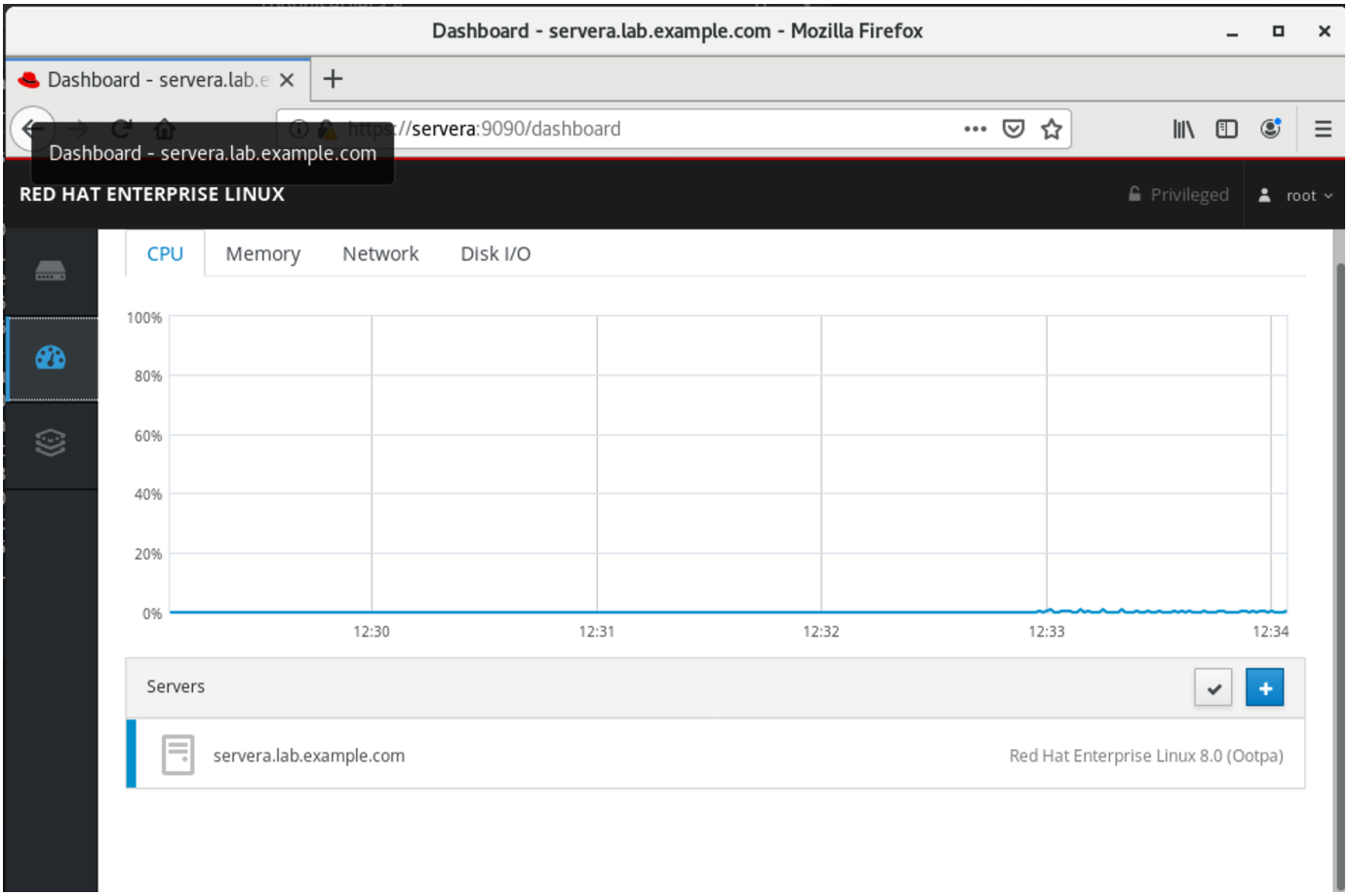


Figure 3. Cockpit Dashboard

Cockpit Plugin and Package Availability

Using the installation method above will install all Cockpit packages and plugins that are available in currently subscribed channels. However, not all components will work until additional back-end components are installed. For example, the **Composer** cockpit plugins need composer and other items installed in order to be able to be fully utilized. This installation gives the **Cockpit Dashboard Plugin** which allows connecting to multiple Web Consoles.

1.5.1.1. Cockpit to Manage Multiple Systems

It is possible to use a single **cockpit** Interface to manage multiple servers.

1. Ensure cockpit socket/service is running and configured on all systems.

Listing 27. Test and Enable Cockpit

```
[student@workstation ~]$ ssh root@server
Activate the web console with: systemctl enable --now cockpit.socket

[root@server ~]# systemctl enable --now cockpit.socket
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket → /usr/lib/systemd/system/cockpit.socket.
```

2. Connect to the Cockpit Web Console

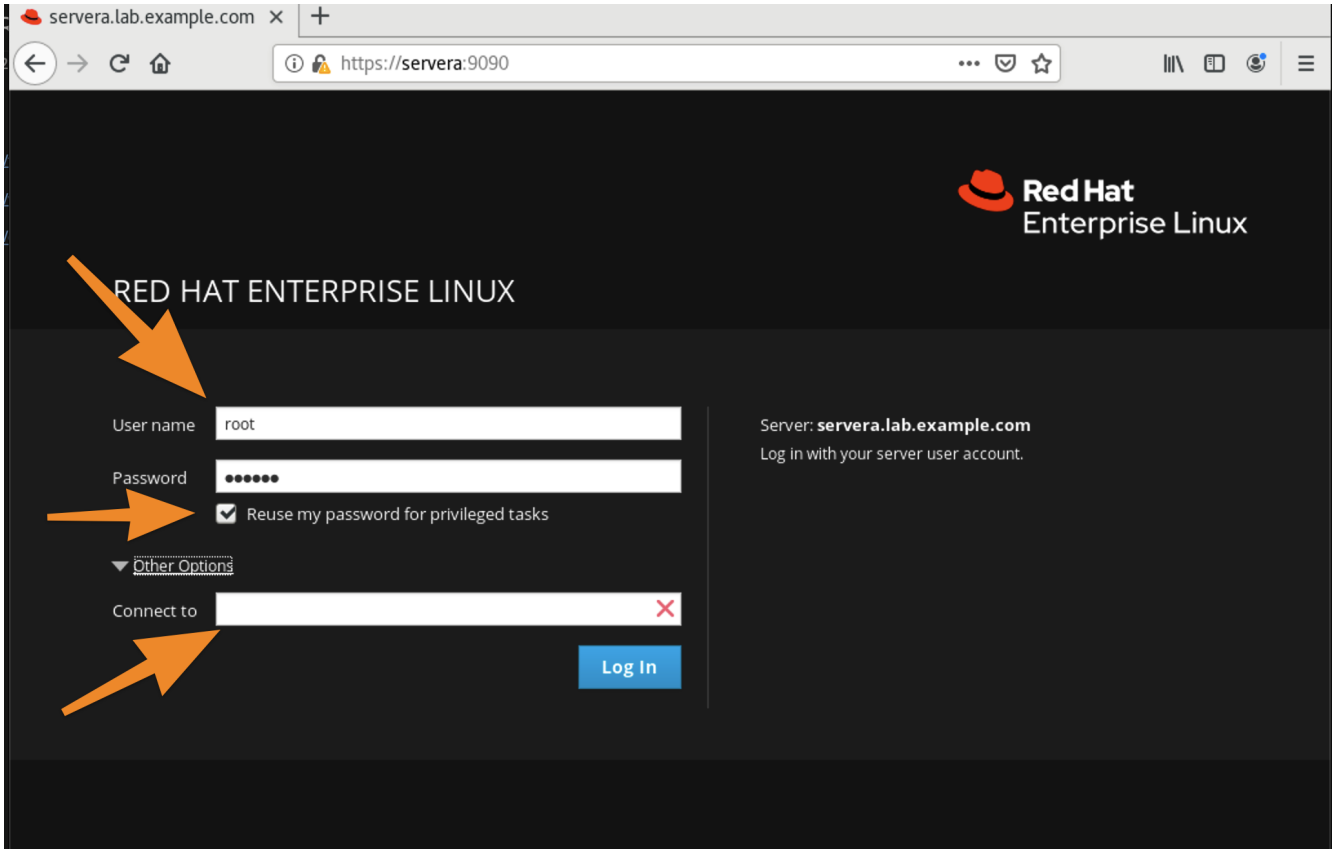


Figure 4. Red Hat Web Console (cockpit)

3. Add another Web Console from the Cockpit Dashboard
    a. Navigate to the Dashboard
    b. Click the "+" and complete the information

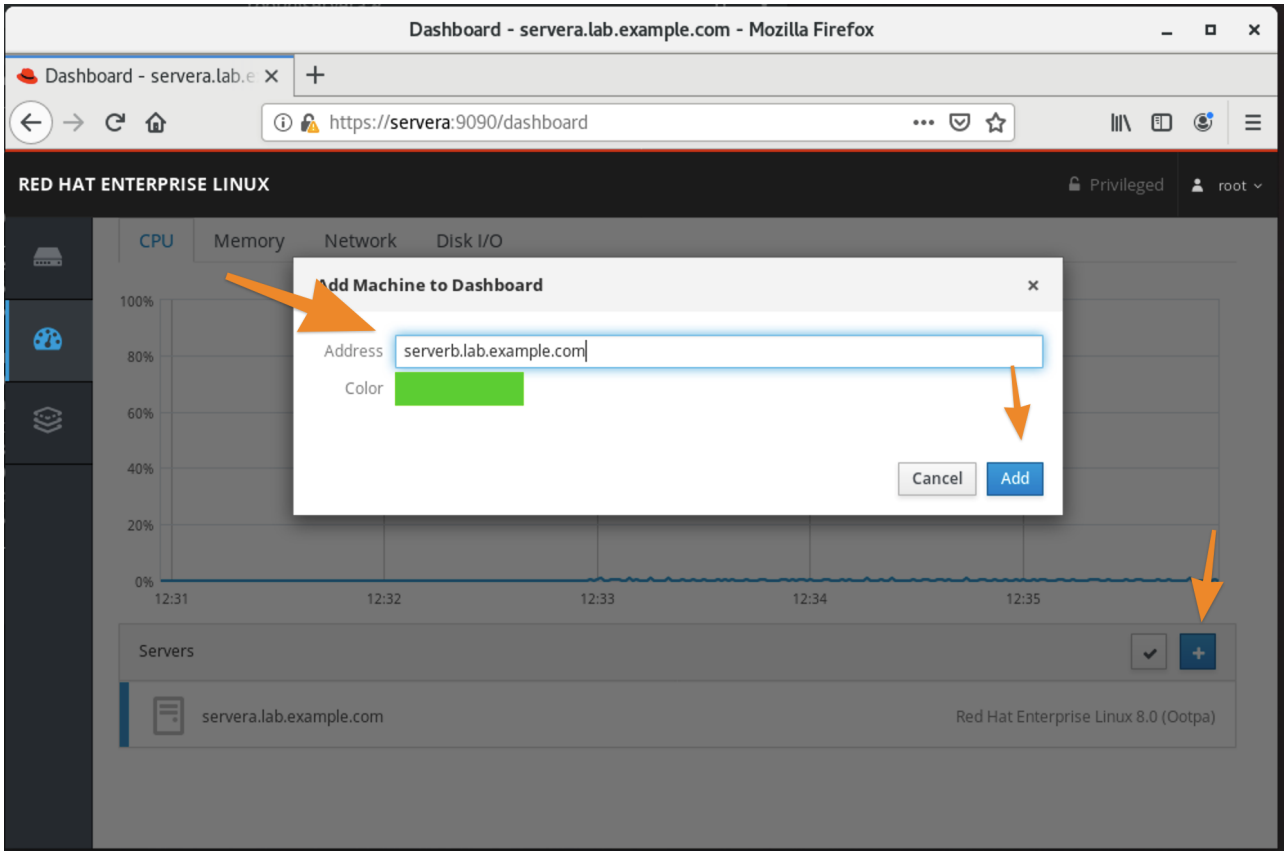


Figure 5. Adding Additional machines

4. Verify the System Fingerprint and click **Connect** image::Chapter3-87075.png[title="Fingerprint Verification", align="center"]
5. It is now possible to switch systems from the System Drop-down menu

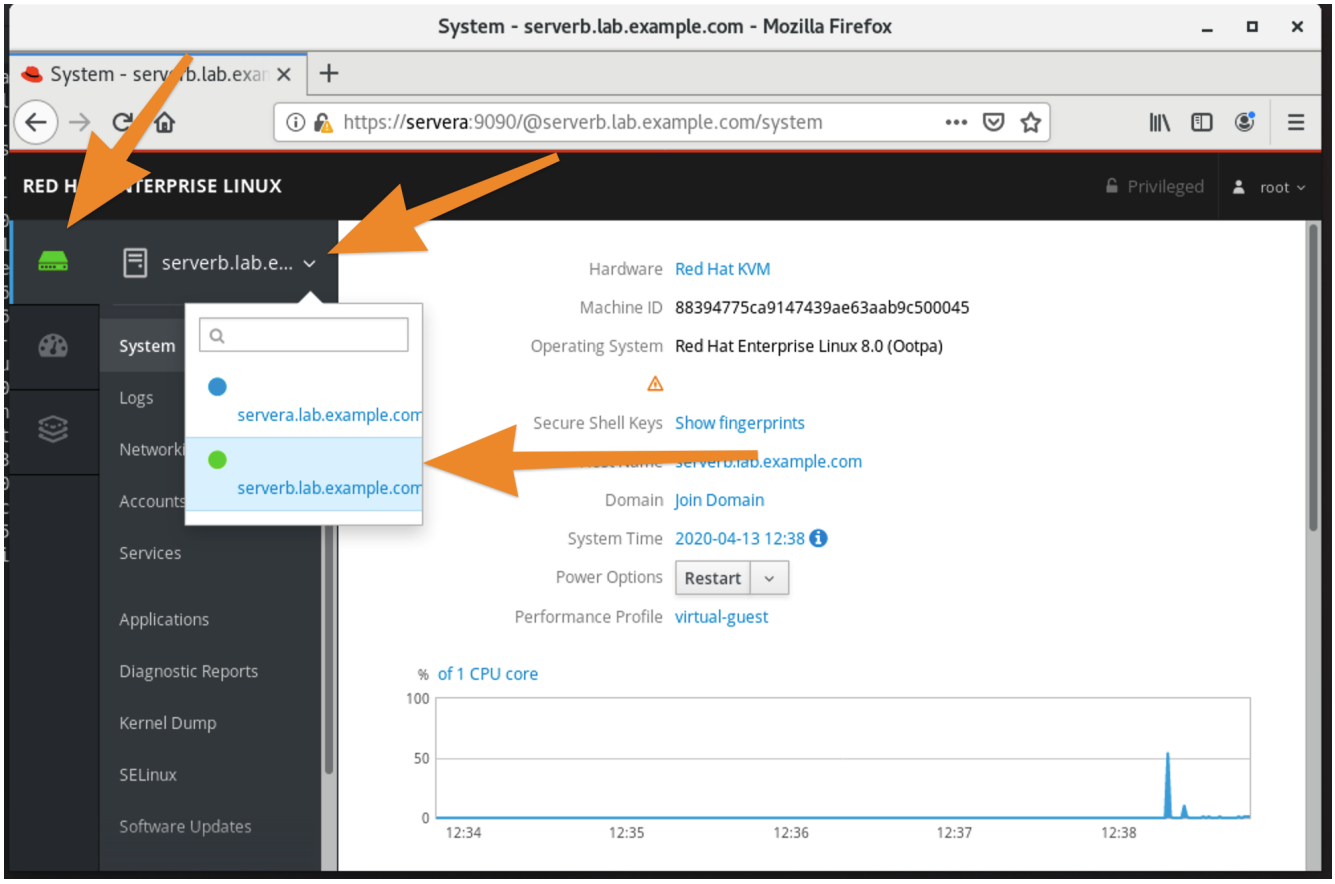


Figure 6. Switching Systems

2. Managing Containers with the New Runtime

2.1. Deploying Containers with the New Container Runtime

2.1.1. The Podman Container Engine

RHEL8 includes he **container-tools** package module. New engine is **podman** replaces **docker** and **moby**. It also contains new tools **buildah** to build container images and **skopeo** to manage images on registries like **runc**. The new toolset allows building/running containers without daemons.

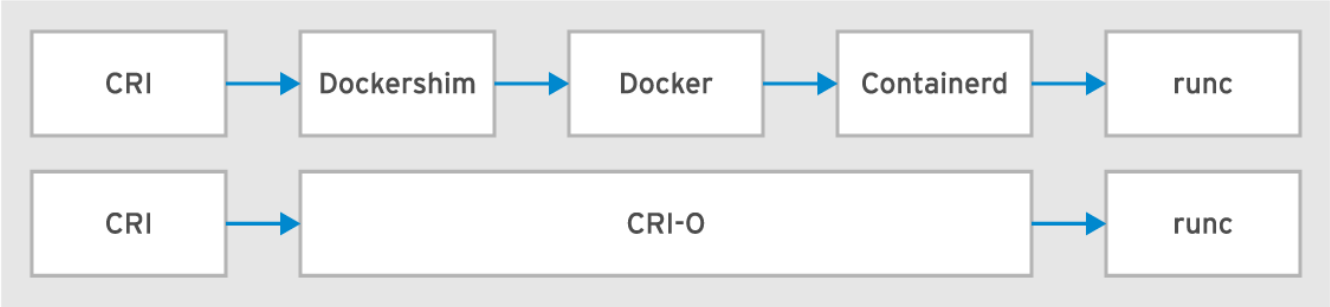


Figure 7. Docker to RHEL8 Container Runtime

Container Runtime Toolset

- Docker replaced with new container runtime
- New toolset supports OCI and reuse of third-party images
- Integrates with **audit** of Docker client-server model
- **container-tools** module provides new container runtime tools and engine.

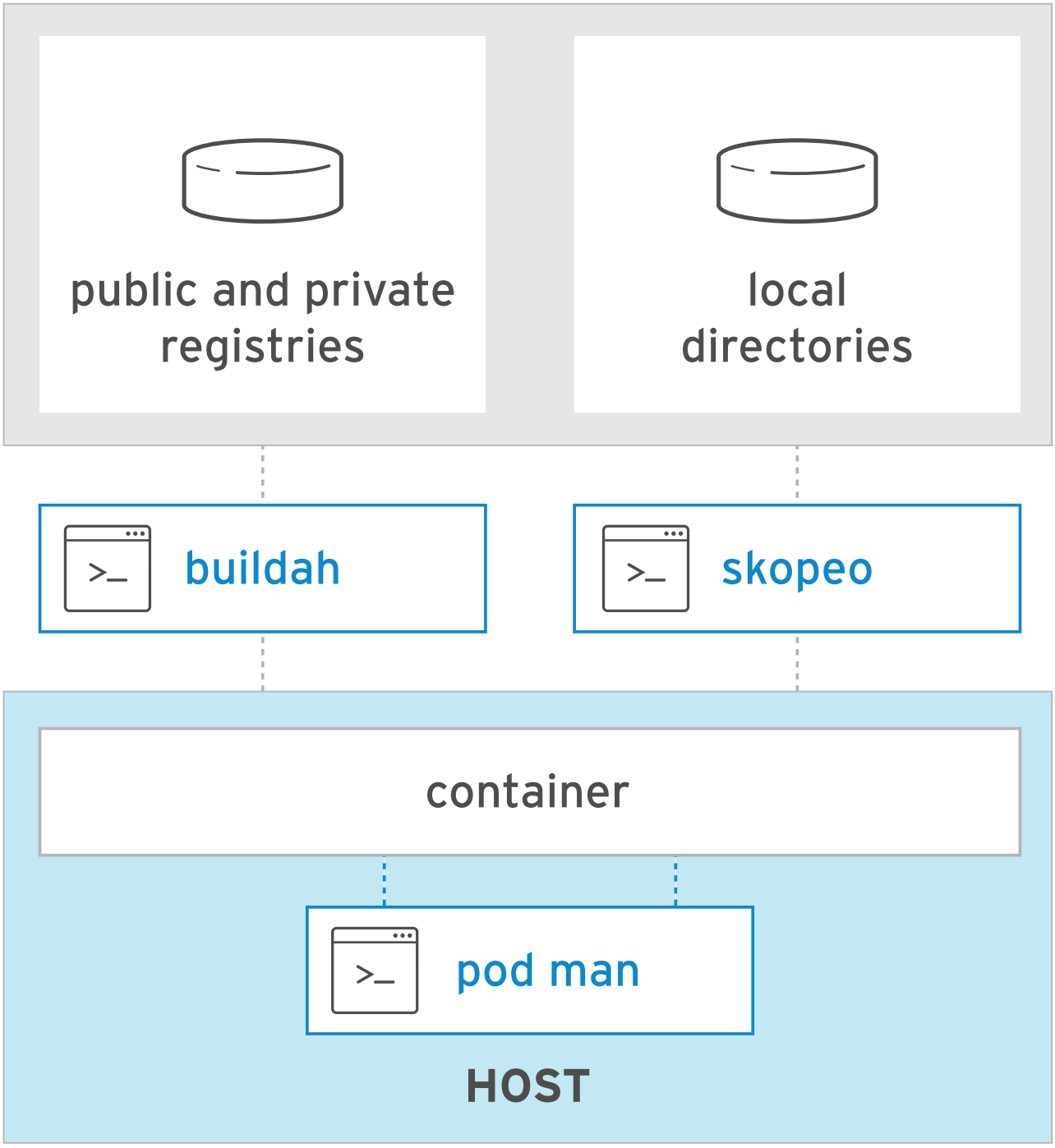


Figure 8. New Container Runtime

Describing new Container Runtime Tool

- The **podman** engine is daemonless and supporting container execution.
- **podman** syntax is similar to the docker command, supporting **Dockerfile** use
- **Buildah** builds container images, from scratch or a Dockerfile.
- Copy and inspect container images in registries with **Skopeo**
- **Skopeo** supports Docker and private registries, the Atomic registry, and local directories, including those which use OCI

RHEL8 includes **Pacemaker** containers with **podman** as a tech preview. Pacemaker supports execution of the container across multiple hosts.

Listing 28. Installation of Container Tools

```
[student@workstation ~]$ sudo yum module install container-tools
```

2.2. Podman Configuration Files

Depending on whether **podman** is being run as a privileged user or as a **rootless** user, the configuration file locations and directives will be difference.

2.2.1. Root-Based Podman

When running Podman as a privileged user, the default configurations are used. Generally these files are not modified, however, it is a common practice to modify the **registries.conf** file to add additional container image registries as well as alter the registry search order.

Default Configuration Files

- **/etc/containers/storage.conf** - Contains information about the default storage for Podman containers when using **podman** in a non-rootless fashion.
- **/etc/containers/registries.conf** - Contains information about the registries used for **podman** when doing **podman search** and also for pulling/running of container images.
- **/etc/cni/net.d/87-podman-bridge.conflist** - Contains information related to the CNI (container network interface) and IP configuration for the container networks.

2.2.2. Rootless Podman

When running **podman** in **rootless** mode, all configuration files will be based on user configurations and are located within the user's home directory.

User-Based Podman Configuration Files

- **/home/student/.config/containers/storage.conf**: Specifies storage configuration for Podman in **rootless** mode.
- **/home/student/.config/containers/registries.conf**: Specifies registry configuration for Podman in **rootless** mode.

Podman User Configuration Files

If the configuration files don't exist in the user's home directory, **podman** will default back up to **/etc** for the configuration files and if they aren't there, it will fall back to the **/usr/share/containers/** directory.

Podman Configuration File Precedence

podman has multiple configuration file locations and the order of precedence depends on the if the files exist. Traditionally in a **rootless** implementation, the configuration files with be in the user's **$HOME/.config/containers** directory. In the case of networking, there is no CNI equivalent for **rootless** Podman as **rootless** containers rely on **SLIRP** for networking.

Containers.conf

1. **$HOME/.config/containers/containers.conf**
2. **/etc/containers/containers.conf**
3. **/usr/share/containers/containers.conf**

Storage Configurations

1. **$HOME/.config/containers/storage.conf**
2. **/etc/containers/storage.conf**

Registry Configurations

1. **HOME/.config/containers/registries.conf**
2. **/etc/containers/registries.d/**
3. **/etc/containers/registries.conf**

Precedence of **1** is the highest and will override all others.

References

- **Podman Project - Config Files and Tutorial**: https://github.com/containers/podman/blob/main/docs/tutorials/rootless_tutorial.md
- **Why can't I use sudo with rootless Podman?**: https://www.redhat.com/sysadmin/sudo-rootless-podman
- **What happens behind the scenes of a rootless Podman container?**: https://www.redhat.com/sysadmin/behind-scenes-podman
- **Rootless containers using Podman**: https://www.redhat.com/sysadmin/rootless-containers-podman
- **Container video series: Rootless containers, process separation, and OpenSCAP**: https://www.redhat.com/sysadmin/container-video-series (Really good by Brian Smith - Former RH TAM, now works in the RHEL Platform BU)
- **Why can't rootless Podman pull my image?**: https://www.redhat.com/sysadmin/rootless-podman

2.3. Container Image Storage

Containers use ephemeral storage for running containers which is an overlay filesystem with a new read/write (RW) layer added to the original container image. This ephemeral storage is removed once the container is removed from the system. Container images on the other-hand are stored locally on the system in the container image storage registry. It is important to know where the image storage is located for both container images and ephemeral storage for running containers so that it can be monitored for pro-active system administration and cleanup. The storage location for both container images and ephemeral storage is generally defined in the **storage.conf** file.

2.3.1. Root-Based Podman

A default installation of Red Hat Container Tools (podman) will create a configuration file for storage located **/etc/containers/storage.conf**. These are the most likely settings to be used when running Podman as a **root** user.

Listing 29. Default Storage Location

```
[storage]

# Default Storage Driver
driver = "overlay"

# Temporary storage location
runroot = "/var/run/containers/storage"

# Primary Read/Write location of container storage
graphroot = "/var/lib/containers/storage"
```

There are plenty of options regarding containers and image storage, but the primary locations to monitor are **/var/run/containers/storage** and **/var/lib/containers/storage** as these will be the most used locations by **podman**.

2.3.2. Rootless Podman

Podman uses the storage configuration file located **$HOME/.config/containers/storage.conf**. These settings are used because the **overlay** filesystem must use a user-space filesystem overlay so it uses **FUSEFS** storage drives for the user-space filesystems.

Listing 30. Default Storage Location for Rootless Podman

```
[storage]
  driver = "overlay"
  runroot = "/run/user/1000"
  graphroot = "/home/student/.local/share/containers/storage" ①
  [storage.options]

... OUTPUT OMITTED ...

    mount_program = "/usr/bin/fuse-overlayfs" ②
```

① Image storage location

② FUSEFS Overlay Filesystem Driver

There are plenty of options regarding containers and image storage, but the primary locations to monitor are

/var/run/containers/storage and **/var/lib/containers/storage** as these will be the most used locations by **podman**.

2.4. Container Networking

Podman is meant to provide all management of containers and the container runtime. Podman is capable of managing the container network (SDN) for root-based Podman containers. The CNI controls the specifications for networking and how the SDN is defined on the system. There is no SDN available for **Rootless** containers as **podman** implements networking for **Rootless** containers using **SLIRP**.

2.4.1. Root-Based Podman

Podman root-level containers can leverage CNI. The containers SDN network is defined in the **/etc/cni/net.d/87-podman-bridge.conflist** configuration file. Containers can communicate to each other within the SDN on the same system.

Listing 31. **/etc/cni/net.d/87-podman-bridge.conflist**

```
{
    "cniVersion": "0.4.0",
    "name": "podman",
    "plugins": [
    {
            "type": "bridge", ①
            "bridge": "cni-podman0", ②
            "isGateway": true,
            "ipMasq": true,
            "ipam": {
        "type": "host-local",
        "routes": [
            {
            "dst": "0.0.0.0/0"
            }
        ],
        "ranges": [ ③
            [
            {
                "subnet": "10.88.0.0/16",
                "gateway": "10.88.0.1"
        }
            ]
        ]
              }
    },
    {
              "type": "portmap",
              "capabilities": {
        "portMappings": true
              }
    },
    {
              "type": "firewall"
    }
      ]
  }
```

① Defines network type as a **Bridge**

② Defines the network name for the bridge as **cni-podman0** for the container SDN

③ Defines the network IP Address range for the container SDN

2.4.2. Rootless Podman

For **rootless** podman the networking for containers leverages **SLIRP**. This is provided by the **slirp4netns** package and provides user-mode networking and namespaces for the networks. Containers running as a **Rootless** container will not receive an IP address from the CNI SDN and cannot communicate with each other or the outside except by using and leveraging **port forwarding**.

Red Hat Container Catalog

Red Hat Container Catalog: https://catalog.redhat.com/software/containers/search

2.5. Managing Containers using the Red Hat Web Console

The Red Hat Web Management Console (Cockpit) can be used to manage container images as well as running containers. However, in order to manage some of the aspects around containers, certain configurations must already be in place.

Cockpit can almost manage the full container lifecycle. using the **Podman containers** plugin. Container images can be downloaded, managed, deleted. Containers can be launched from container images, stopped, restarted. Containers can even be analyzed providing runtime information, container logs, and even console access. Existing containers can also be turned into new container images using the **Commit** functions.

Container Runtime Details

The **Details** tab can provide container runtime information. This is similar to running the **podman inspect** without providing all the details.

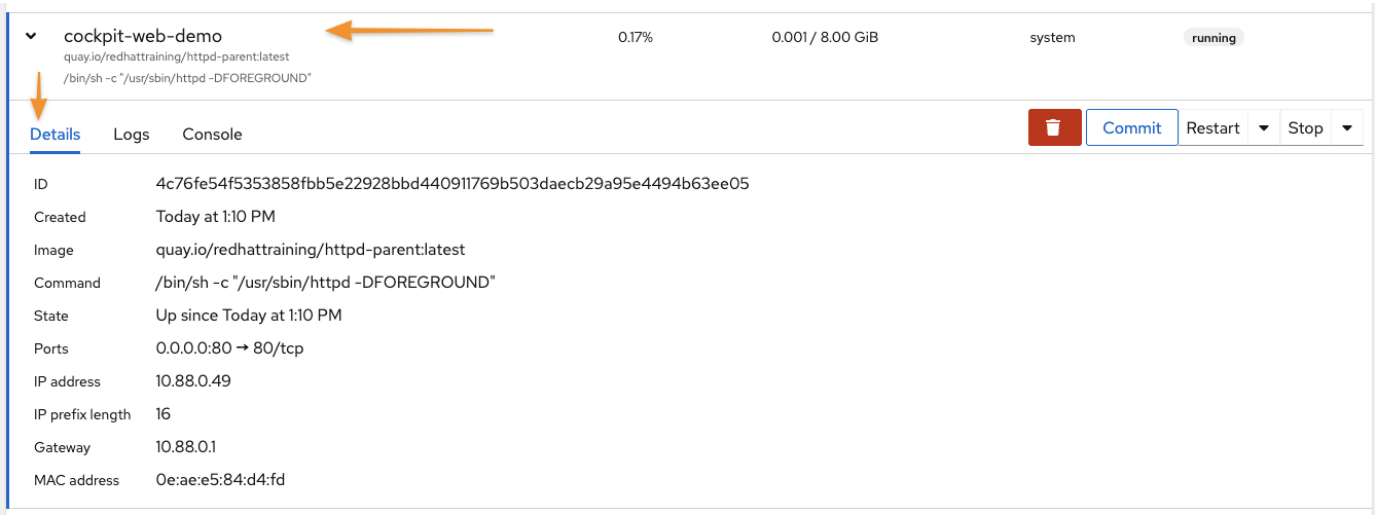


Figure 9. Container Runtime Information

Container Logs

The **Logs** tab is capable of sharing the container logs. This is similar to running the **podman logs** command.

Figure 10. Container Logs

Container Console

Cockpit also allows accessing a console directly inside the running container. Accessing the console via cockpit is the same as running the **podman exec -it <Container> /bin/bash** command.

+ image::Chapter2-416de.png[title="Container Console", align="center"]

2.5.1. Installing Cockpit Podman Plugins

In order to utilize Cockpit to manage containers with Podman there must be two things that are in place.

Cockpit **podman** *Requirements*

- Podman cockpit plugins
- Podman service started and running on the system

1. Install **cockpit-podman** Modules

Listing 32. Installation of Cockpit **podman** *Plugins*

```
[root@servera ~]# yum install cockpit-podman

... OUTPUT OMITTED ...

Complete!
[root@servera ~]#
```

2. Enable **podman** Service in Cockpit

   a. Sign into Cockpit

   b. Click "**Podman containers**"

c. Click "**Start podman**"

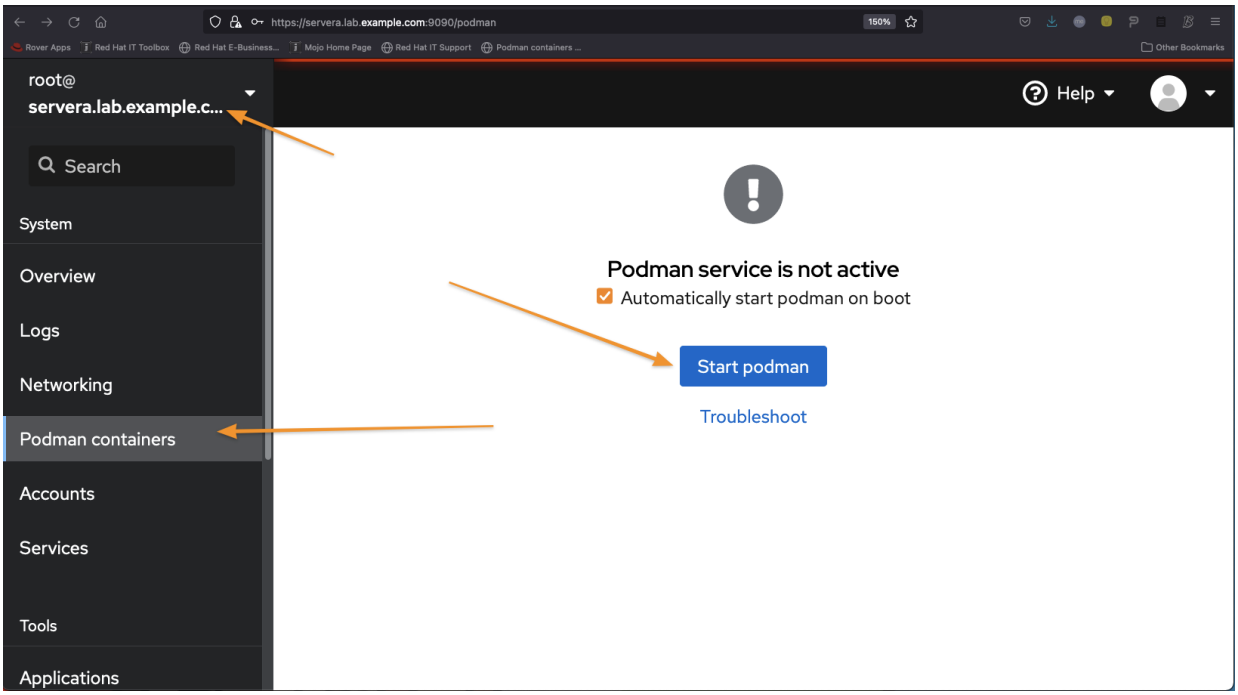


Figure 11. Cockpit - Podman Containers (Setup)

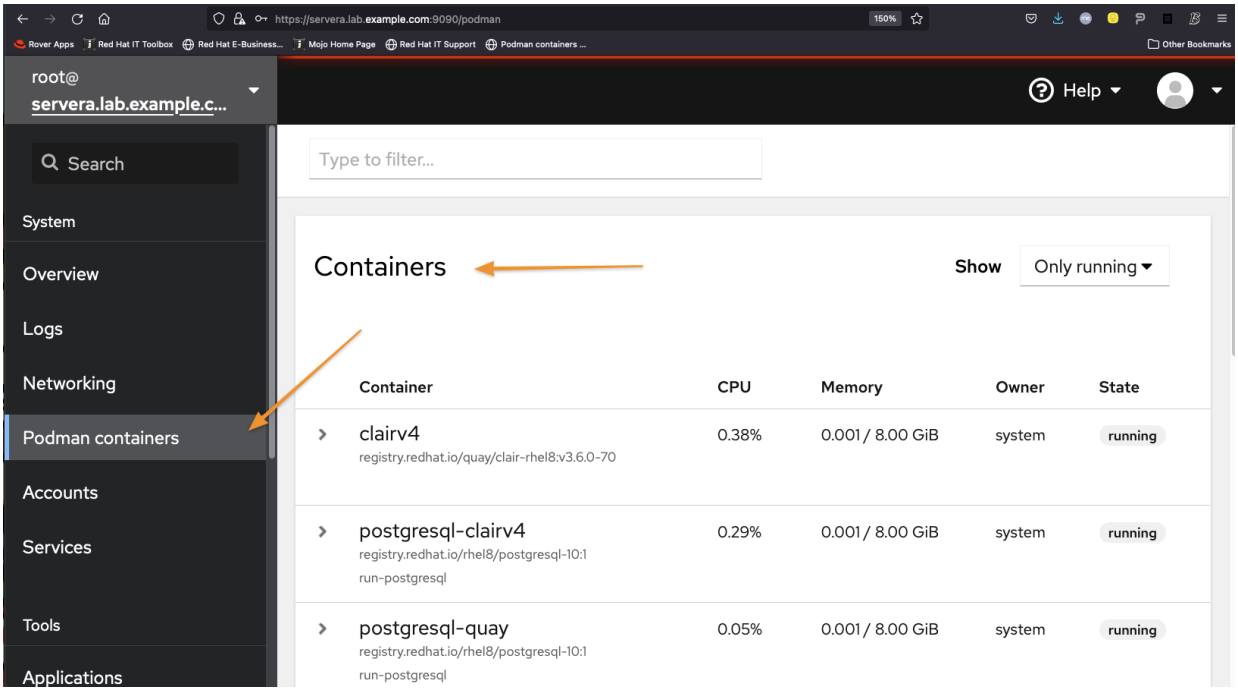


Figure 12. Cockpit - Podman Containers (In-Use)

Ensure Cockpit is both Installed and Enabled

Listing 33. Installing Cockpit

```
[root@servera ~]# yum install cockpit

... OUTPUT OMITTED ...

subscription-manager-cockpit-1.28.13-2.el8.noarch
tracer-common-0.7.5-2.el8.noarch

Complete!
```

Listing 34. Enabling Cockpit Socket

```
[root@servera ~]# systemctl enable cockpit.socket --now
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket →
/usr/lib/systemd/system/cockpit.socket
```

2.5.2. Using Cockpit to Manage Containers

Once installed and enabled, the **Podman containers** plugin can begin managing containers and images. There are a few things to remember and consider as the **plugin** can't do everything.

Plugin Considerations and Limitations

- **Registries**: Image registries must be setup and specified ahead of time in the **/etc/containers/registries.conf** file.
  - Image registries must be in the configuration file in order for images to be downloaded and used. If the configuration file is updated after the **podman** service has been started, the service must be "restarted" in order for Cockpit to see the changes.
  - There are some difficulties with providing authentication information to registries as there isn't currently an interface to login to the registry within Cockpit.
- **Storage Volumes**: It might be necessary to have volumes and permissions setup to properly mount/map the volumes into the running containers

Registry Credentials

In order for Podman to authenticate, you must provide credentials. There needs to be a config file and you may also need to login to **podman** from the command line.

Listing 35. ***.docker/config.json*** *Config File*

```
[student@servera ~]$ cat .docker/config.json
{
  "ServerURL": "https://registry.redhat.io/v1",
  "Username": "RHNID",
  "Secret": "RHNPassword"
}
```

Listing 36. CLI Login

```
[student@servera ~]$ podman login registry.redhat.io
Authenticating with existing credentials...
Existing credentials are valid. Already logged in to registry.redhat.io
```

Container Image Management with Cockpit

1. Navigate to the **Images** section and select **Get new image**

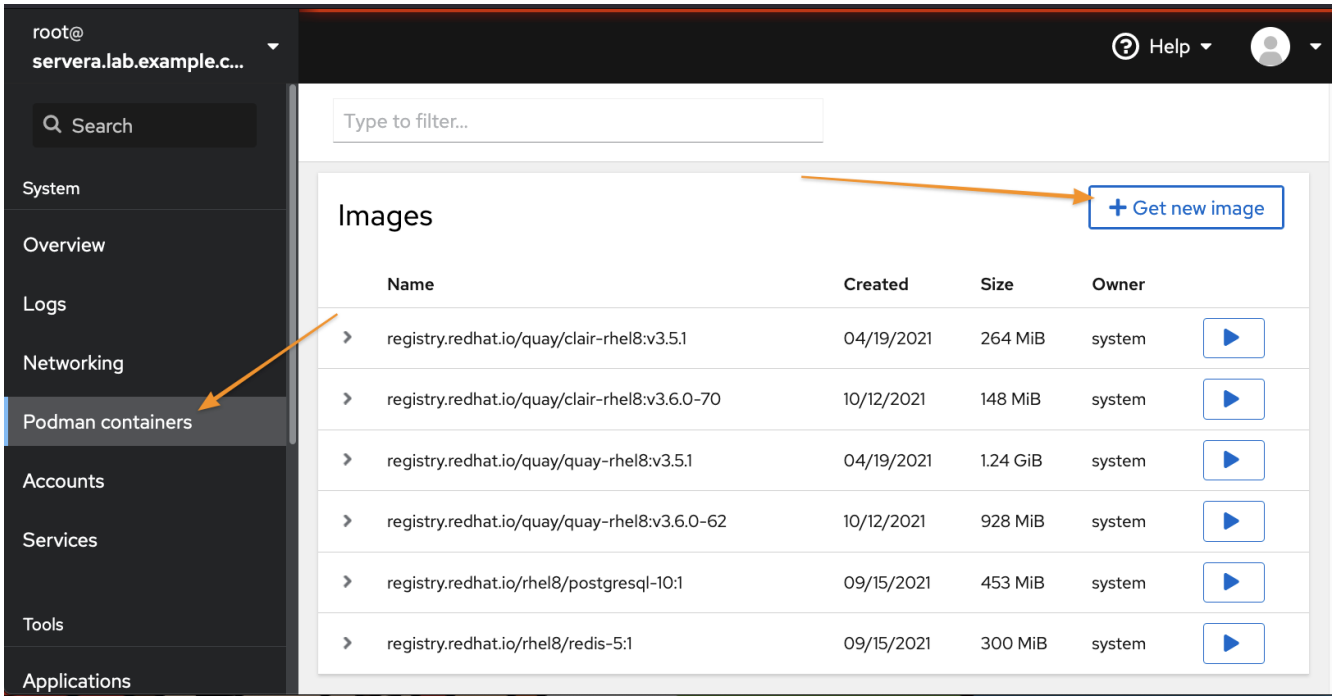


Figure 13. Cockpit - Podman Container Images

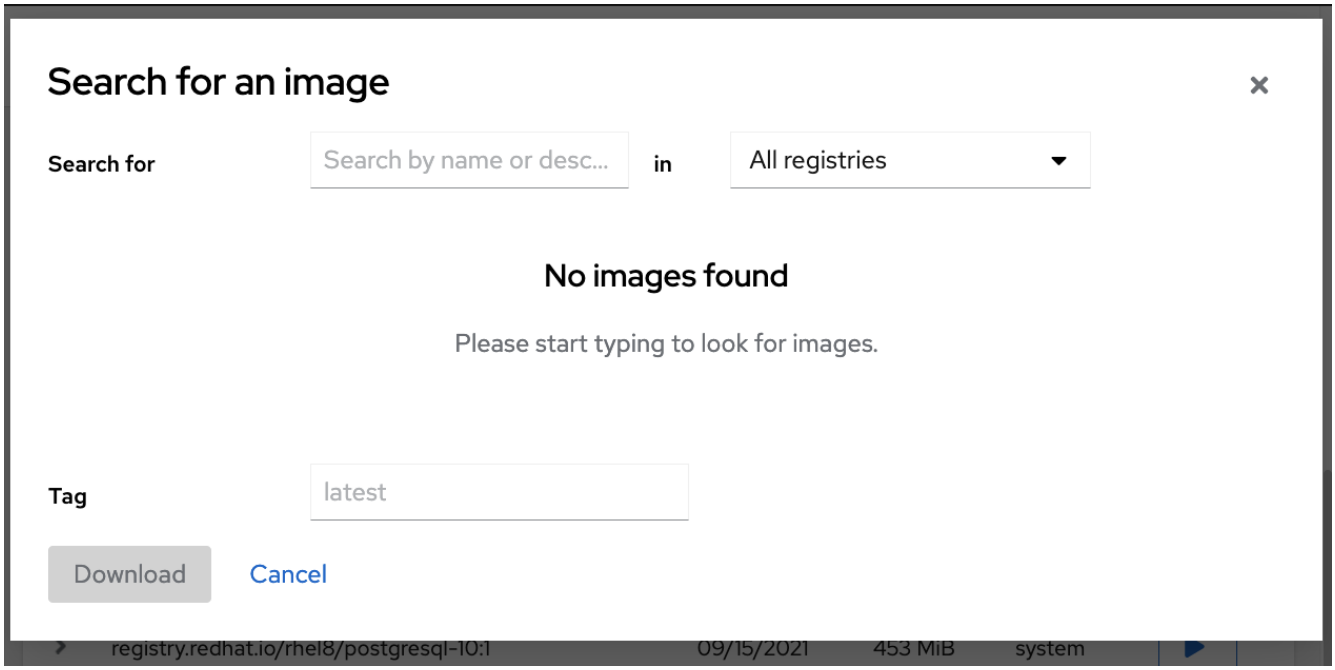


Figure 14. Search for Images

2. Select the registry to search, tags, and image name/description, then click "Download"
    a. Search for the HTTPD-Parent image in the **quay.io** registry.

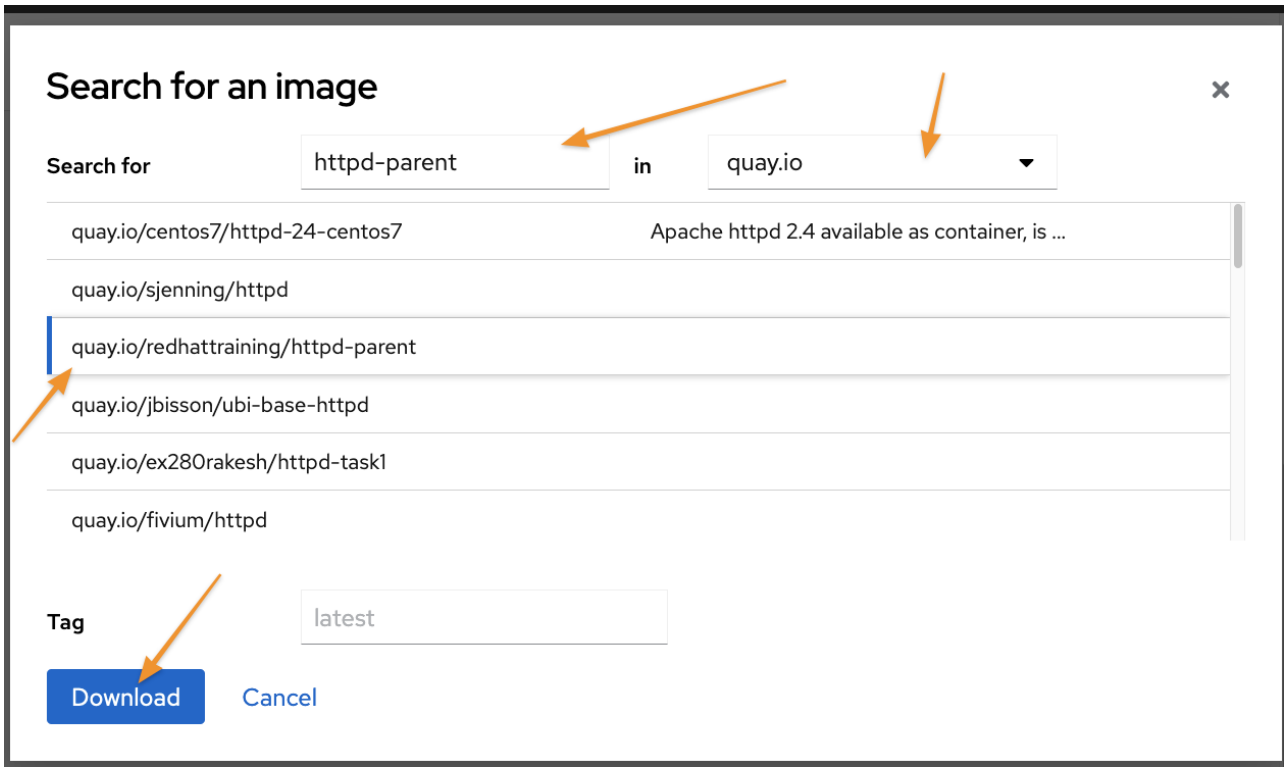


Figure 15. HTTPD 2.4 Parent Image from Red Hat Training - Quay.io

Adding the **quay.io** *Registry to* **/etc/containers/registries.conf**

Update the **registries.conf** file and then restart the **podman** service.

Listing 37. Edit **/etc/containers/registries.conf**

```
# To ensure compatibility with docker we've included docker.io in the default search list. However Red Hat
# does not curate, patch or maintain container images from the docker.io registry.
[registries.search]
registries = ['registry.access.redhat.com', 'registry.redhat.io', 'docker.io', 'quay.io']①
```

① Adding **quay.io**

Listing 38. Restart **podman**

```
[root@servera ~]# systemctl restart podman.service
```

Registry Authentication

It is important to keep in mind, images requiring authentication to a registry will provide errors in Cockpit as you will be accessing the registry as an unauthenticated user.

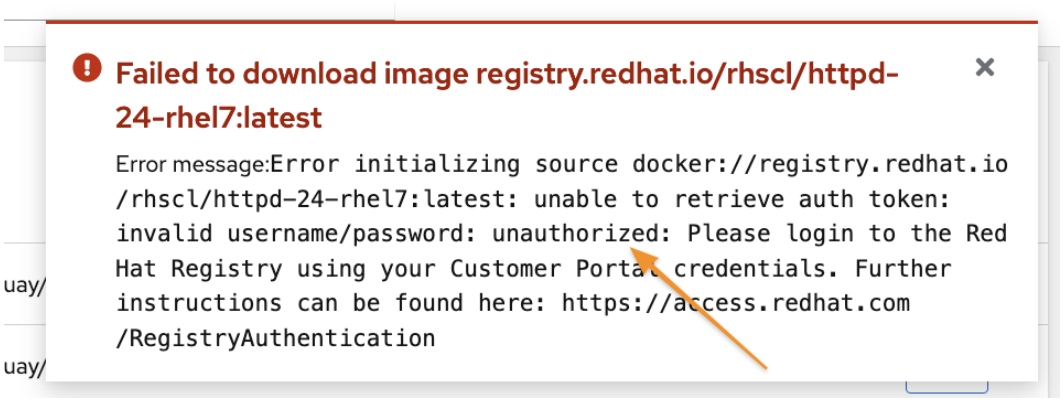


Figure 16. Registry Authentication Issues

Based on timeline and platform difficulties, images downloaded in the Red Hat Web Management Console will be coming from public (unauthenticated) registries.

3. Confirm image was downloaded.

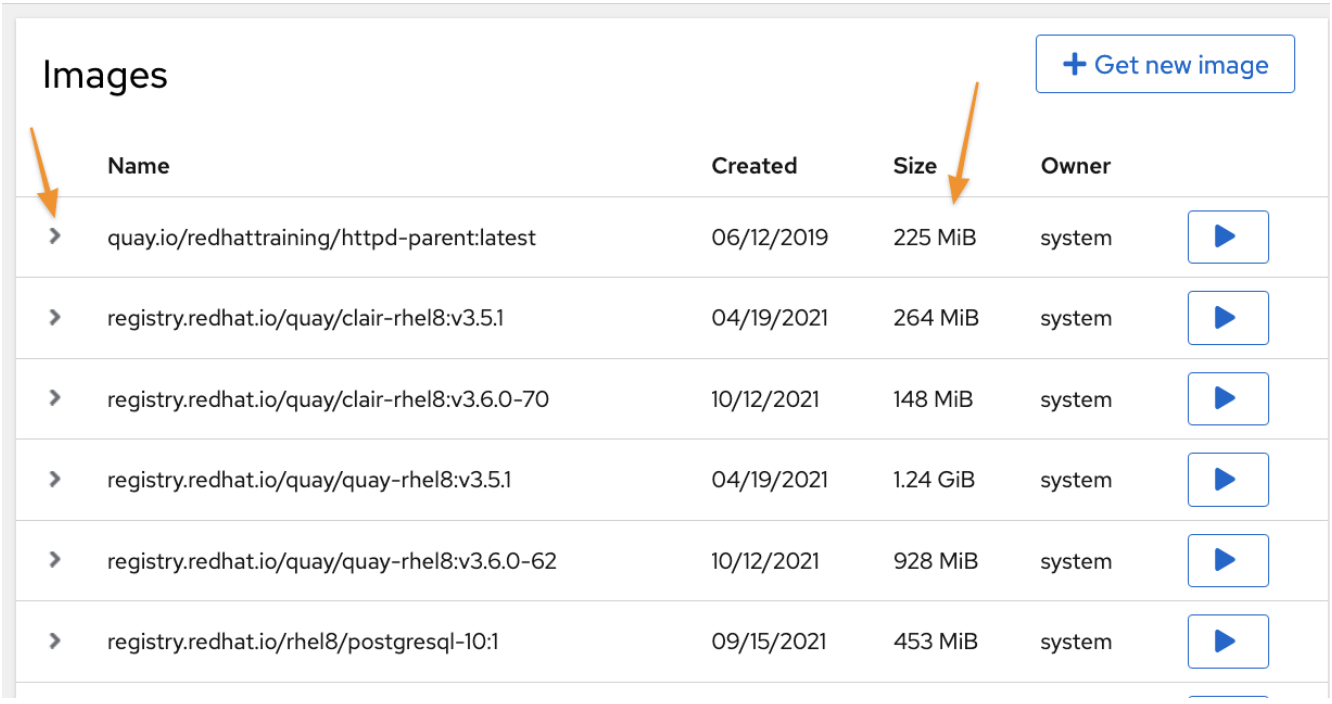


Figure 17. Images in Local Registry

4. Explore image details by clicking the **>** and expanding the available information.

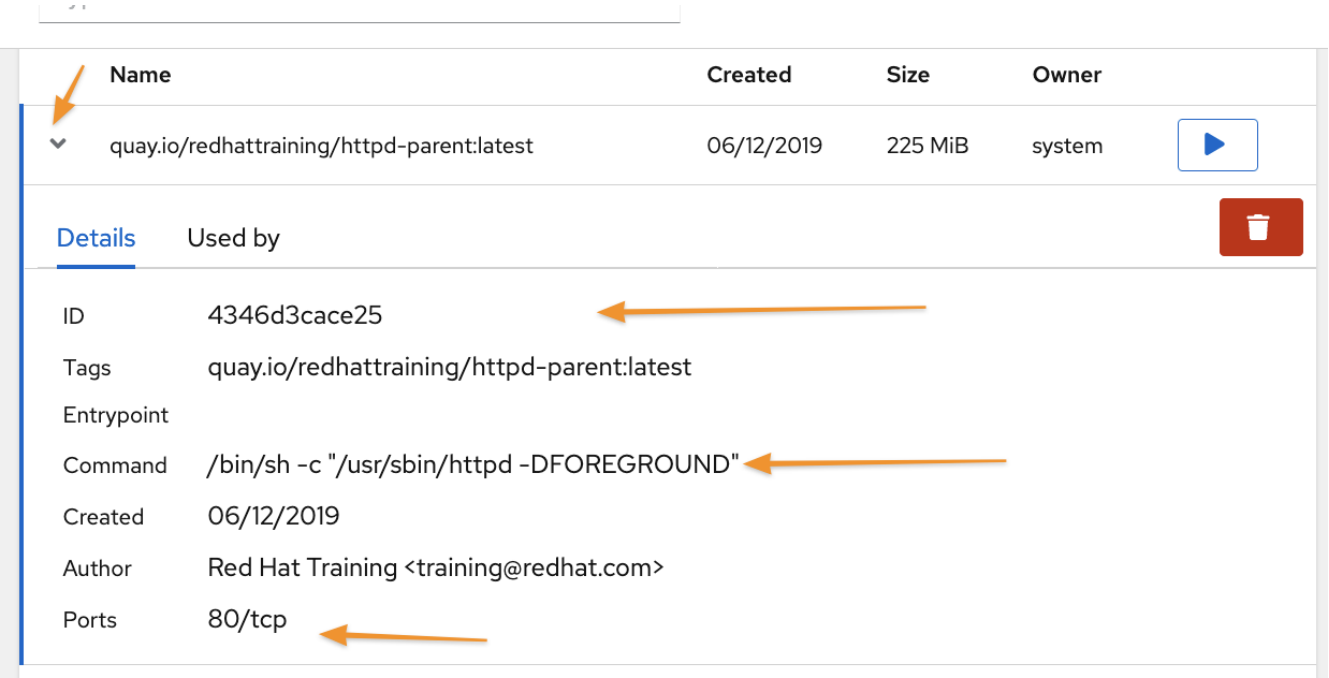


Figure 18. HTTPD-Parent Image Details - Image Information

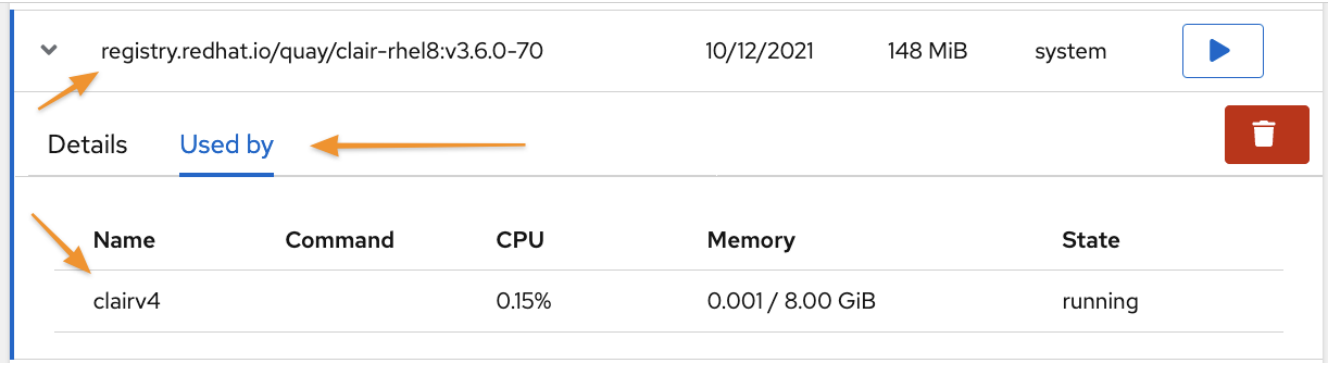


Figure 19. Clair Image Details - Containers Using Image

Container Management with Cockpit

It is possible to manage both **rootless** and **root-based** containers in Cockpit depending on the user you are using to access the management console. In order to create and launch new containers from Cockpit, the image must already be downloaded and existing in the local image registry.

Container Creation

1. Select the image to be used to launch the container by clicking the **Play** button.

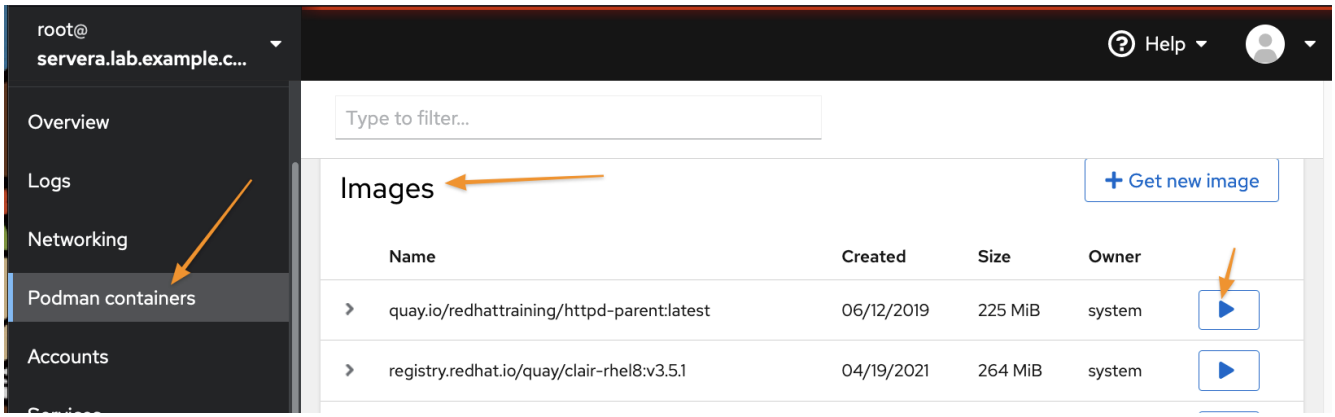


Figure 20. HTTPD-Parent Image Selection

2. Enter details you wish to use for your image and click "Run"
    a. Container Name
    b. Port mapping
    c. Volume mapping

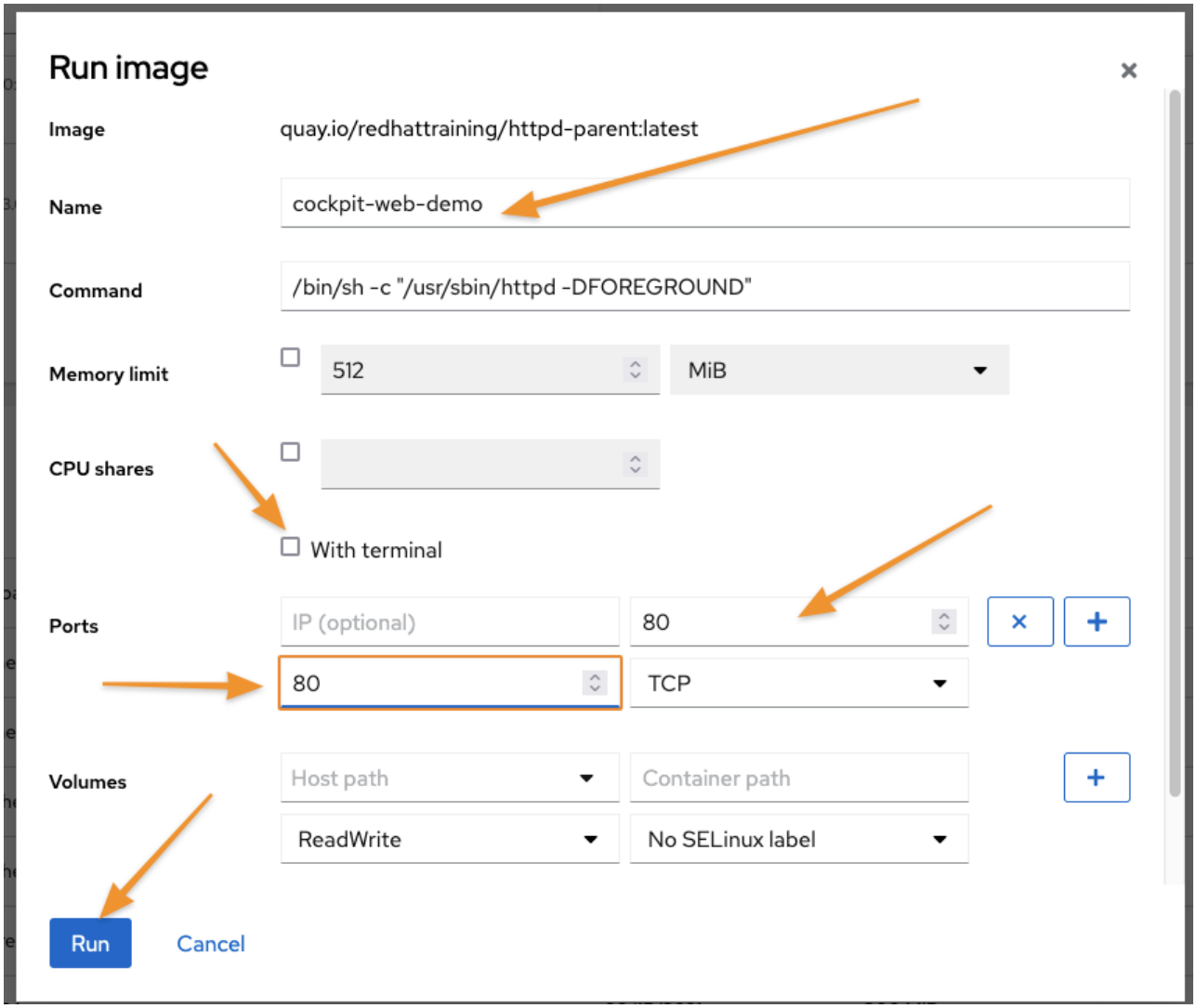


Figure 21. Launcing a Demo Container

NOTE *Regarding* **With terminal**

For several containers, you should **uncheck** the ***With terminal*** option as this could cause the container to exit unexpectedly.

3. Verify container was launched

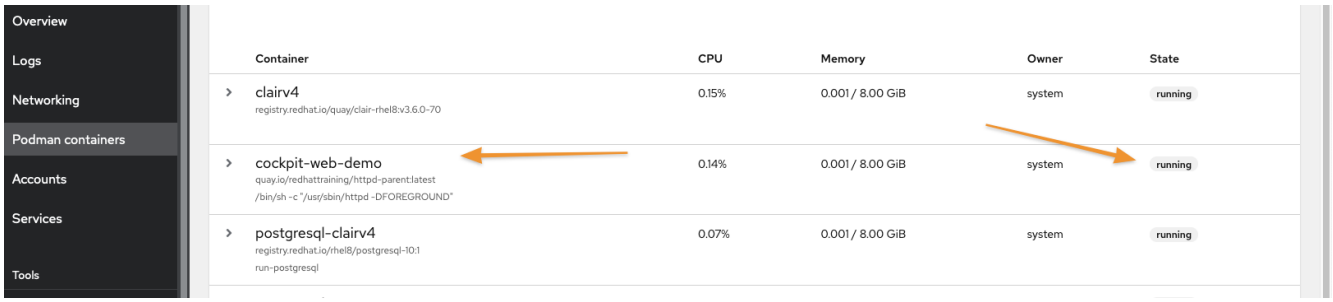


Figure 22. Running Container Verification

Figure 23. Running Container Details

4. Allow connection to container externally through Firewall
   a. Click **Networking**
   b. Click **Edit rules and zones** for the Firewall

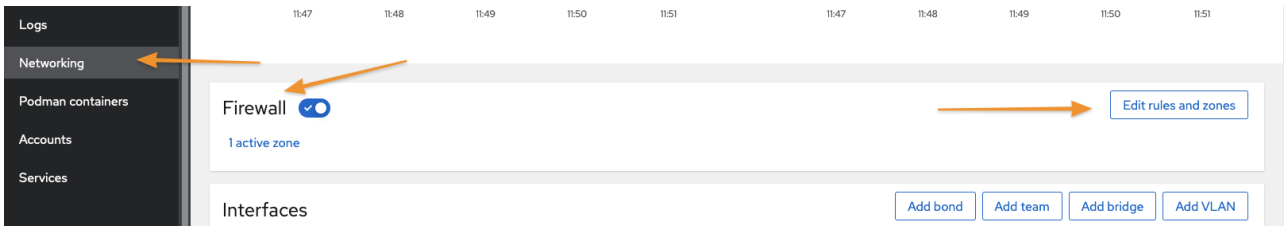


Figure 24. Using Cockpit to Manage Firewalls

5. Click **Add Services** and either select an existing service or create custom ports, then click the **Add Services** button on the **Add services to public zone**.
   a. Verify the service was added

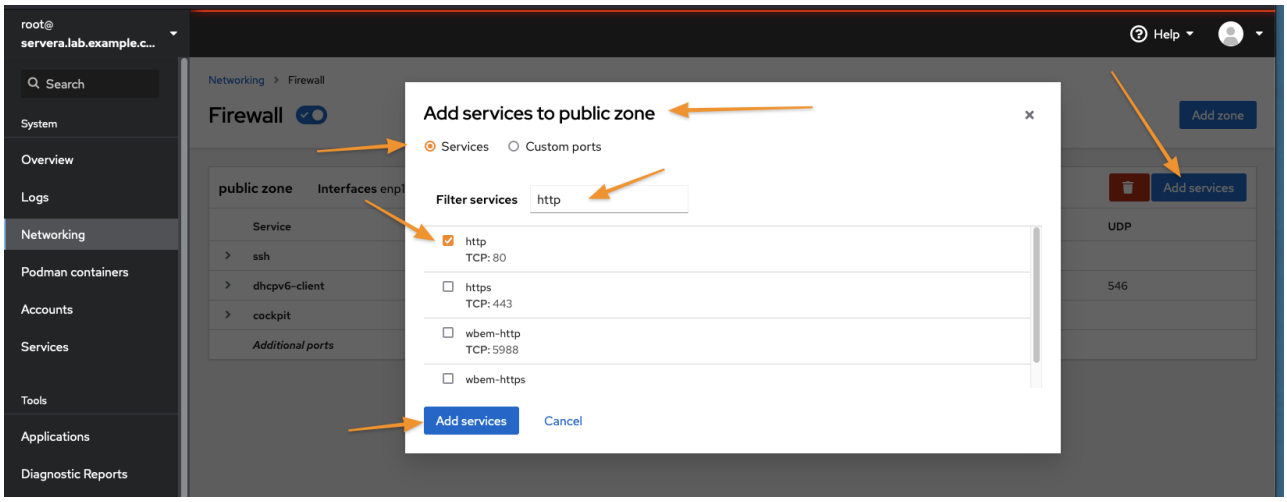


Figure 25. Add the HTTP Service

Figure 26. HTTP Service Available

6. Verify the container is running and hosting the website via the console and through a web browser.

Listing 39. Local Container Verification

```
sh-4.4# curl localhost
Hello from the httpd-parent container!
sh-4.4#
```

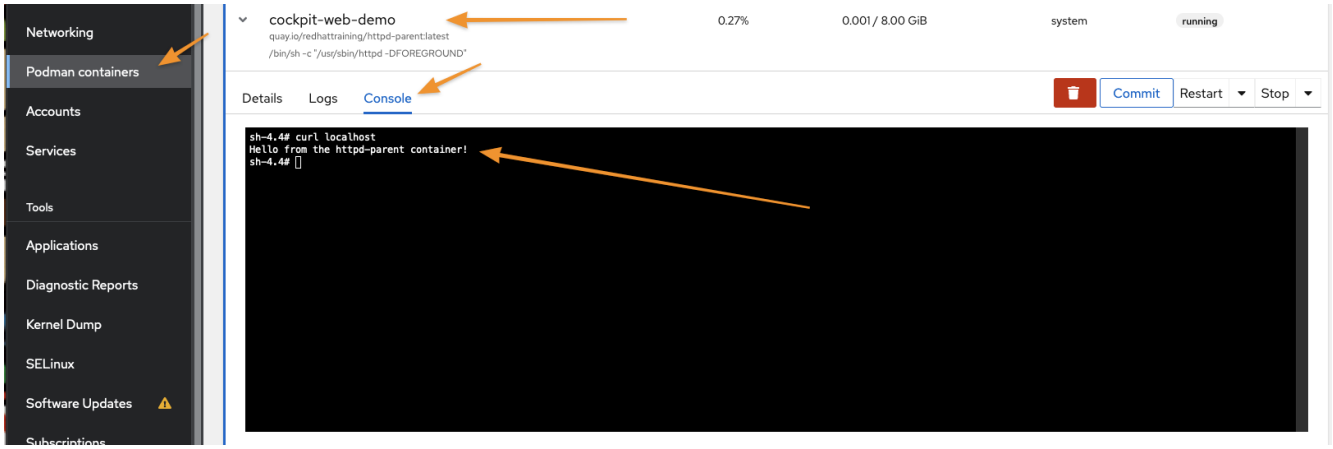


Figure 27. Container Console in Cockpit

Figure 28. Application Verification in Firefox

Saving a Container Image from a Runnign Container

1. Select the container to use as a base for the image
   a. Click **Commit**

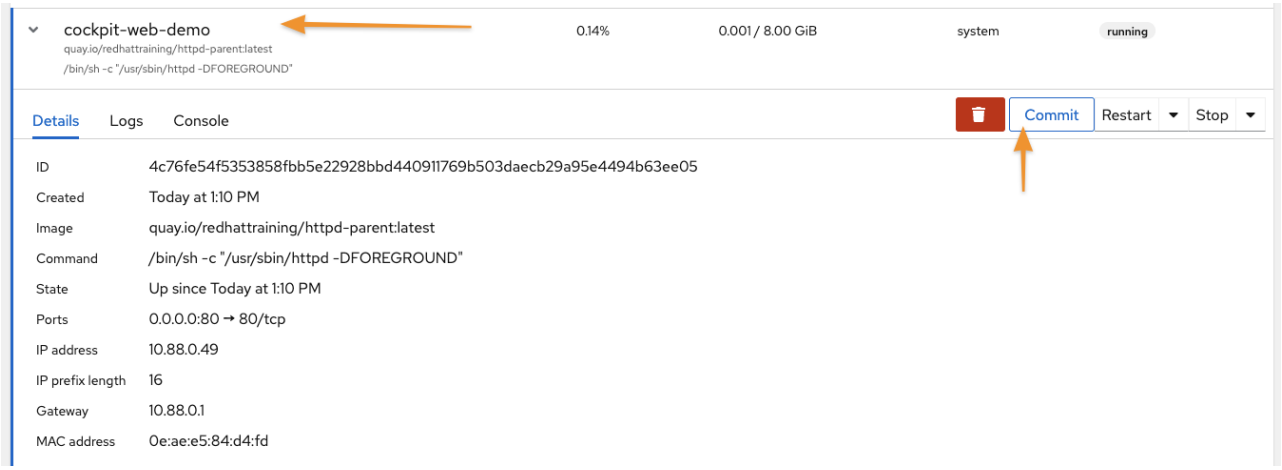


Figure 29. Creating an new image from a container.

2. Select and complete the **Commit image** form

a. Specify **Image name**

b. Other fields are optional

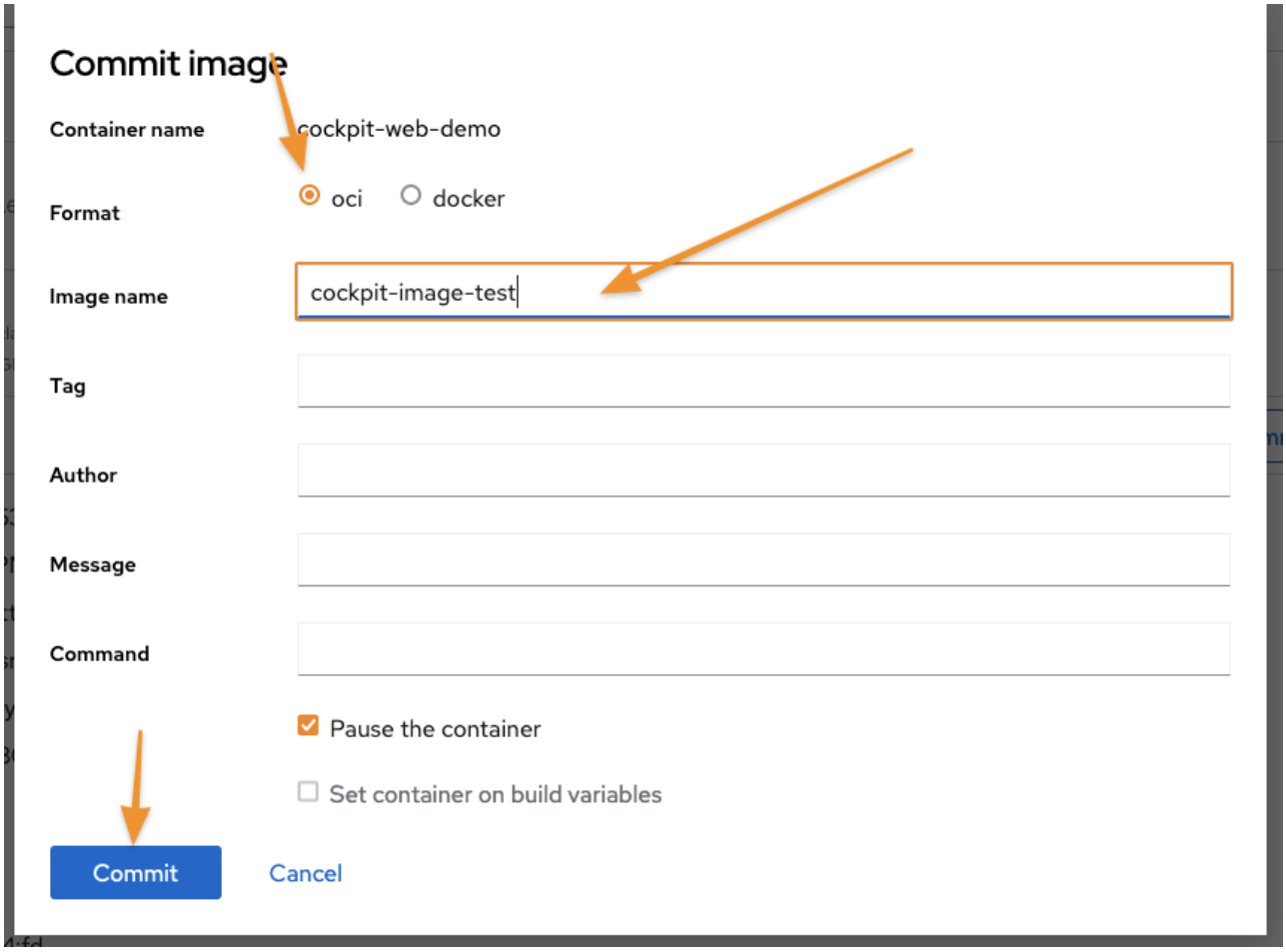


Figure 30. Committing image

Images from a Running Container

It is possible to create images from running container. If the container is running, it will be paused briefly while the image is created.

3. Verify the image was created

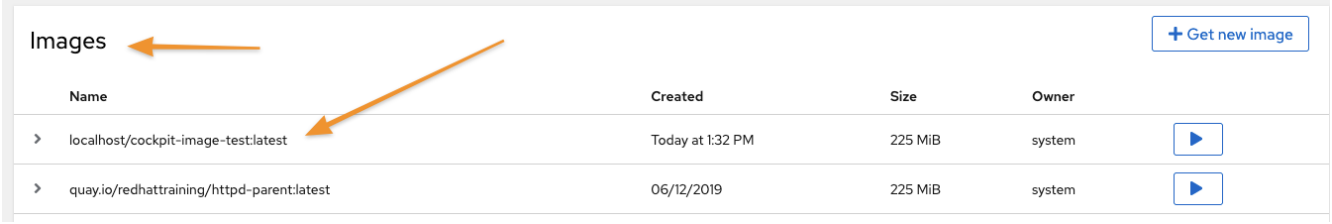


Figure 31. Image Verification

Image Tags

If there is no tag specified when the image is being committed, **podman** will automatically add the ***latest*** tag to the image.

Container Cleanup

1. Select the container to stop and remove.
   a. If you just want to stop a running container, click **Stop**
   b. To stop and remove the container, click the **Trashcan**

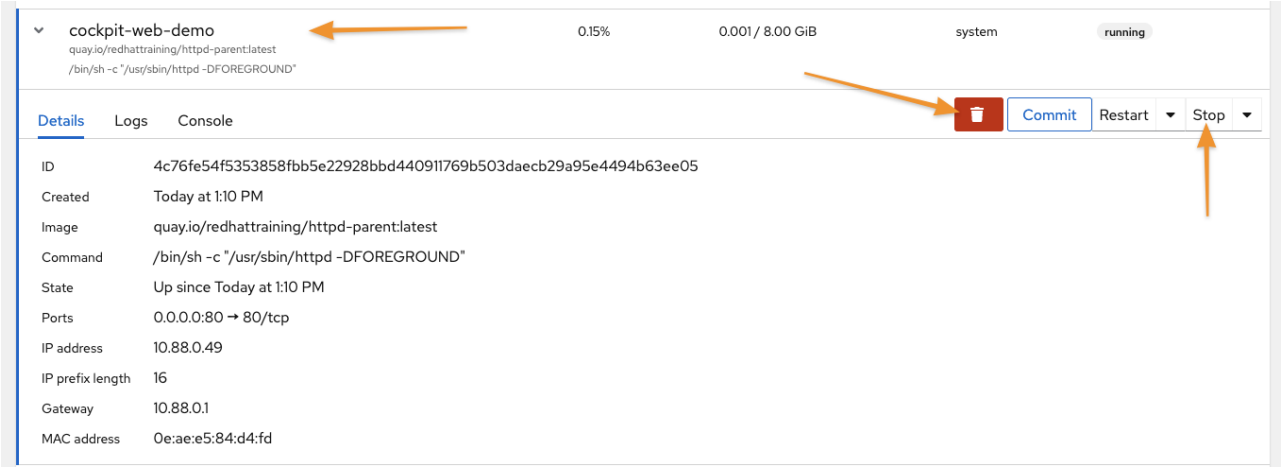


Figure 32. Container Cleanup

Deleting a Running Container

Running containers must be stopped before they can be deleted or removed. If a container is running, and you click the **Trashcan** it will stop and delete the container. This is similar to performing a **podman rm -f <Container>** as it will force stop and then remove the container.

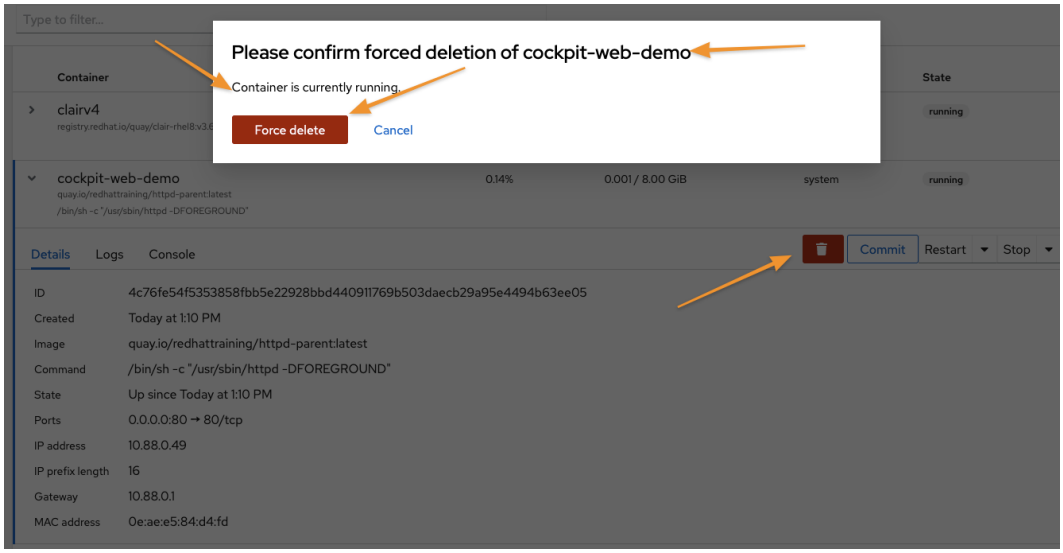


Figure 33. Force Removing a Running Container

References

Oracle Linux: Use Cockpit to Manage Podman Containers: https://docs.oracle.com/en/operating-systems/oracle-linux/8/tutorial-cockpit-podman/#Before-You-Begin

Managing your Podman containers with Cockpit on Fedora: https://www.tutorialworks.com/podman-monitoring-cockpit-fedora/

3. Podman Pods

This will talk about Podman Pods

3.1. Using and Leveraging Pods with Podman

3.2. Podman Image and Container Pruning

References

Managing Containers and Pods: https://developers.redhat.com/blog/2019/01/15/podman-managing-containers-pods?ts=1634314817672#podman_pods__what_you_need_to_know

Managing Containers using Podman and Skopeo: https://www.tecmint.com/manage-containers-using-podman-in-rhel/

Podman: https://docs.podman.io/en/latest/

Moving from docker-compose to Podman pods: https://www.redhat.com/sysadmin/compose-podman-pods

Podman System Prune:

man pages

podman-system-prune, *podman-container-cleanup*,

4. Container Images

4.1. Building Containers with Buildah

4.1.1. Building images as the root user

Example 2. EXAMPLE - Creating a Custom Container Image Using Buildah

Listing 40. Creating a Custom Container

```
[root@workstation ~]# buildah from scratch
working-container
```

Listing 41. Naming and Inspecting a Custom Container

```
[root@workstation ~]# buildah config --label name=My-Container working-container
[root@workstation ~]# buildah inspect working-container
```

Listing 42. Installing Packages on Working Container

```
[root@workstation ~]#  buildah mount working-container ①


[root@workstation ~]# yumdownloader --destdir=/tmp redhat-release-server ②

[root@workstation ~]# rpm -ivh --root
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged /tmp/redhat-release-8.0-
0.39.el8.x86_64.rpm ③

[root@workstation ~]# cp  /etc/yum.repos.d/rhel_dvd.repo
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged/etc/yum.repos.d/ ④

[root@workstation ~]# yum install --installroot
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged httpd ⑤


[root@workstation ~]# echo "This is a custom webserver container for me" >>
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged/var/www/html/index.html ⑥

[root@workstation ~]#  yum install --installroot
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged httpd-manual ⑦

[root@workstation ~]# buildah config --cmd "/usr/sbin/httpd -DFOREGROUND" working-container ⑧

[root@workstation ~]# buildah config --port 80/tcp working-container ⑨

[root@workstation ~]# yum clean all --installroot
/var/lib/containers/storage/overlay/a6a136063f0ada2b1ed4b01eff9a04b4d6419ae828bc4b49e742bca594e08560/merged ⑩

[root@workstation ~]#  buildah unmount working-container ⑪

[root@workstation ~]# buildah commit working-container my-container-image ⑫

[root@workstation ~]# buildah images ⑬
```

① Mount container image filesystem for modification

② Download Red Hat Release RPM for installation

③ Install Red Hat Release RPM

④ Create repository for container image so files can be installed

⑤ Install the HTTP package for a webserver

⑥ Create an **index.html** file for the webserver

⑦ Install the Apache manual for reference documentation

⑧ Configure webserver to run

⑨ Configure and open port **80** for the **TCP** protocol for the container

⑩ Clean up yum data to minimize required disk space

⑪ Unmount the container image filesystem

⑫ Commit the container image

⑬ List container images

Listing 43. Testing the Container Image

```
[root@workstation ~]# podman run -d -p 8080:80 localhost/my-container-image

[root@workstation ~]# curl localhost:8080
This is a custom webserver container for me

[root@workstation ~]# curl http://localhost:8080/manual/
```

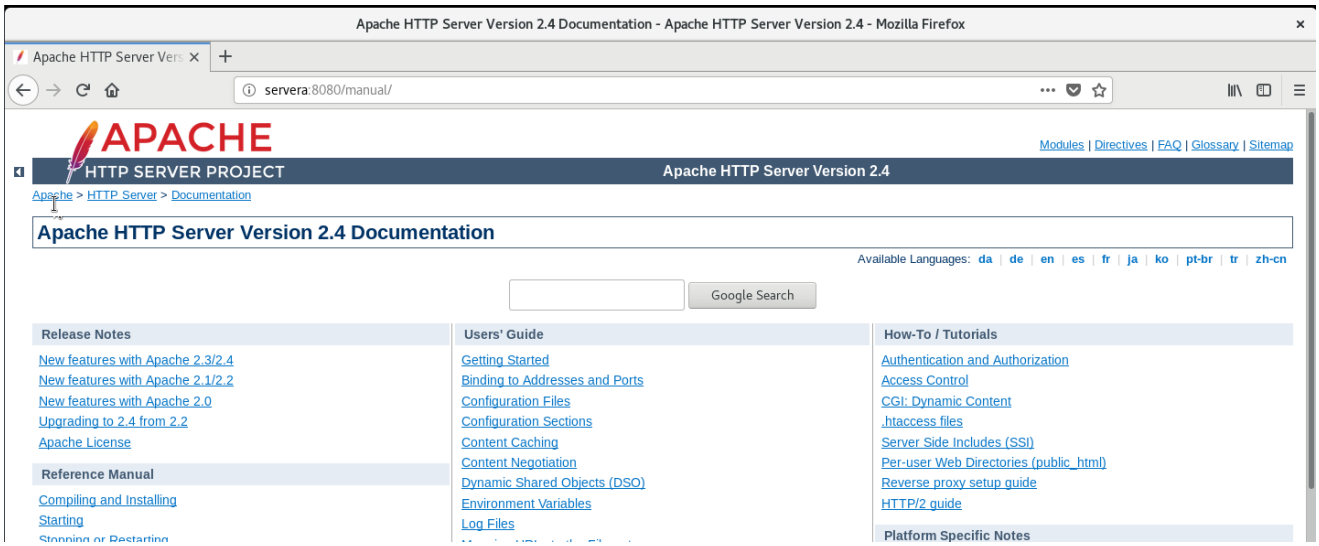


Figure 34. Testing Container

Listing 44. Stopping and Cleanup of Image

```
[root@workstation ~]# podman list ①
CONTAINER ID  IMAGE                                COMMAND               CREATED        STATUS           PORTS                   NAMES
9bd572633953  localhost/my-container-image:latest  /usr/sbin/httpd -...  2 seconds ago  Up 1 second ago  0.0.0.0:8080->80/tcp
cranky_stonebraker

[root@workstation ~]# podman stop 9bd572633953 ②
9bd572633953276ac75417db3ac8e70875a0f2713e8cdfd32253fe343d06153d

[root@workstation ~]# podman stop -a ③

[root@workstation ~]# podman rm cranky_stonebraker ④

[root@workstation ~]# podman rm 9bd572633953276ac75417db3ac8e70875a0f2713e8cdfd32253fe343d06153d ⑤

[root@workstation ~]# podman rmi localhost/my-container-image ⑥

[root@workstation ~]# buildah delete working-container ⑦
```

① Listing Running Containers

② Stopping Single Container by ID

③ Stopping All Running Containers

④ Remove Container by Name

⑤ Remove Container by ID

⑥ Removing Container Image from Registry

⑦ Delete Working Container from System

4.1.2. Building an Image Using Buildah Rootless

4.2. Managing Images and System Storage

References

Getting into the weeds with Buildah: The buildah unshare command: https://www.redhat.com/sysadmin/buildah-unshare-command

How rootless Buildah works: Building containers in unprivileged environments: https://opensource.com/article/19/3/tips-tricks-rootless-buildah

Building and managing container images with Buildah: https://mohitgoyal.co/2021/05/16/building-and-managing-container-images-with-buildah/

podman-image-prune: https://docs.podman.io/en/latest/markdown/podman-image-prune.1.html

podman-system-prune: https://docs.podman.io/en/latest/markdown/podman-system-prune.1.html

podman-container-prune https://docs.podman.io/en/latest/markdown/podman-container-prune.1.html

Man Pages: *man podman-image-prune*, *man podman-system-prune*, *man podman-container-prune*, *man buildah*, *man podman*

5. Containers as System Services

References

Managing containerized system services with Podman: https://developers.redhat.com/blog/2018/11/29/managing-containerized-system-services-with-podman

6. Container Management with Ansible

Containers can now be managed with the Ansible Podman collection leveraging Ansible Automation Plaatform (AAP). Red Hat curates and maintains several supported Ansible Collections at Ansible Automation Hub. The community collections are available to be installed from Ansible Galaxy.

6.1. Ansible Refresher

The following is a small reminder about Ansible and the basics around an Ansible Control node for managing systems with Ansible.

6.1.1. Ansible Basics

In order to use Ansible, the following items are required:

- **ansible** is installed on the Control Node
- An **ansible.cfg** file exists and points to a working inventory
- An **inventory** file exists containing managed nodes

6.1.1.1. Ansible Concepts and Architecture

Ansible Machine Types

- **Control Node**: Location where Ansible is installed and used to run playbooks and execute Ansible ad-hoc commands
- **Managed Host**: Network device that is managed by an Ansible control node

The required Ansible components are the following:

- **ansible.cfg**: Ansible configuration file with directives on how Ansible should work
- **inventory**: Listing and organization of ansible managed nodes/hosts
- **module**: Small piece of code to perform a variety of tasks (generally written in Python or Powershell)
- **plugins**: Code that can be used to extend and adapt Ansible to new uses. Capable of manipulating data and extracting information from output. (*Used more in the next course DO447*)
- **Ansible Tower/API**: Framework to control and manage Ansible at scale. Ansible Tower is not a core part of Ansible, but is an add-on product to more effectively utilize Ansible with teams.

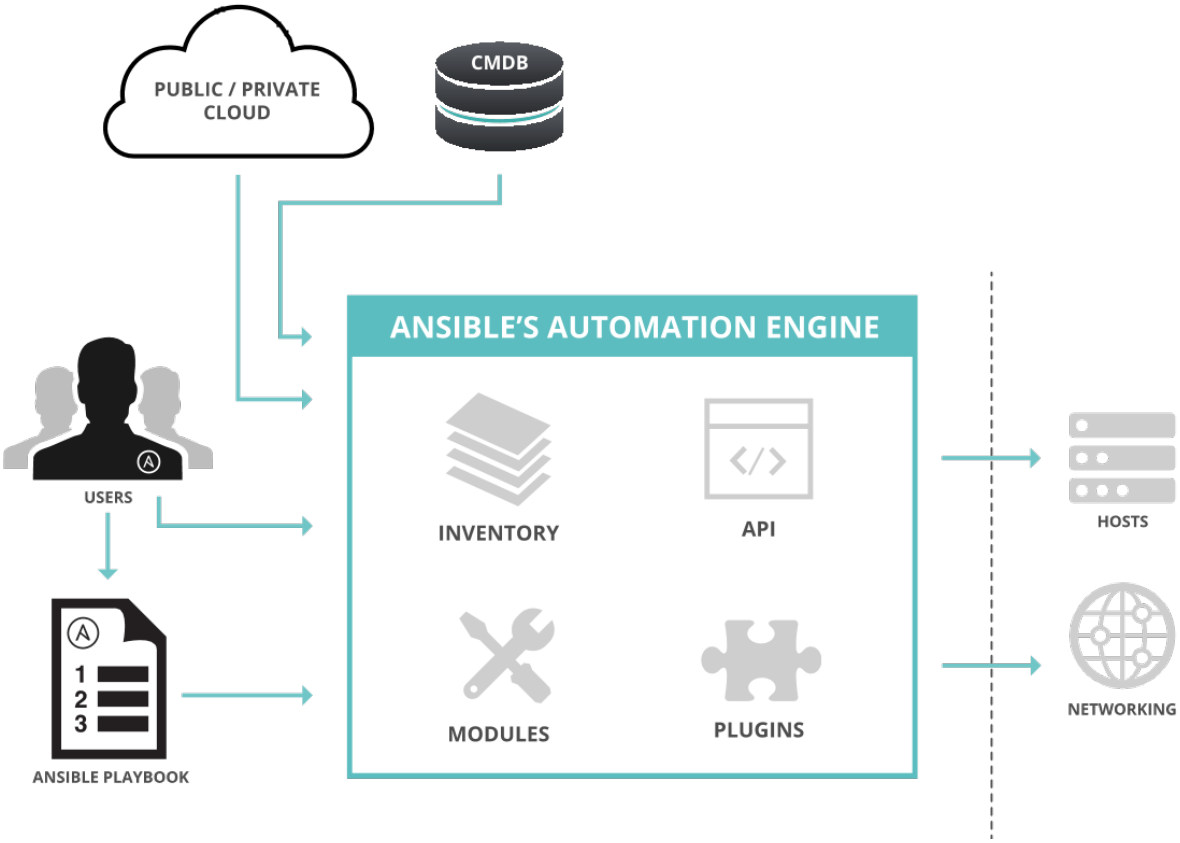


Figure 35. Ansible Architecture

6.1.1.2. Building an Ansible Inventory

An **inventory** file is the minimum needed item for Ansible to run. Inventory files provide a listing of hosts for Ansible to manage. There are multiple ways to define an inventory file.

6.1.1.2.1. Static Inventory

Static inventory files are generally defined in the INI format.

Listing 45. Simple Inventory

```
web1.example.com
web2.example.com
db1.example.com
db2.example.com
192.0.2.42

[webservers]
web1.example.com
web2.example.com
192.0.2.42
```

6.1.1.3. Ansible Configuration Files

It is important to note, there can be multiple Ansible configuration files. Ansible will look for and process configuration files in order.

1. Location specified in **ANSIBLE_CONFIG**
2. The **ansible.cfg** file in the current working directory
3. The **.ansible.cfg** file in the user's home directory
4. The default **/etc/ansible/ansible.cfg** file

It is important to note that the first file in this order Ansible sees is what will be used. Therefore, if anything is setup for the **ansible.cfg** file in locations 1-3, it won't ever use the default file ***/etc/ansible/ansible.cfg***

Ansible Configuration file documentation: http://docs.ansible.com/ansible/intro_configuration.html

Table 1. **Ansible Settings**

Setting	Command Line Option
inventory	Location of the Ansible inventory file.
remote_user	The remote user account used to establish connections to managed hosts.
become	Enables or disables privilege escalation for operations on managed hosts.
become_method	Defines privilege escalation method on managed hosts.

Setting	Command Line Option
become_user	User account to escalate privileges to on managed hosts.
become_ask_pass	Defines whether privilege escalation on managed hosts should prompt for password.

Ansible Collections

Starting with Ansible 2.9 and the new Ansible (AAP), collections have been introduced and allow modules, roles, and other components to be stored in Ansible collections. Similar to Ansible Roles, Ansible Collections must be installed on the system and available for Ansible playbooks.

Ansible Collections can be specified in the **ansible.cfg** file for the path where collections have been installed. As with Ansible Roles, it is common to install Ansible collections to the working directory in a sub-directory called **collections**. When working with Ansible Collections it is necessary to provide this information in the **ansible.cfg** file.

Listing 46. **ansible.cfg** *for Using Collections*

```
[defaults]
remote_user = root
inventory = inventory
COLLECTIONS_PATHS = ~/.ansible/collections:/usr/share/ansible/collections:./collections

[privilege_escalation]
become = False
become_method = sudo
become_user = root
become_ask_pass = False
```

6.2. Ansible Podman Collection

The Ansible Podman collection is a required component in order to have the Ansible Modules needed to manage containers using Podman. Currently, the **containers.podman** is only available via Ansible Galaxy.

6.2.1. Obtaining Podman Collections

In order to leverage Ansible to official manage Podman containers, the **containers.podman** collection must be installed on the Ansible Control node. Currently, there are two main sources for installation of Ansible collections.

Ansible Collections

- Red Hat supported collections for Ansible can be downloaded from Ansible Automation Hub (https://console.redhat.com/ansible/automation-hub).
- Ansible Community collections can be downloaded from Ansible Galaxy (https://galaxy.ansible.com/containers/podman)

Ansible Automation Hub and Ansible Galaxy

It is required that you have a Red Hat Subscription for Ansible Automation Platform (AAP2.0) in order to access Red Hat Supported Ansible collections. The lab and exercises will use the Ansible Galaxy Podman collection.

Listing 47. Installing Ansible Galaxy Podman Collection

```
ansible-galaxy collection install containers.podman
```

containers.podman *Documentation*

https://galaxy.ansible.com/containers/podman https://docs.ansible.com/ansible/latest/collections/containers/podman/index.html https://docs.ansible.com/ansible/latest/collections/containers/podman/podman_container_module.html

6.2.2. Installing and Using the containers.podman Collection

There are multiple methods for the Podman collection to be installed, however, for this course and the exercises, the collection will be installed locally to the **./collections** sub-folder using a requirements.yml file. The **ansible.cfg** will point to this as already available in the path and the Ansible playbooks being utilized will reference the **collections** at the top of the playbook similar to Ansible roles so that throughout the playbook tasks, the shorter module name can be referenced.

Installing the Podman Collection

Similar to installation of Ansible Roles, Ansible Collections can be installed in the local working directory in a sub-directory called **collections**. The collections needed for the Ansible projects can be specified in a **requirements.yml** file and installed just like Ansible Roles.

1. Create a **requirements.yml** File

Listing 48. Podman Collection **requirements.yml** *File*

```
collections:
- name: containers.podman
```

2. Install the roles/collections from the **requirements.yml** File

Listing 49. Installing the Collections

```
[student@workstation Ansible_Image]$ ansible-galaxy collection install -r requirements.yml -p ./collections
Process install dependency map
Starting collection install process
Installing 'containers.podman:1.8.1' to
'/home/student/github/OCP_Demos/Containers/labs/Ansible_Image/collections/ansible_collections/containers/podman'
```

3. Verify the collection(s) are installed

Listing 50. Verifying Installed Collections

```
[student@workstation Ansible_Image]$ tree collections/
collections/
└── ansible_collections
    └── containers
        └── podman
            ├── ansible-collection-containers-podman.spec
            ├── CHANGELOG.rst
            ├── changelog

... OUTPUT OMITTED ...

└── sanity
    ├── ignore-2.10.txt
    ├── ignore-2.11.txt
    ├── ignore-2.12.txt
    ├── ignore-2.9.txt
    └── requirements.txt

64 directories, 146 files
```

Using Collections

Referencing the **collection** and using shorter module names. This allows a cleaner and more streamlined approach and is generally consistent with the older Ansible versions. It is 100% fully acceptable to utilize the fully qualified module names, however, for the course and ease of use we will continue referencing Ansible modules by the shortened/condensed name.

Listing 51. Sample Playbook with Collections

```
---
- name: Deploy Quay Mirror
  hosts: quay
  vars:
    QUAY_DIR: /quay
  vars_files:
    - registry_login.yml
  collections: ①
    - containers.podman

    tasks:

  ## Podman Collections Needed for Login
      - name: Login to Container Registry
        podman_login: ②
          username: "{{ registry_un }}"
          password: "{{ registry_pass }}"
          registry: "{{ registry_url }}"
```

① Importing the collection(s) to be used and referenced by short module names in the playbook.

② Leveraging modules by short module names in the playbook.

6.3. Building Container Images Using Ansible

The **podman_image** module from the **containers.podman** collection is needed in order to work with an manipulate container images. In order to build images, it is necessary to have either a valid **Containerfile** or valid **Dockerfile** for image building.

Building Podman Container Images with Ansible

1. Verify the **ansible.cfg** file for proper collection usage

Listing 52. **ansible.cfg**

```
[defaults]
remote_user = root
inventory = inventory
COLLECTIONS_PATHS = ~/.ansible/collections:/usr/share/ansible/collections:./collections
```

2. Start with the top playbook definitions and directives

Listing 53. Ansible Playbook Directives

```
---
- name: Build and Upload a Container Image
  hosts: localhost
  vars_files:
    - vars/registry_login.yml ①
  collections:
    - containers.podman ②

... OUTPUT OMITTED ...
```

① Always provide registry login information

② Always specify the collections to be used

3. Create Playbook with Tasks

Listing 54. Playbook Tasks

```
tasks:
  - name: Login to Container Registry ①
    podman_login:
      username: "{{ registry_un }}"
      password: "{{ registry_pass }}"
      registry: "{{ registry_url }}"

  - name: Build and Push Custom Container Image ②
    podman_image:
      name: httpd_demo_ansible ③
      path: ./ ④
      push: yes ⑤
      username: "{{ registry_un }}" ⑥
      password: "{{ registry_pass }}" ⑦
      push_args: ⑧
        dest: "{{ registry_url }}/{{ registry_un }}"
```

① Initial task to login to registry

② Task using the **podman image** module to build and push an image

③ Name/Tag of the image being built

④ Path to the **Containerfile** or **Dockerfile**

⑤ Define whether or not to push to remote image registry

⑥ Registry Username

⑦ Registry Password

⑧ Arguments used for pushing to remote registry. Works in conjunction with **push: yes**. Must have registry URL specified along with the location (*which is generally the repository username*).

6.4. Deploying Podman Containers Using Ansible

The **podman_container** module from the **containers.podman** collection is needed in order to create/run/remove containers. In order to create containers from container images, it is necessary to have access to the container images locally.

Creating and Running Podman Containers with Ansible

1. Verify the **ansible.cfg** file for proper collection usage

Listing 55. **ansible.cfg**

```
[defaults]
remote_user = root
inventory = inventory
COLLECTIONS_PATHS = ~/.ansible/collections:/usr/share/ansible/collections:./collections
```

2. Start with the top playbook definitions and directives

Listing 56. Ansible Playbook Directives

```
---
- name: Build and Upload a Container Image
  hosts: localhost
  vars_files:
    - vars/registry_login.yml ①
  collections:
    - containers.podman ②

... OUTPUT OMITTED ...
```

① Always provide registry login information

② Always specify the collections to be used

3. Create the Playbook with Tasks

Listing 57. Playbook Tasks

```
tasks:

## Podman Collections Needed for Login
  - name: Login to Container Registry
    podman_login:
      username: "{{ registry_un }}"
      password: "{{ registry_pass }}"
      registry: "{{ registry_url }}"

## Start and Run the HTTPD Container
  - name: Start the Quay Container
    podman_container:
      name: Website_Demo
      image: quay.io/redhattraining/httpd-parent:2.4
      state: started
      restart: yes
      ports:
        - "7080:80"
      volume:
        - "/Webhosting:/var/www/html:Z"
```

4. Analyzing Running Containers with Ansible

The **podman collection** has several components to manage and collect information from containers. The **podman_container_info** module can be used to get information about running containers. It requires the name of the container and collects all information as a list/dictionary.

Podman Container Info Module

It is important to note that many of the Ansible **fact** modules have been replaced with **info** modules so the facts are returned as lists/dictionaries. You should consult each module's information page and examine the **Return Values** section.

podman_container_info: https://docs.ansible.com/ansible/latest/collections/containers/podman/podman_container_info_module.html#ansible-collections-containers-podman-podman-container-info-module

It is also important to know that in order to access specific elements from the information returned, a mapping of the value must be performed.

Listing 58. Mapping Element Needed

```
- name: Gather Information on Running Containers
  podman_container_info: ①
    name: Website_Demo
  register: container_info

- name: Set Fact for Running Container Image
  set_fact: ②
    ImageName: "{{ container_info.containers | map(attribute='ImageName') | list }}"

- name: Display Running Container Image
  debug: ③
    var: ImageName
```

① Using the **podman_container_info** to gather all facts and information

② Using the **set_fact** module to map the attribute of the information to a known variable name

③ Displaying the information from the set fact

1. Controlling Running Containers with Ansible

The **podman collection** has several components to manage and collect information from containers. The **podman_container** module can be used to interact with containers.

Listing 59. Example of Stopping / Removing a Container and Deleting a Container Image

```
---
- name: Container Information
  hosts: localhost
  vars_files:
    - vars/registry_login.yml
  collections:
    - containers.podman

  tasks:

    - name: Stop Running Container
      podman_container:
        name: Website_Demo
        state: stopped

    - name: Remove Stopped Container
      podman_container:
        name: Website_Demo
        state: absent

    - name: Remove Container Image
      podman_image:
        name: quay.io/redhattraining/httpd-parent:2.4
        state: absent
```

References

Podman Collections Github Project: https://github.com/containers/ansible-podman-collections

Podman Collection Ansible Galaxy: https://galaxy.ansible.com/containers/podman

Podman Collections from Ansible Docs: https://docs.ansible.com/ansible/latest/collections/containers/podman/

7. Quay Image Registry

The Quay image registry depends on multiple sources and configuration files. The exercises here will go through a local "Proof of Concept" deployment. It will be necessary to open up firewall ports as well as provide persistent storage mount points that can be mounted and accessed within the containers.

There are multiple options for deploying Quay locally. For the purpose of this course, we will be installing the Quay registry with both the Security Scanner and the Mirroring Worker.

Quay Deployment Options

- Stand-Alone Quay Registry
- Quay Registry with a Mirroring Worker
- Quay Registry with Clair Security Scanner
- **Quay Registry with Clair Security Scanner and Mirroring Worker**

SSL and Security

All deployment scenarios allow the use of creating SSL/TLS certificates to secure the network connection. For the purpose of this content, we will be omitting the steps for generating the SSL/TLS certificates and accessing the registry via HTTPS.

Table 2. Network Ports for Firewalls and Port Mapping

Service Name	Network Port
Quay HTTPS	8443/TCP
Quay HTTP	8080/TCP
Quay HTTP	80/TCP

Service Name	Network Port
Quay HTTPS	8443/TCP
Postgresql Quay	5432/TCP
Postgresql ClairV4	5433/TCP
Redis for Quay	6379
ClairV4 HTTP	8081/TCP

Local Quay Instance Details

- **Server FQDN**: quay.local
- **Images Used**:
  - registry.redhat.io/rhel8/postgresql-10:1
  - registry.redhat.io/rhel8/redis-5:1
  - registry.redhat.io/quay/quay-rhel8:v3.6.0-62
  - registry.redhat.io/quay/clair-rhel8:v3.6.0-70

Table 3. Container Port Mapping

Container Name	Mapped Port
clairv4	8001:8080
postgres-clairv4	5433:5432
redis	6379:6379
postgres	5432:5432
quay (*Config Container Only*)	8080:8080
quay (*Running Container*)	80:8080

Table 4. Storage Mount Points

Container Name	Mount Point on Host System	Purpose
quay	/quay	Storage for configuration files and images. Multiple sub-directories under this local directory mounted to Quay, Postresql, Redis for persistent storage.
clairv4	/etc/clairv4 ⇒ /clair	Storage for configuration files and images.

7.1. Installing the Quay Image Registry Manually

7.2. Installing the Quay Image Registry with Ansible

There is a Github project that can be used to install a Quay image registry locally which includes a Mirroring and ClairV4 scanning process. The Github repository should have been cloned as part of the exercise in Chapter 1. The Github repository is: https://github.com/tmichett/quay_lab_poc and can deploy a fully functional Quay image registry locally.

The Quay playbooks have been broken down so that they can be run to easily deploy Quay. There are options for deploying based on the configuration files being pre-populated as well as using the Quay container to create the TGZ configuration. There is also a special playbook created to bring already deployed containers back online.

7.2.1. Deploying Quay with Ansible

1. Change to the **github/quay_lab_poc** directory

```
[student@workstation ~]$ cd github/quay_lab_poc/
```

2. Create a registry credential file and update the file with your information

Listing 60. Copy **registry_login.yml_example** *to* **registry_login.yml**

```
[student@workstation quay_lab_poc]$ cp registry_login.yml_example registry_login.yml
```

Listing 61. Edit the file **registry_login.yml**

```
[student@workstation quay_lab_poc]$ vim registry_login.yml
registry_un: UN_Goes_Here ①
registry_pass: Password_Goes_Here ②
registry_url: registry.redhat.io
```

① Replace with your Red Hat Login ID

② Replace with your Red Hat Login Password

3. Run the **Quay_Prepare.yml** playbook

```
[student@workstation quay_lab_poc]$ ansible-playbook Quay_Prepare.yml

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

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

... OUTPUT OMITTED ...

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

TASK [Open the Quay Config Container Site]
*********************************************************************************************************
[Open the Quay Config Container Site] ①
Go to the Quay Configuration container website and enter the appropriate configuration values. For help, look at
https://github.com/tmichett/quay_lab/References/Quay-3.5_Deployment.pdf. Login with the credentials provided which are UN: quayconfig and
PW: secret. Press 'Enter' when you've completed the configuration and downloaded the file. The file should be placed in the files
directory for this playbook.: ②

... OUTPUT OMITTED ...'

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

① This allows you to open the Quay Config Site to create a custom configuration file.

② Once you've opened the Quay config site and completed the configuration as well as downloaded the file you can hit "Enter". It is also possible to hit "Enter" and skip this step so that the already existing configuration file can be used.

Ansible Failure Possible

It is possible that you will receive an Ansible failure message like this

```
[student@workstation quay_lab_poc]$ ansible-playbook Quay_Prepare.yml
ERROR! vars file registry_login.yml was not found ①
Could not find file on the Ansible Controller.
If you are using a module and expect the file to exist on the remote, see the remote_src option
```

① This error means you forgot to create/edit the **registry_login.yml** file.

4. Run the configuration file deployment based on using the TAR config file or the file-based configuration method. ***NOTE: You can only choose one method for configuration.***

Listing 62. File-Based Configuration Method

```
[student@workstation quay_lab_poc]$ ansible-playbook Quay_Config_Deploy_Files.yml
```

File-Based Configuration Considerations

The **./files/config/config.yaml** file will be used and deployed to control the configuration of the Quay environment.

Listing 63. TAR File Configuration Method

```

```

TAR-Based Configuration Considerations

The **./files/quay-config.tar.gz** file will be used and deployed to control the configuration of the Quay environment. This file MUST have been created as part of the Quay configuration container process with the WebUI and it must be placed in the **./files/quay-config.tar.gz** before running the playbook.

5. Deploy the ClairV4 scanning image by executing the **Quay_Clair_Deploy.yml** playbook

```
[student@workstation quay_lab_poc]$ ansible-playbook Quay_Clair_Deploy.yml
```

6. Deploy the Quay container registry by executing the **Quay_Deploy.yml** playbook.

```
[student@workstation quay_lab_poc]$ ansible-playbook Quay_Deploy.yml
```

7. Deploy the Quay Mirroring Container by executing the **Quay_Mirror_Deploy.yml** playbook.

```
[student@workstation quay_lab_poc]$ ansible-playbook Quay_Mirror_Deploy.yml
```

7.2.2. Setting up the Quay Web Console

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.

7.2.2.1. 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.

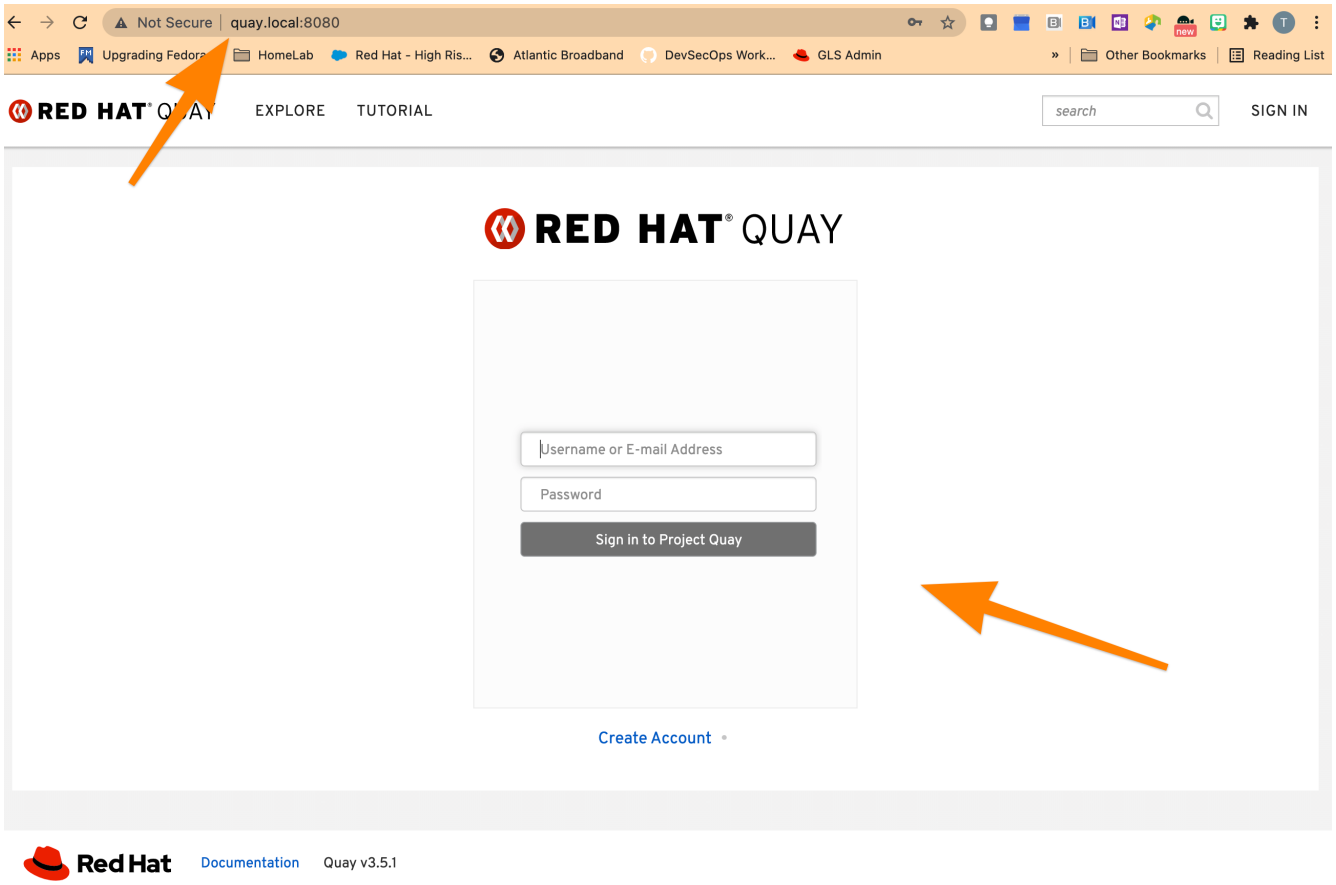
Configure Quay Super Users

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.

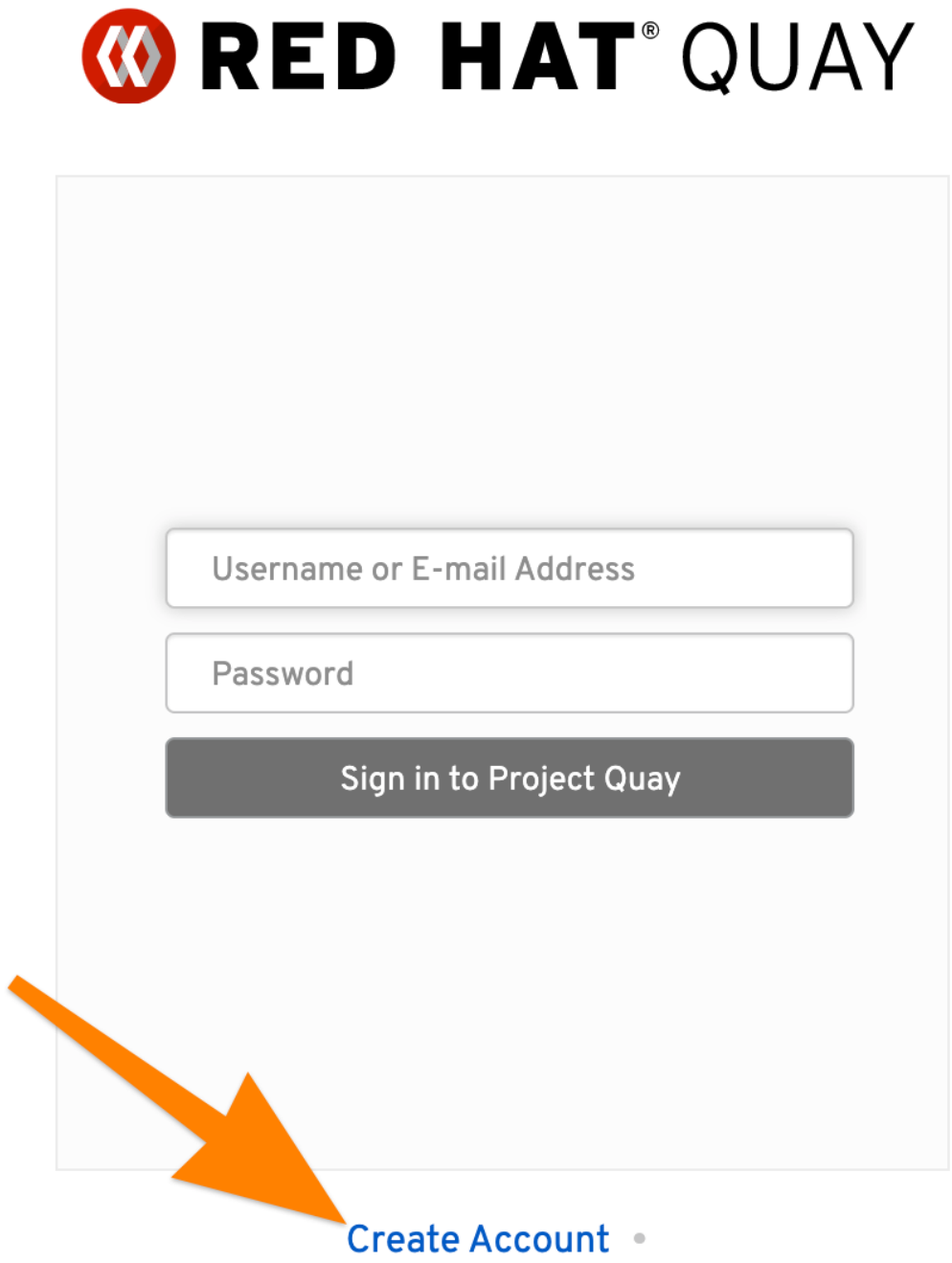
Listing 64. Quay Super Users

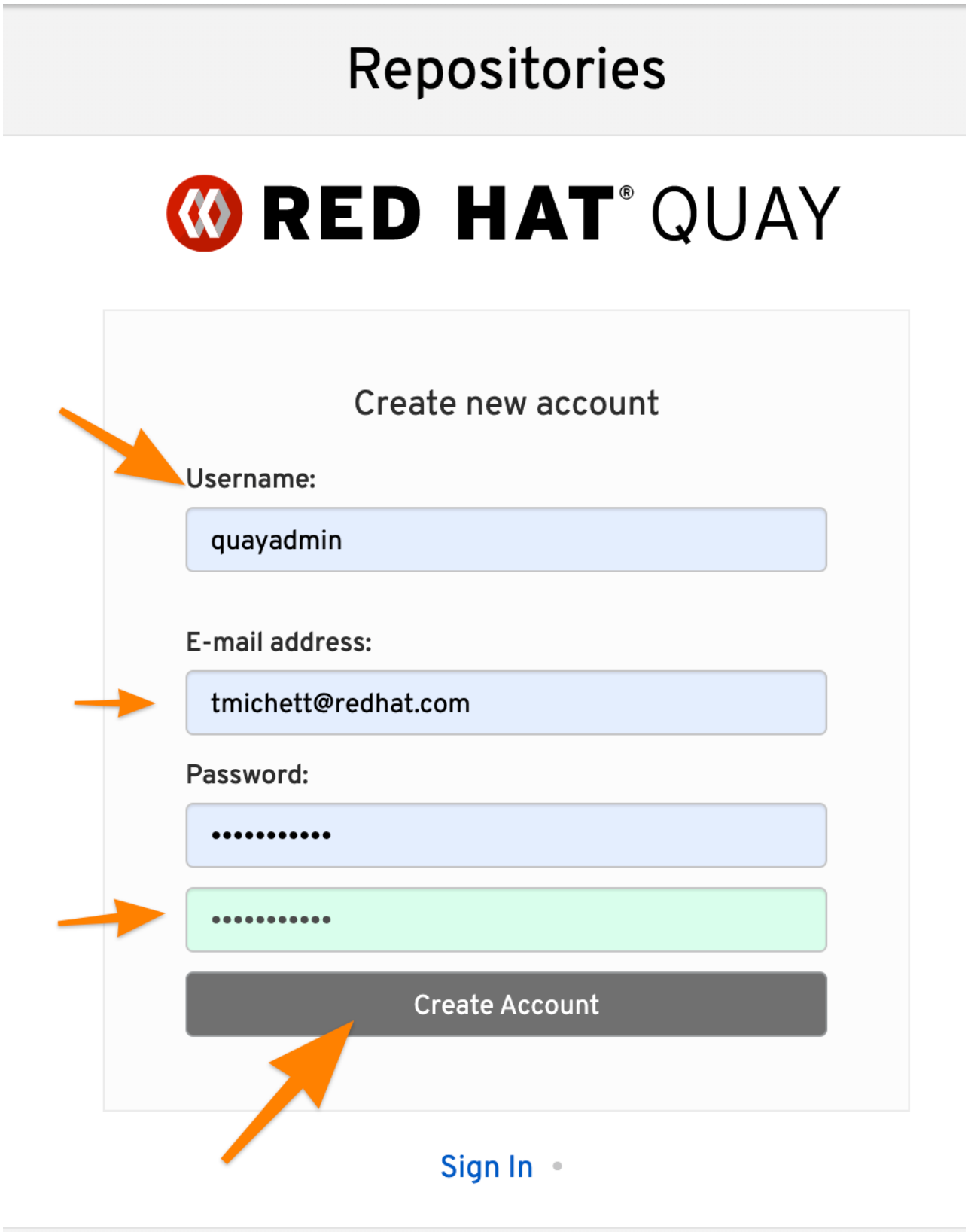
```
SUPER_USERS:
    - quayadmin
    - travis
```

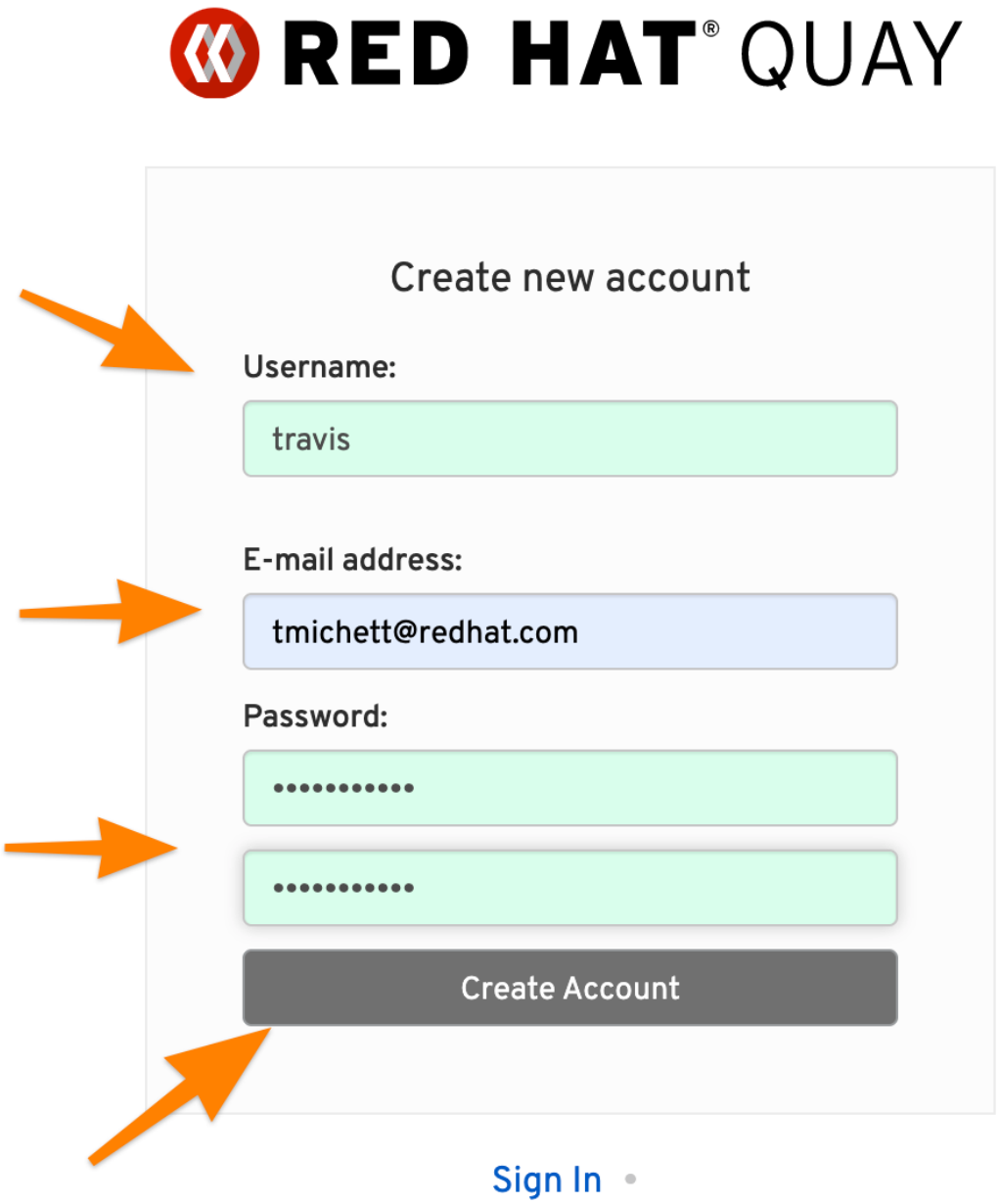
1. Open the Quay web console by navigating to it in your favorite browser using **http://Quay-FQDN:8080**



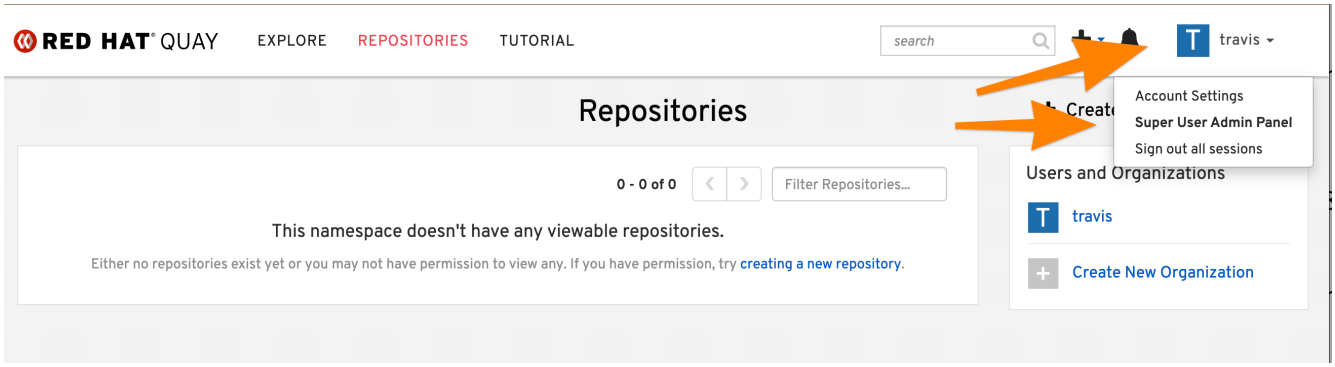
2. Click **Create Account** to create the administrator/superuser accounts for Quay as defined in the **config.yaml** file.
    - Repeat this step for all super users in the **config.yaml** file.


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**.



7.2.3. Testing Quay and ClairV4 Image Scanning

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 65. **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 66. 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
```

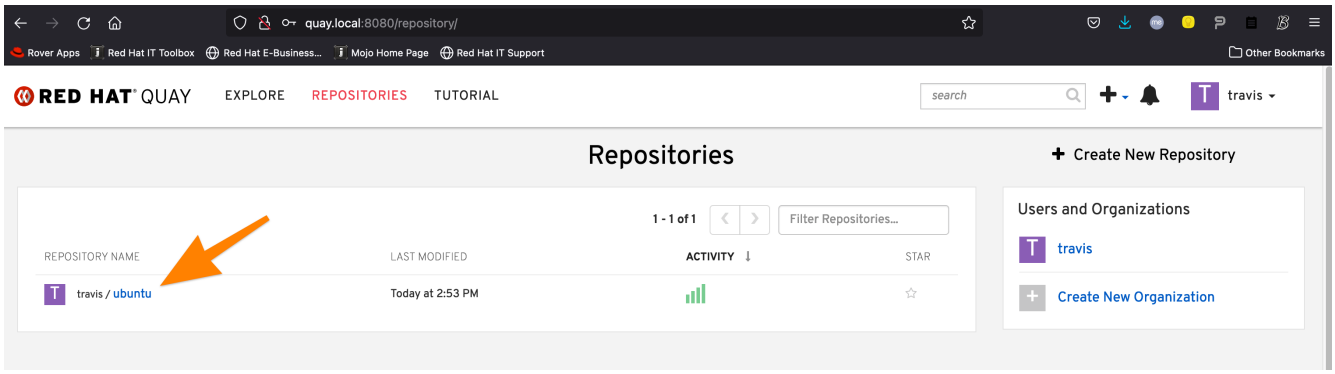
Listing 67. Tagging image

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

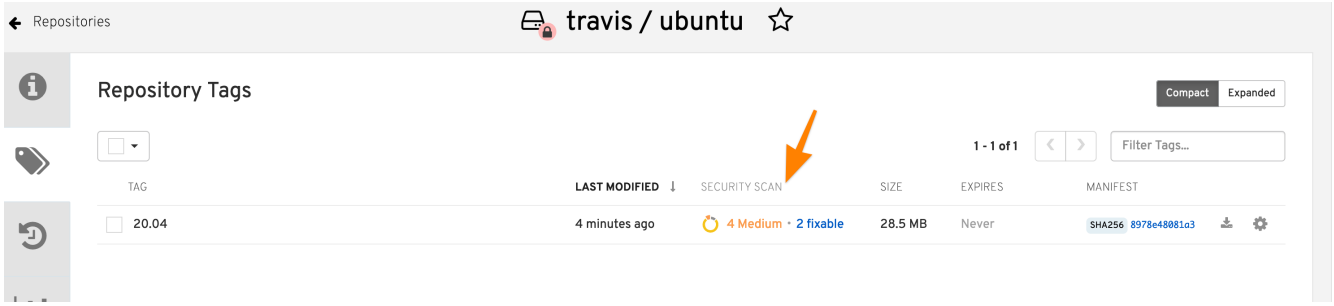
Listing 68. Push image

```
[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
```

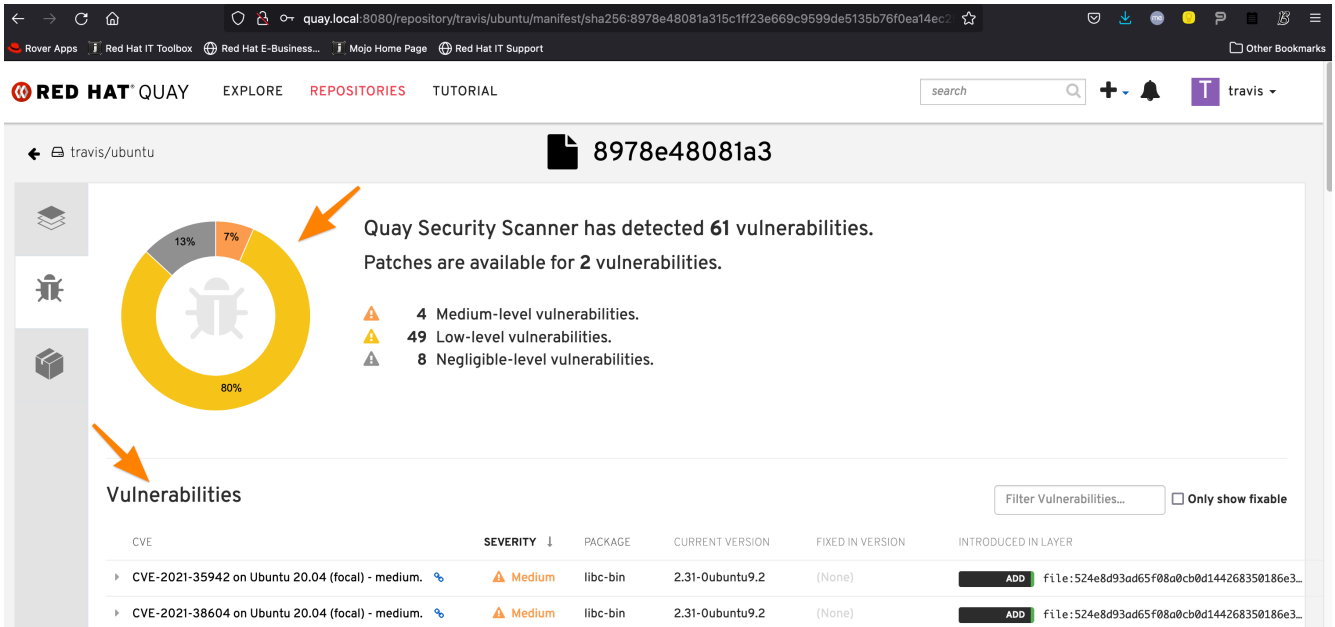
3. Verify image exists in Quay



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



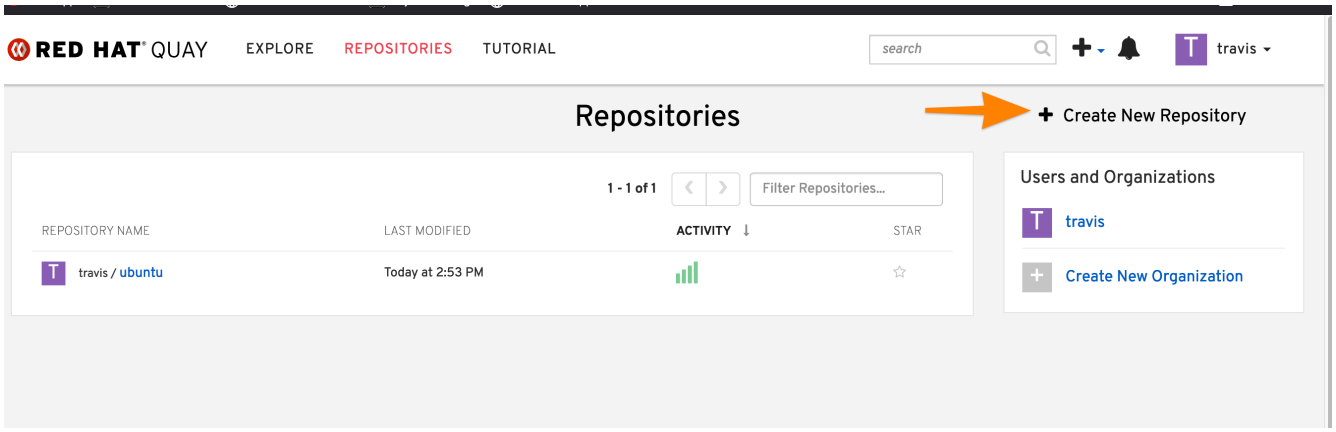
5. Click on Security scan to view the vulnerabilities



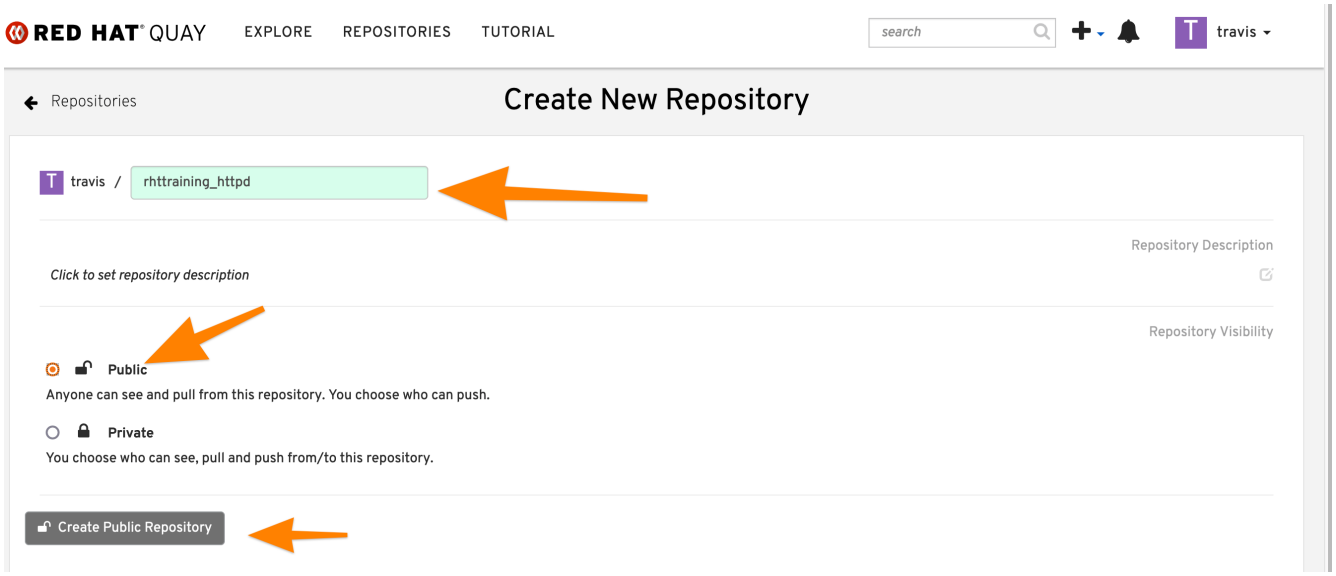
7.2.4. Testing Repository Mirroring

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.

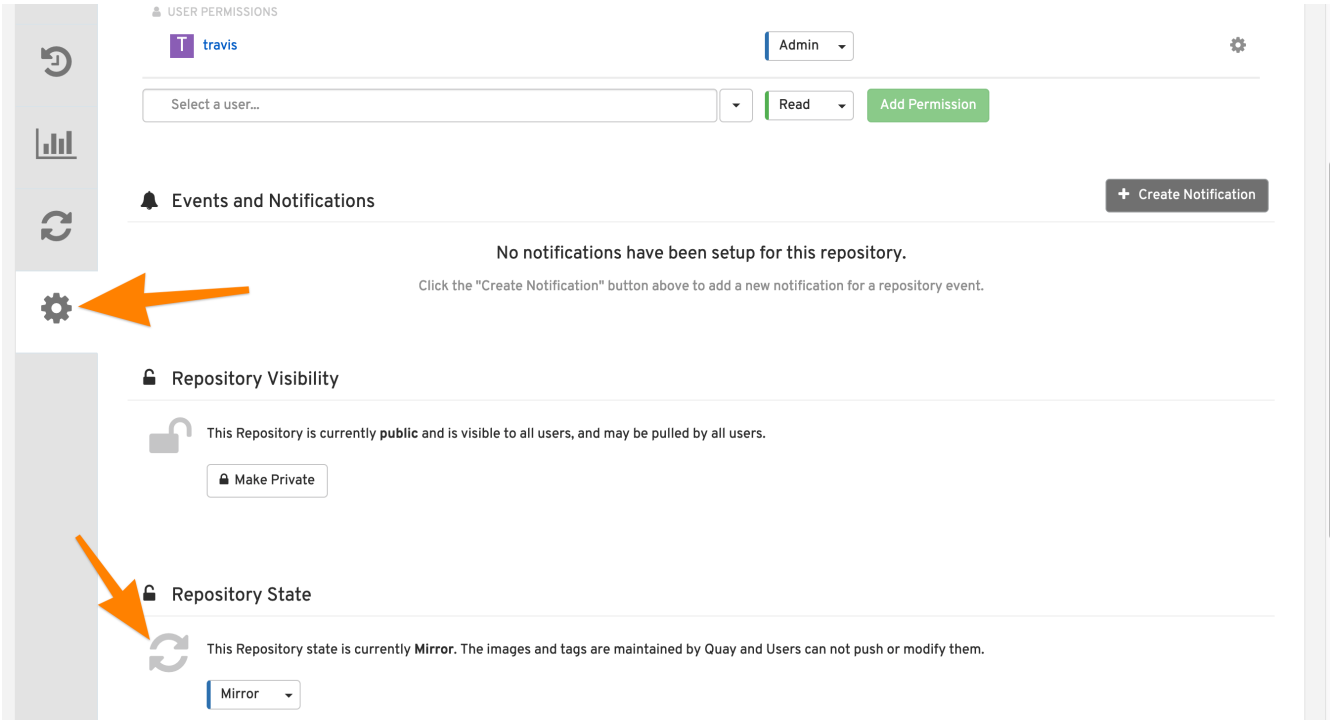
1. Create a new repository in Quay by clicking **Create New Repository**



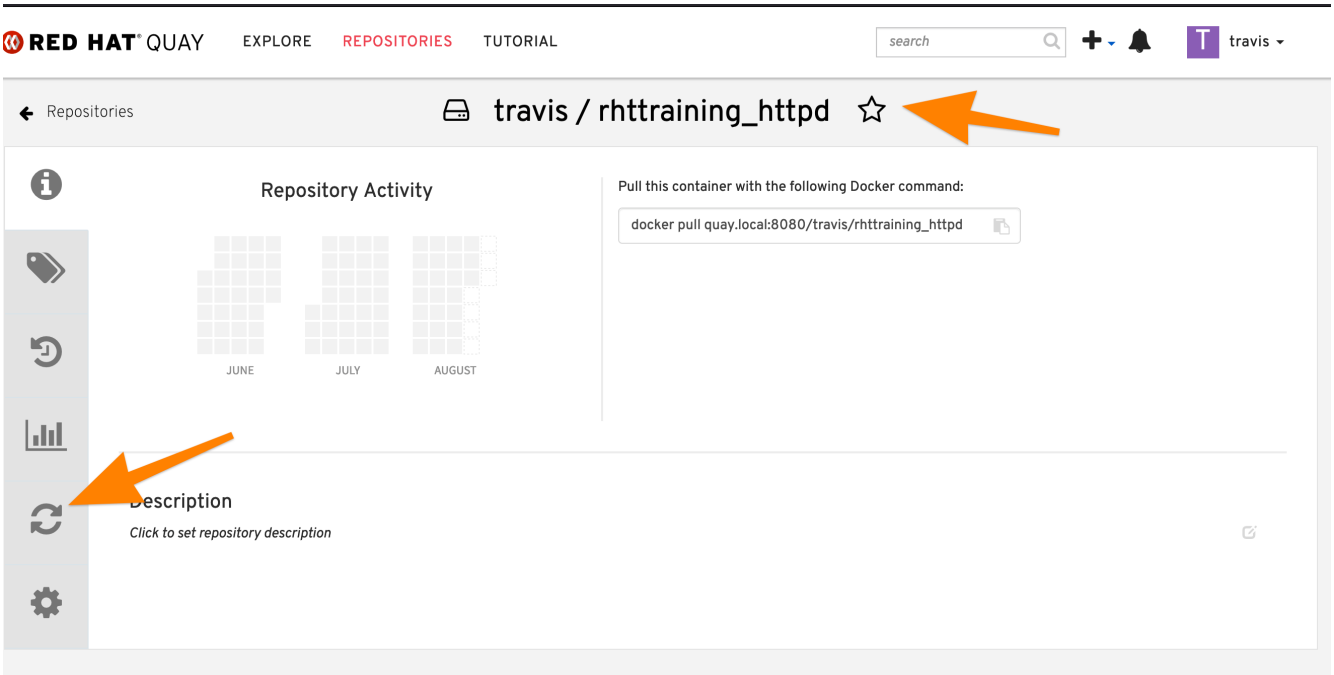
2. Give repository a name and setup the repository visibility



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



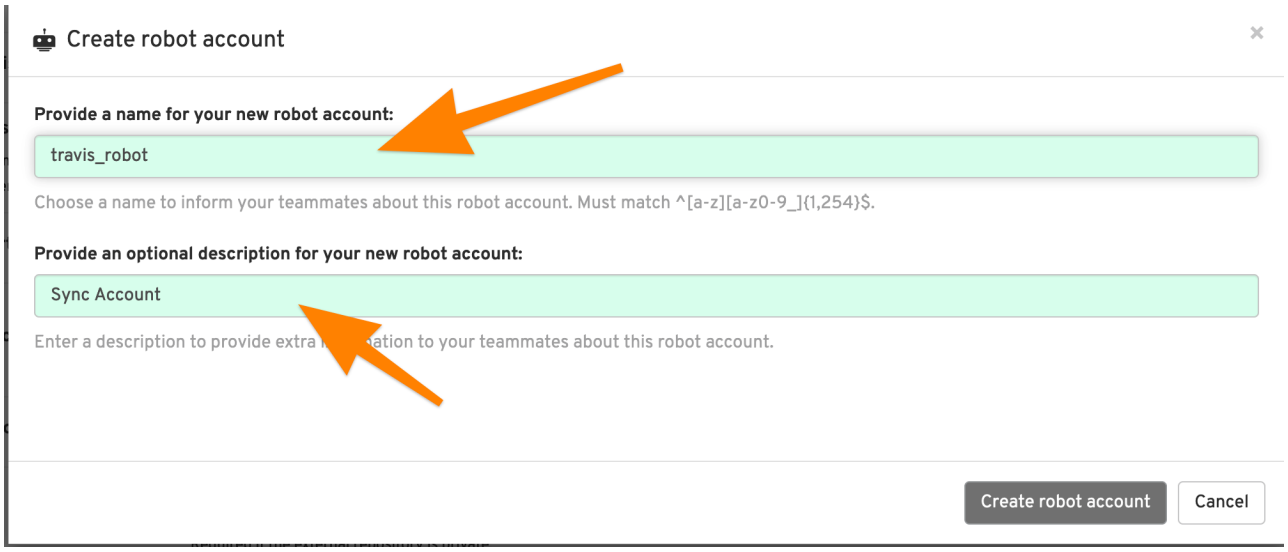
4. In the newly created repository, click the **Mirroring** option from the left-side navigation menu.

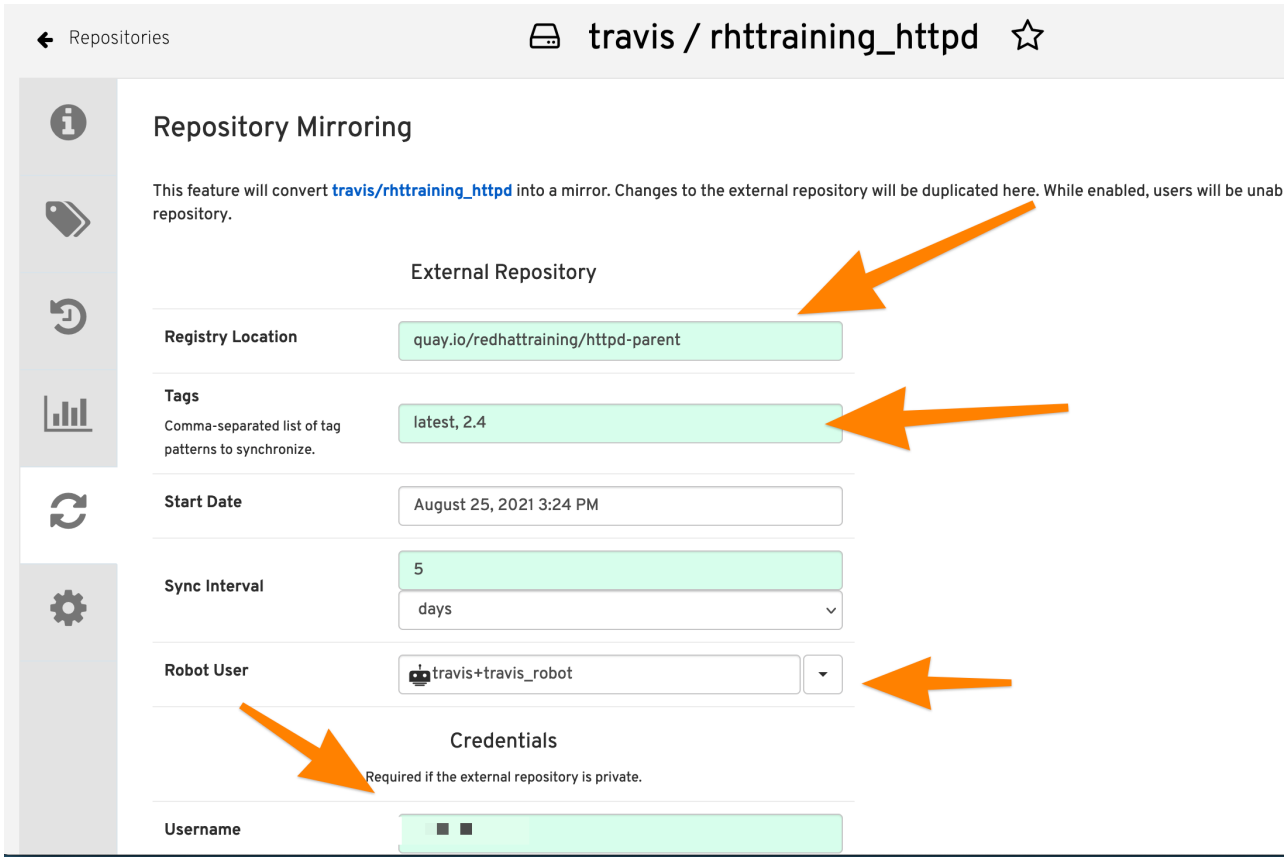


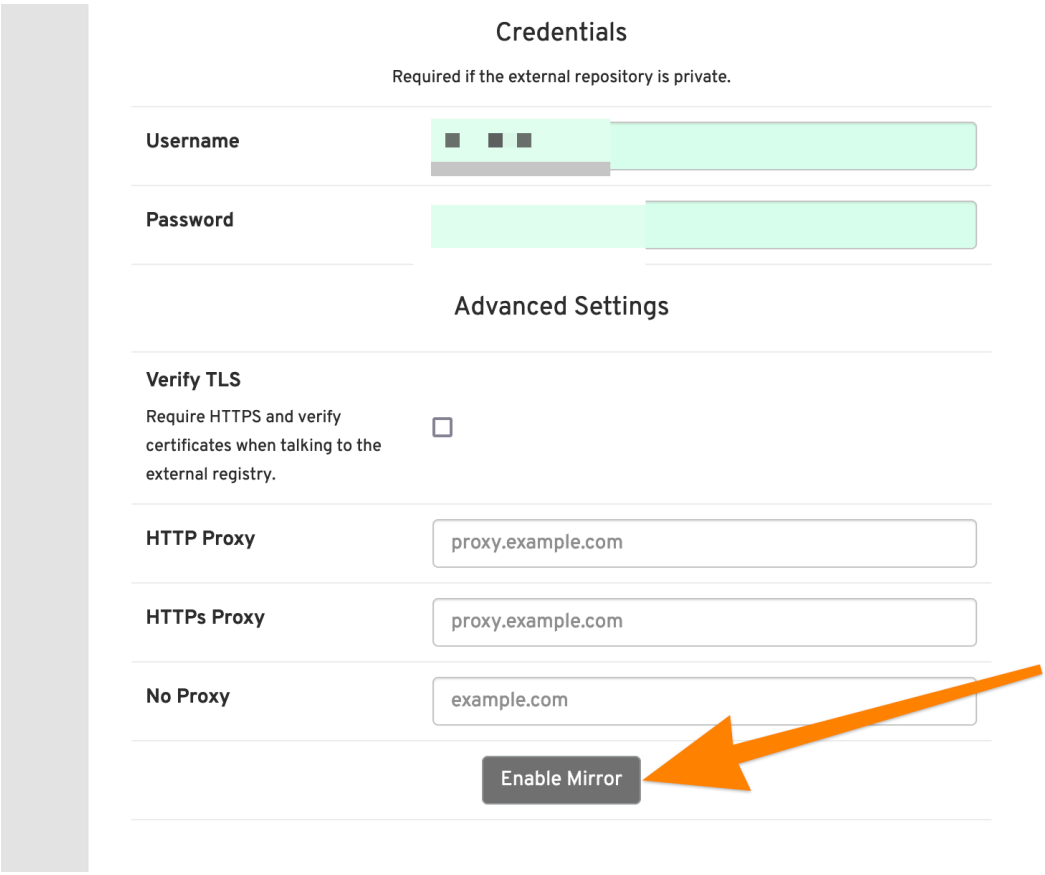
5. In the **Mirroring** tab, complete the required information for the repository and create a **Robot User**. Click **Enable Mirror**

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

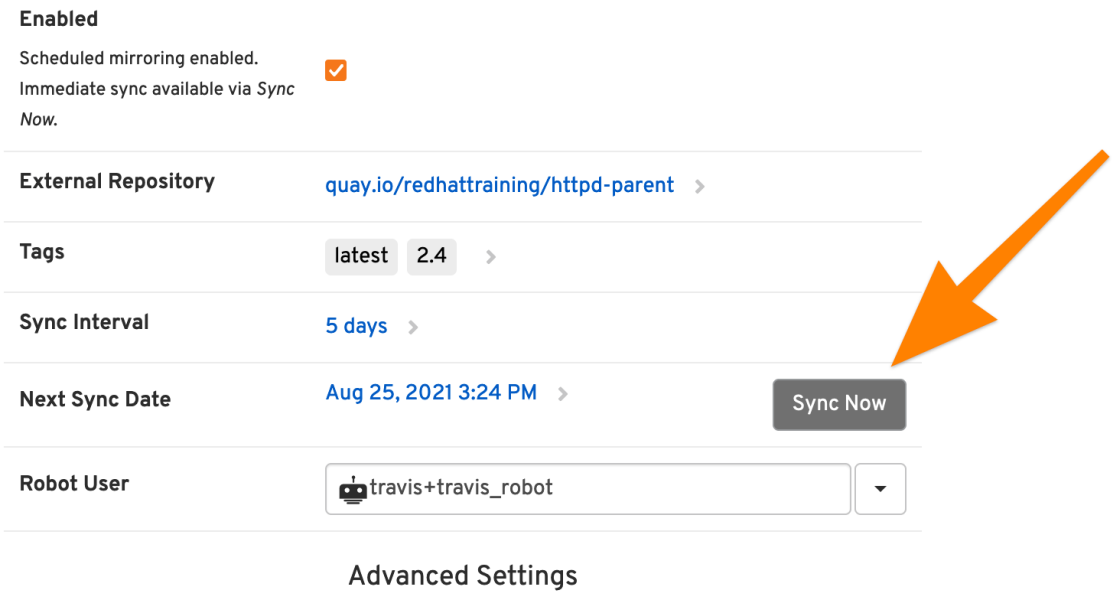
b. Tags: latest and 2.4



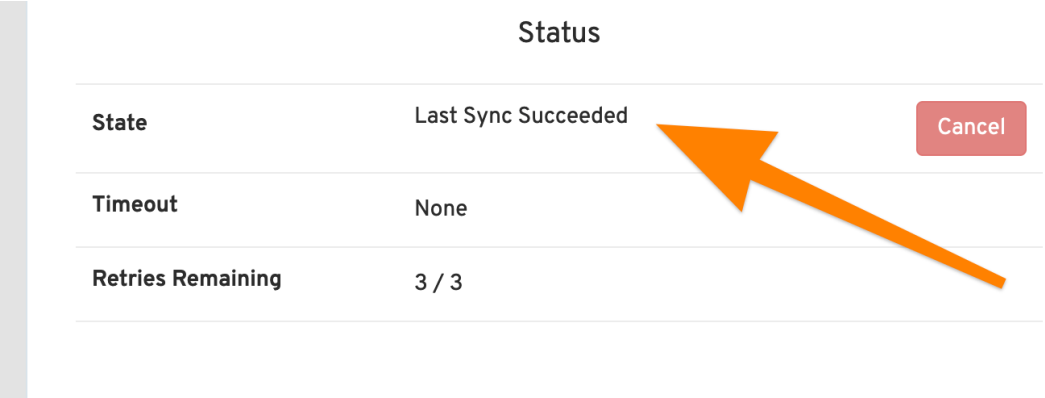


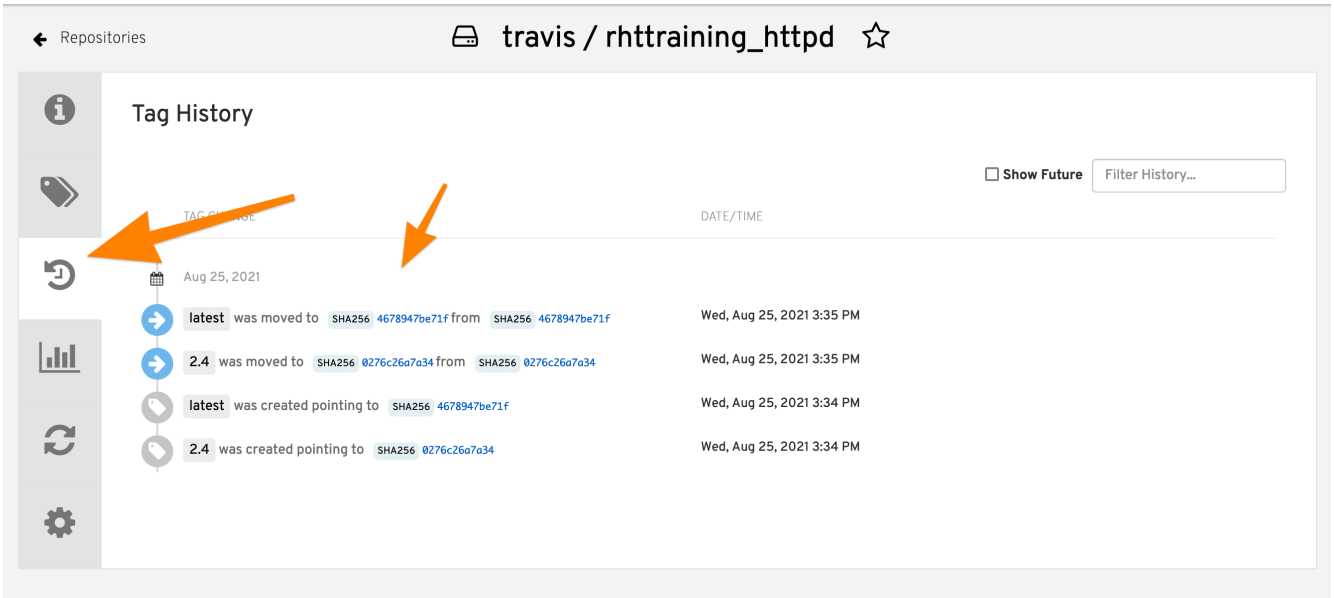


6. Click "**Sync Now**" to perform immediate synchronization



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





7.3. Using the Quay Image Registry

7.4. Inspecting Images with Skopeo on Remote Registries

References

Deploy Red Hat Quay for proof-of-concept (non-production) purposes: https://access.redhat.com/documentation/en-us/red_hat_quay/3/html/deploy_red_hat_quay_for_proof-of-concept_non-production_purposes/index Deploying Quay as a local registry server using local storage and containerized services.

Quay Lab PoC: https://github.com/tmichett/quay_lab_poc Deploying the Quay Registry locally based on the Proof-of-Concept local deployment.

Appendix A: System Security Policy and Compliance

System security and compliance is a primary concern for people when thinking about the protection of systems and integrity of data. This set of hands-on procedures will focus on obtaining the content and necessary packages to perform a basic scan and remediate the system based on scan results. As part of the lab, you will be customizing your own SCAP content for a scan, view the results, and generate an Ansible playbook based on the failed results.

A.1. Customizing SCAP Content

Red Hat includes the SCAP Workbench application as a GUI application which allows scanning, customizing, and saving SCAP scans and results. The SCAP Workbench can be used to perform scans of remote systems over an SSH connection or it can be utilized to scan the local system. Since the SCAP Workbench is a GUI application, it must run on a system with X-Windows installed.

Servers to Configure

- workstation

Packages to Install

- scap-workbench

Since we are using an application on a VM that requires a GUI, we can use X11 forwarding with the SSH connection by specifying a **-X** on the command line.

Step 1 - SSH to Workstation

The fist step is to connect to the workstation and forward X11 traffic back to the local system.

Example 3. Connecting to the Workstation VM

Listing 69. Connecting to Workstation Using SSH

```
# ssh root@workstation -X
```

Step 2 - Install Packages

After connecting to the Workstation VM, you will need to install the SCAP Workbench and its dependencies.

Example 4. Installing SCAP Workbench

Listing 70. Using Yum to Install SCAP Workbench

```
# yum install scap-workbench

Loaded plugins: langpacks, search-disabled-repos
Resolving Dependencies
--> Running transaction check
---> Package scap-workbench.x86_64 0:1.1.6-1.el7 will be installed
--> Processing Dependency: openscap-utils >= 1.2.0 for package: scap-workbench-1.1.6-1.el7.x86_64
--> Processing Dependency: scap-security-guide for package: scap-workbench-1.1.6-1.el7.x86_64
--> Running transaction check
---> Package openscap-utils.x86_64 0:1.2.16-8.el7_5 will be installed
--> Processing Dependency: openscap-containers = 1.2.16-8.el7_5 for package: openscap-utils-1.2.16-8.el7_5.x86_64
---> Package scap-security-guide.noarch 0:0.1.36-9.el7_5 will be installed
--> Processing Dependency: openscap-scanner >= 1.2.5 for package: scap-security-guide-0.1.36-9.el7_5.noarch
--> Running transaction check
---> Package openscap-containers.noarch 0:1.2.16-8.el7_5 will be installed
---> Package openscap-scanner.x86_64 0:1.2.16-8.el7_5 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                Arch      Version             Repository           Size
================================================================================
Installing:
 scap-workbench         x86_64    1.1.6-1.el7         rhel--server-dvd    1.8 M
Installing for dependencies:
 openscap-containers    noarch    1.2.16-8.el7_5      rhel_updates         27 k
 openscap-scanner       x86_64    1.2.16-8.el7_5      rhel_updates         61 k
 openscap-utils         x86_64    1.2.16-8.el7_5      rhel_updates         27 k
 scap-security-guide    noarch    0.1.36-9.el7_5      rhel_updates        2.6 M

Transaction Summary
================================================================================
Install  1 Package (+4 Dependent packages)

Total download size: 4.5 M
Installed size: 64 M
Is this ok [y/d/N]:
```

Some things might already be installed for you, if SCAP Workbench is already installed, please move on to the next step. Also note the dependencies for SCAP Workbench as they are automatically installed.

Step 3 - Launching SCAP Workbench

In order to run a scan or customize SCAP content, you will need to launch the SCAP Workbench application.

You **must** use SCAP Workbench from a GUI, so it will need to run either locally or through an SSH connection with X11 forwarded.

Example 5. Launching SCAP Workbench

Listing 71. Using SCAP Workbench

```
# scap-workbench
```

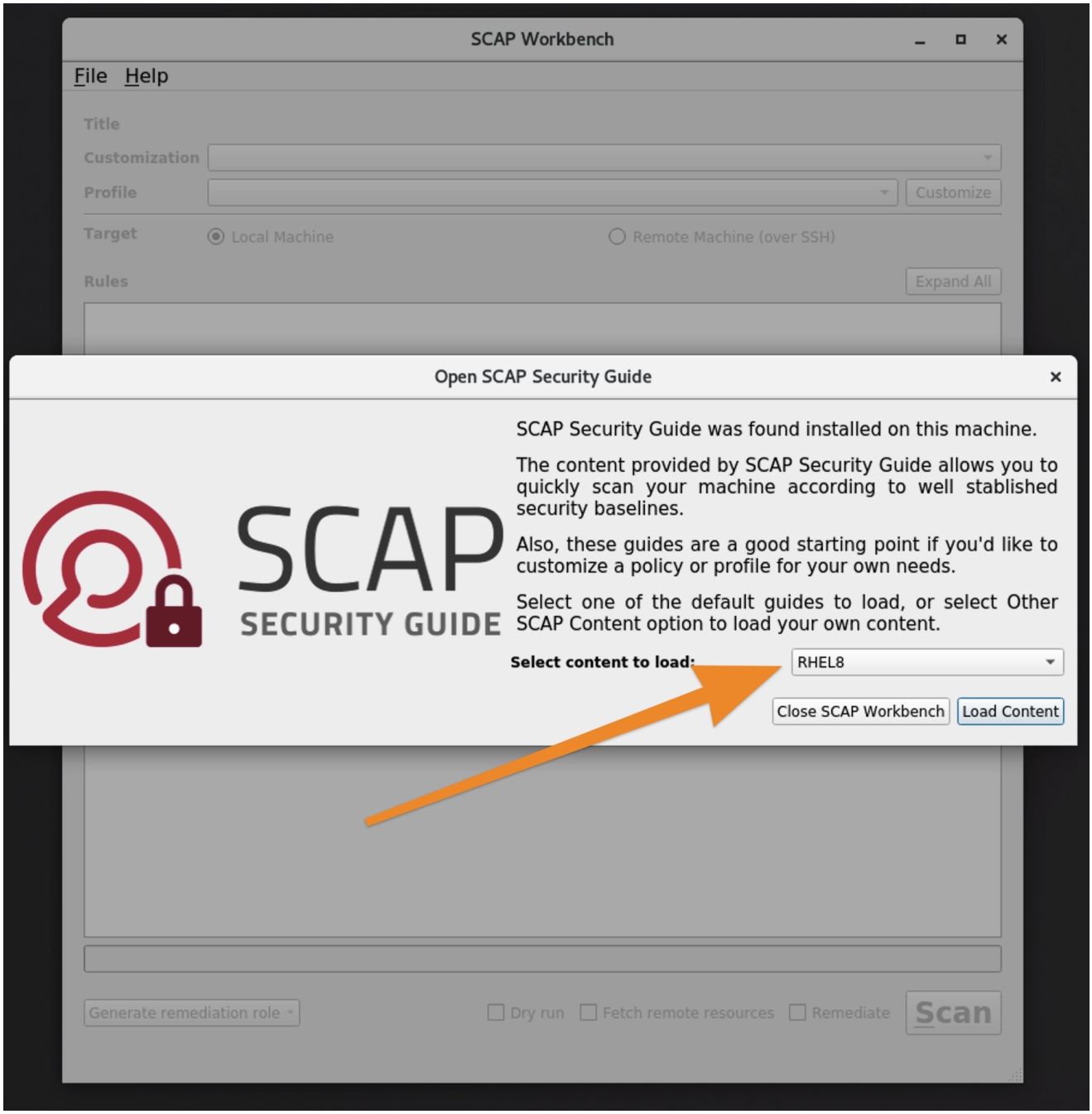


Figure 36. SCAP Workbench Startup

Step 4 - Creating Custom Content

Once SCAP Workbench has been launched, select the content to load. For this lab, we will be using the RHEL7 content.

Example 6. Creating Custom Content

1. For **Select content to load:** select "**RHEL8**", then click "**Load Content**"
2. Select the **Profile** you want to use to start customization

For this example, we will use the **OSPP - Protection Profile for General Purpose Operating Systems** Baseline

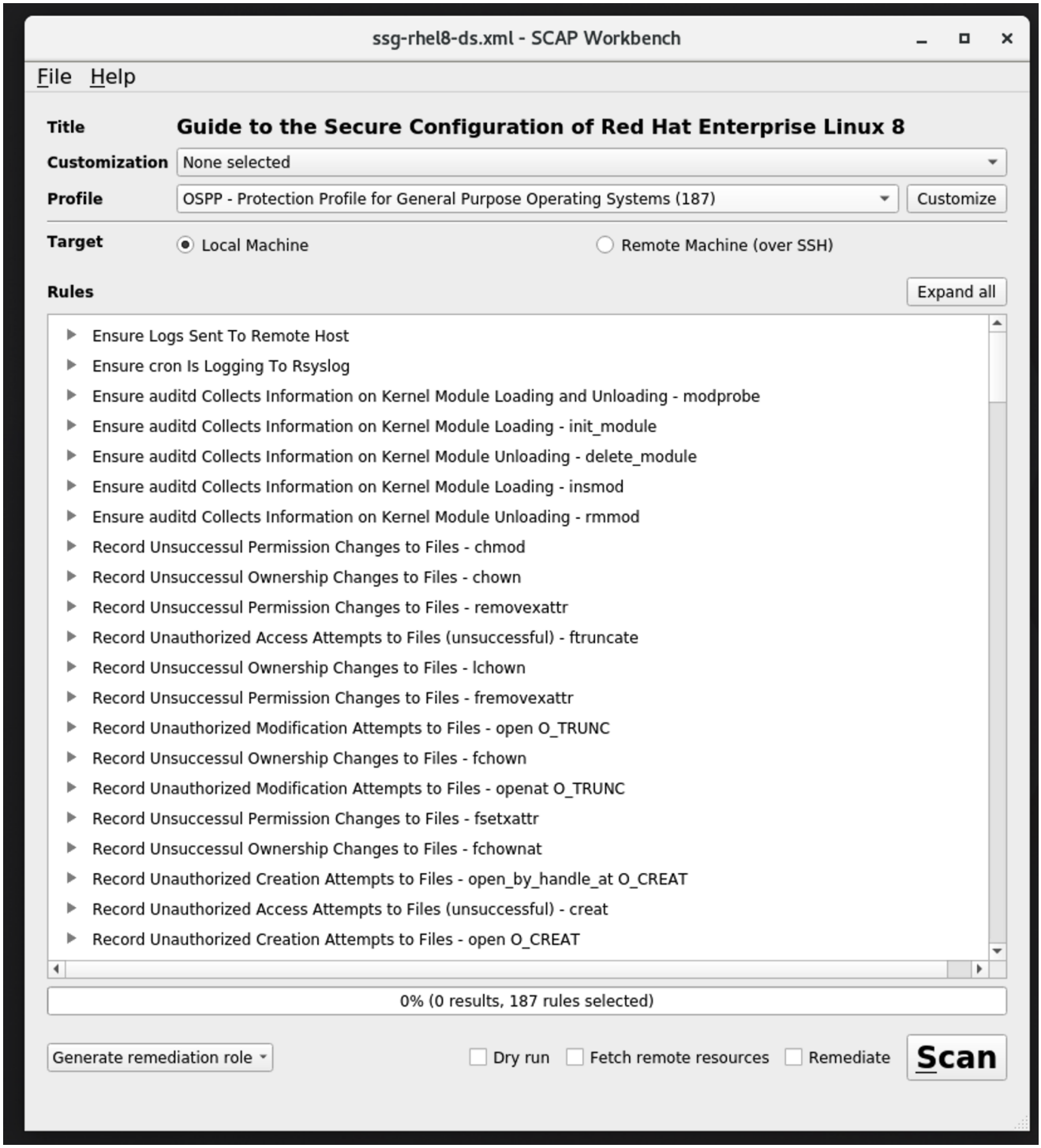


Figure 37. SCAP Workbench OSPP Profile

3. Click "**Customize**" to create custom SCAP content based on the chosen profile, and give it a name.

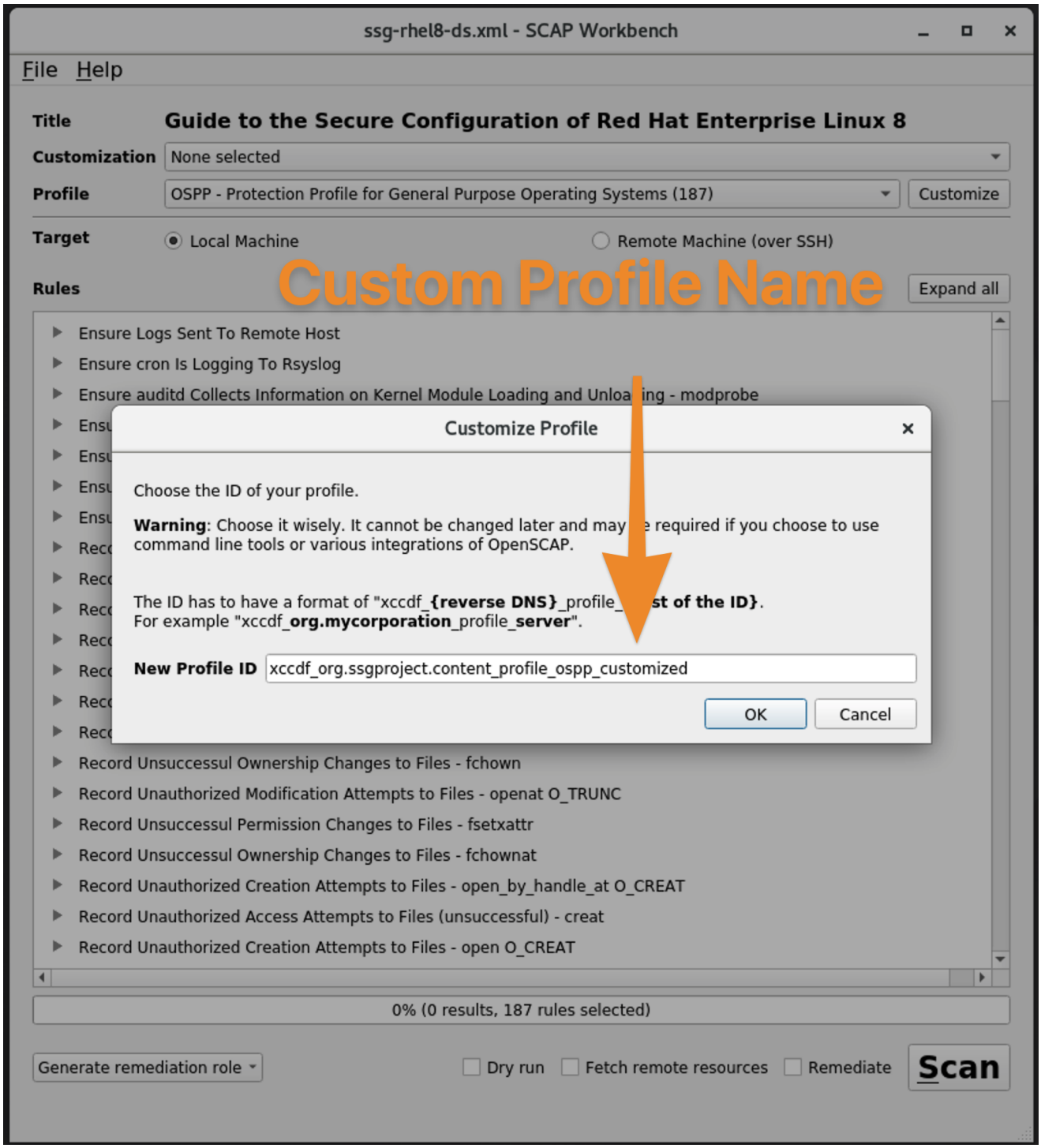


Figure 38. SCAP Custom OSPP Profile Creation

The name for this is: **xccdf_org.ssgproject.content_profile_ospp_customized**

4. Click "Deselect All" so that you can select the items you wish to include in your custom scan profile. **NOTE:** we are doing this to also limit it to a few checks for the example.

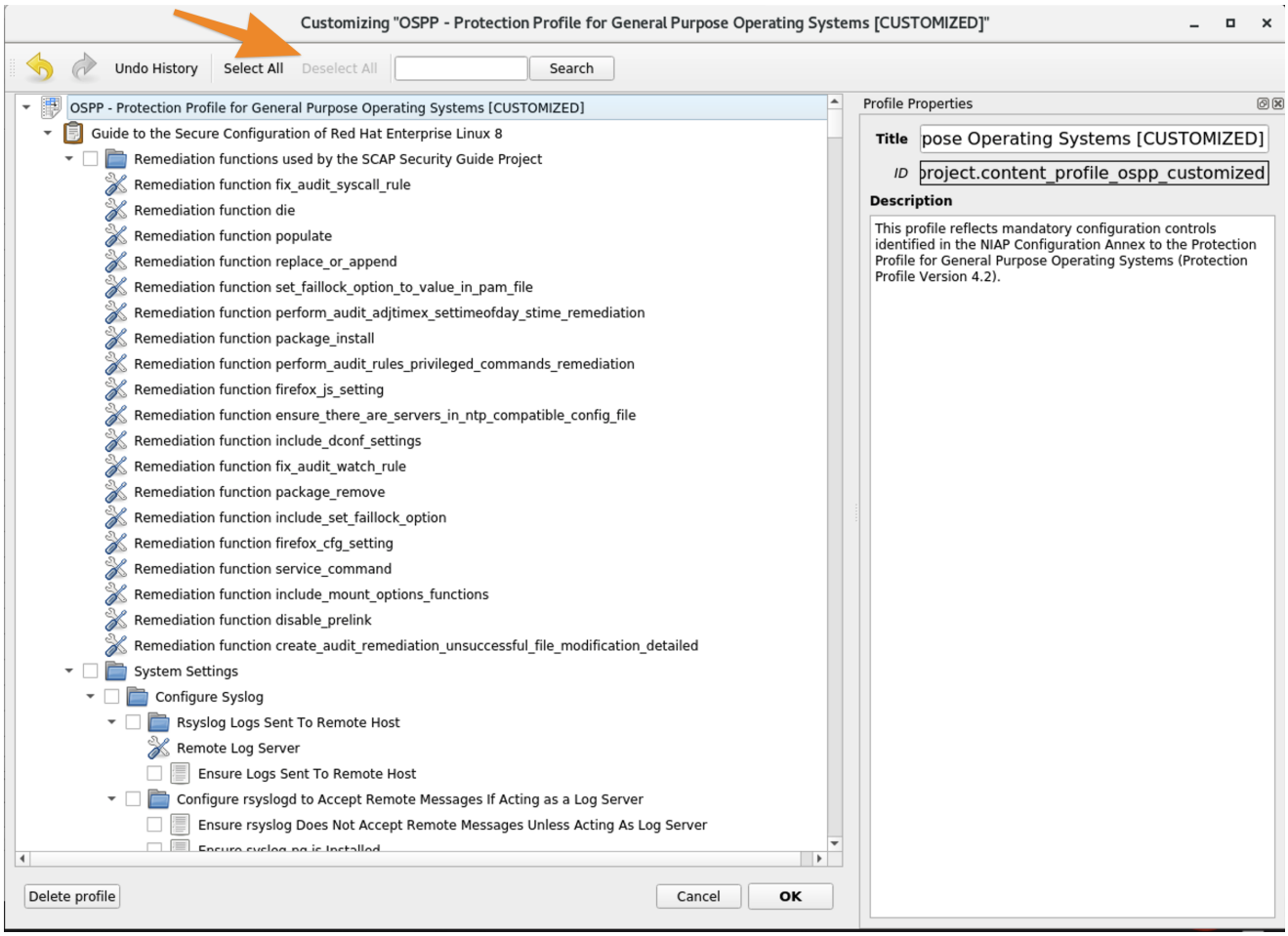


Figure 39. SCAP Custom Profile Selections

For this lab, we will be setting the minimum password length and PAM Password quality settings

5. Search for Password to set **minimum password length** and set the values in **login.defs**. Check **Set Password Minimum Length in login.defs** and click on the **minimum password legnth** and set the value to **18**

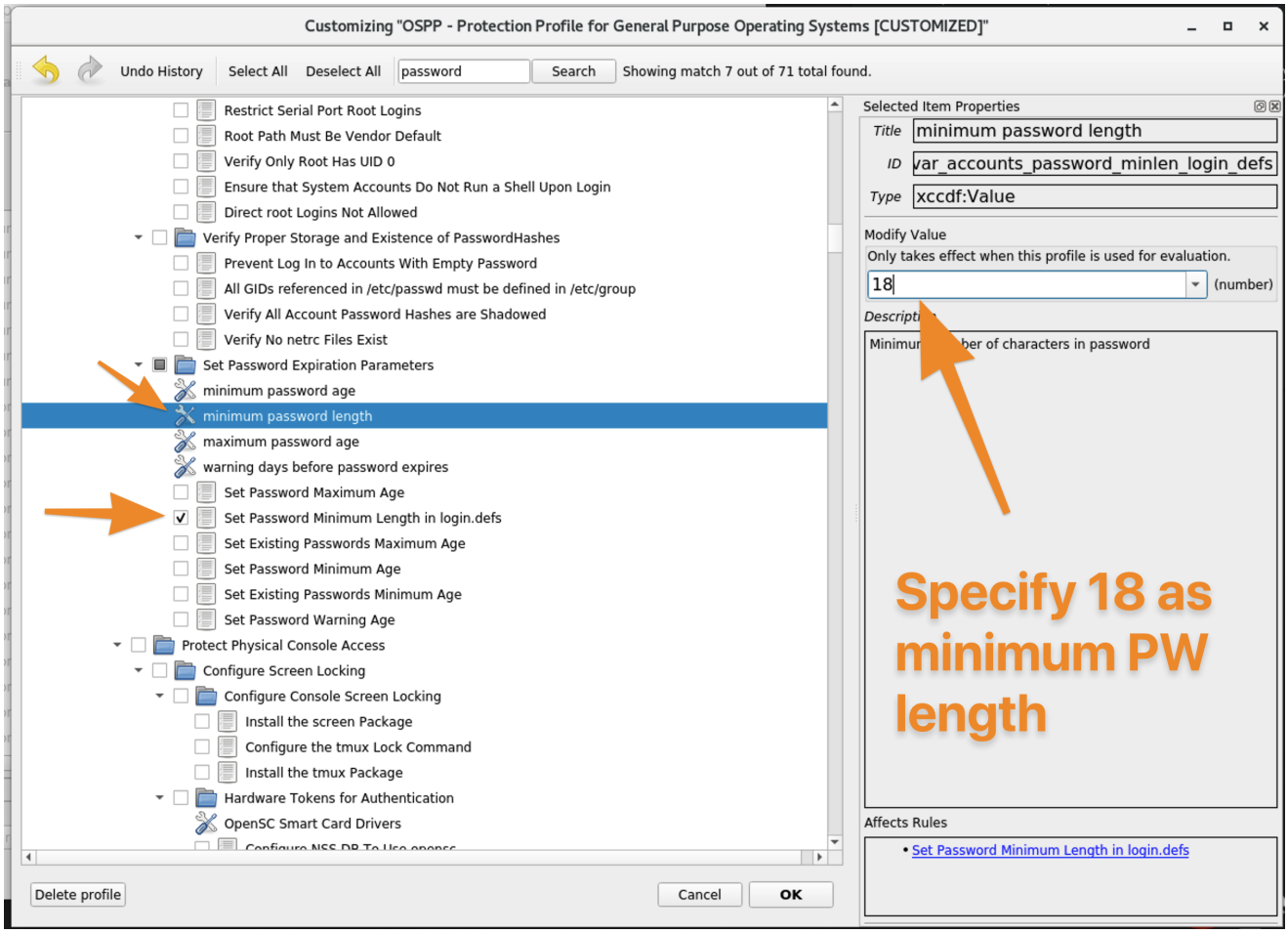


Figure 40. SCAP Custom Profile Password Settings for Login.Defs

6. Set password quality requirements with PAM. Search for the minlen and set it to **18**. Also, place a checkbox in **Set Password Quality Requirements with pam_quality**. Then click "OK"

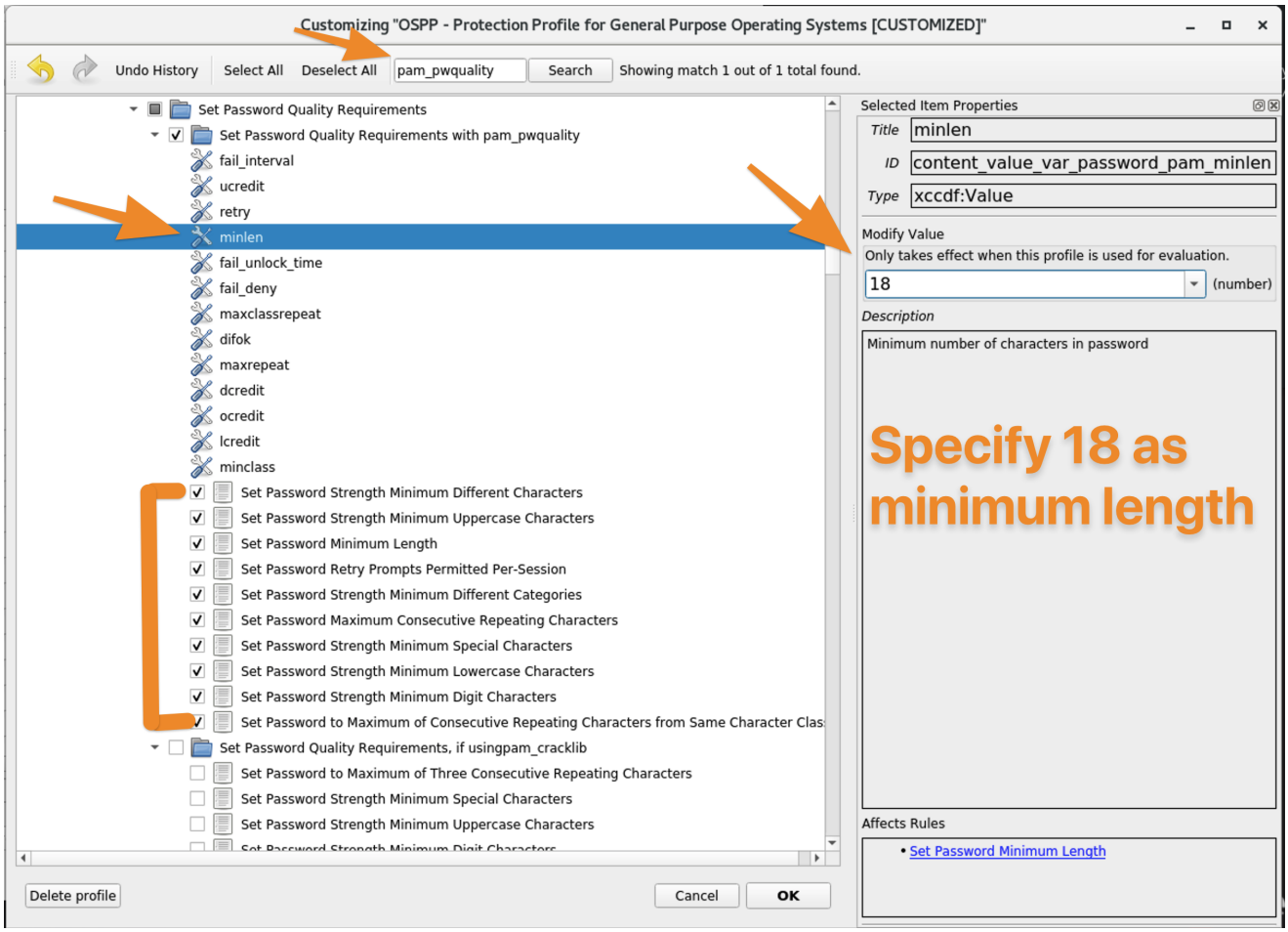


Figure 41. SCAP Custom Profile PAM Quality Requirements

7. At this point, we have taken the default settings from the OSPP profile with only the tailored pieces that we selected. The next step is to click "File ⇒ Save Customization Only" to save the custom content

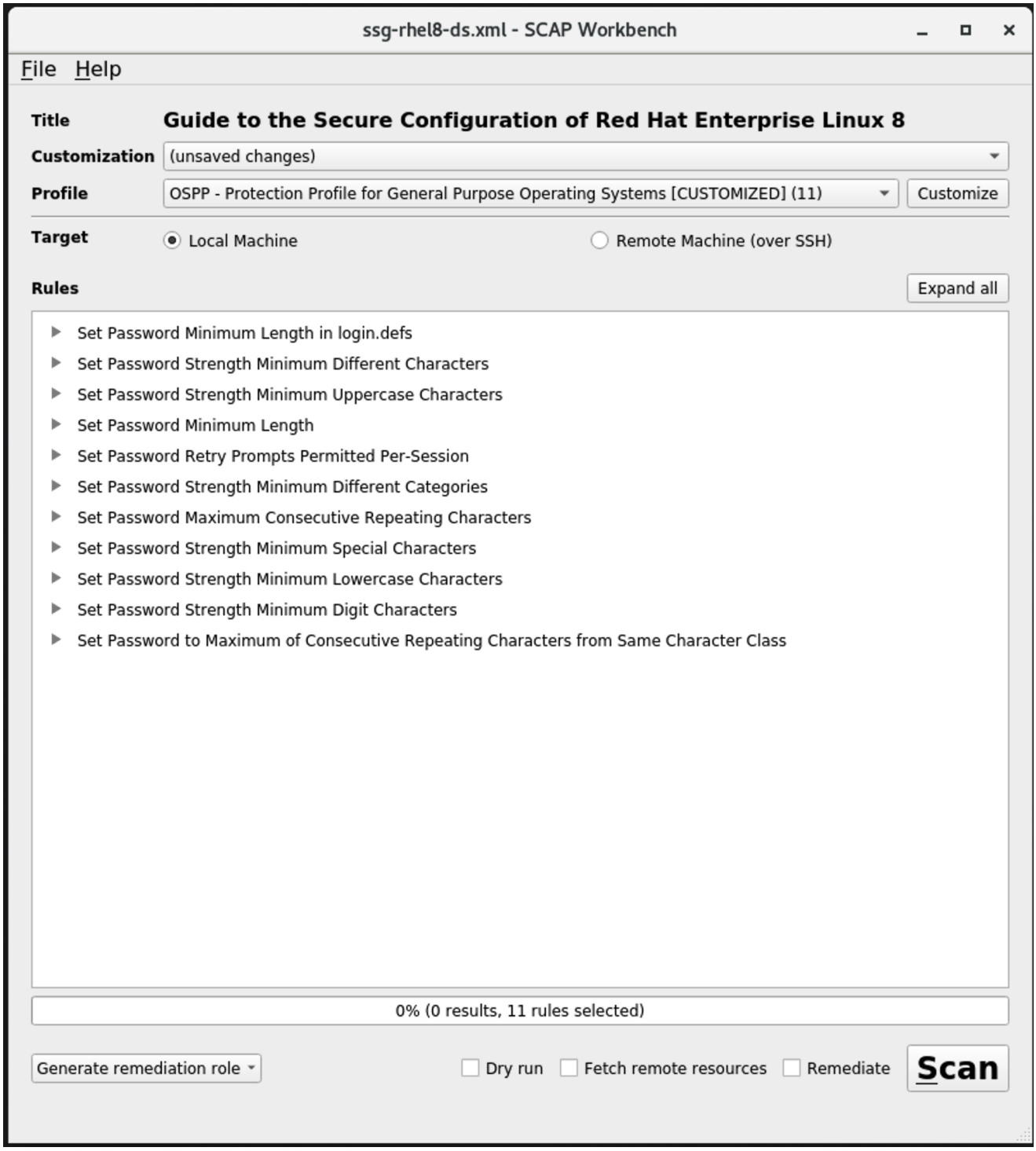


Figure 42. SCAP Custom Profile Selected Settings View

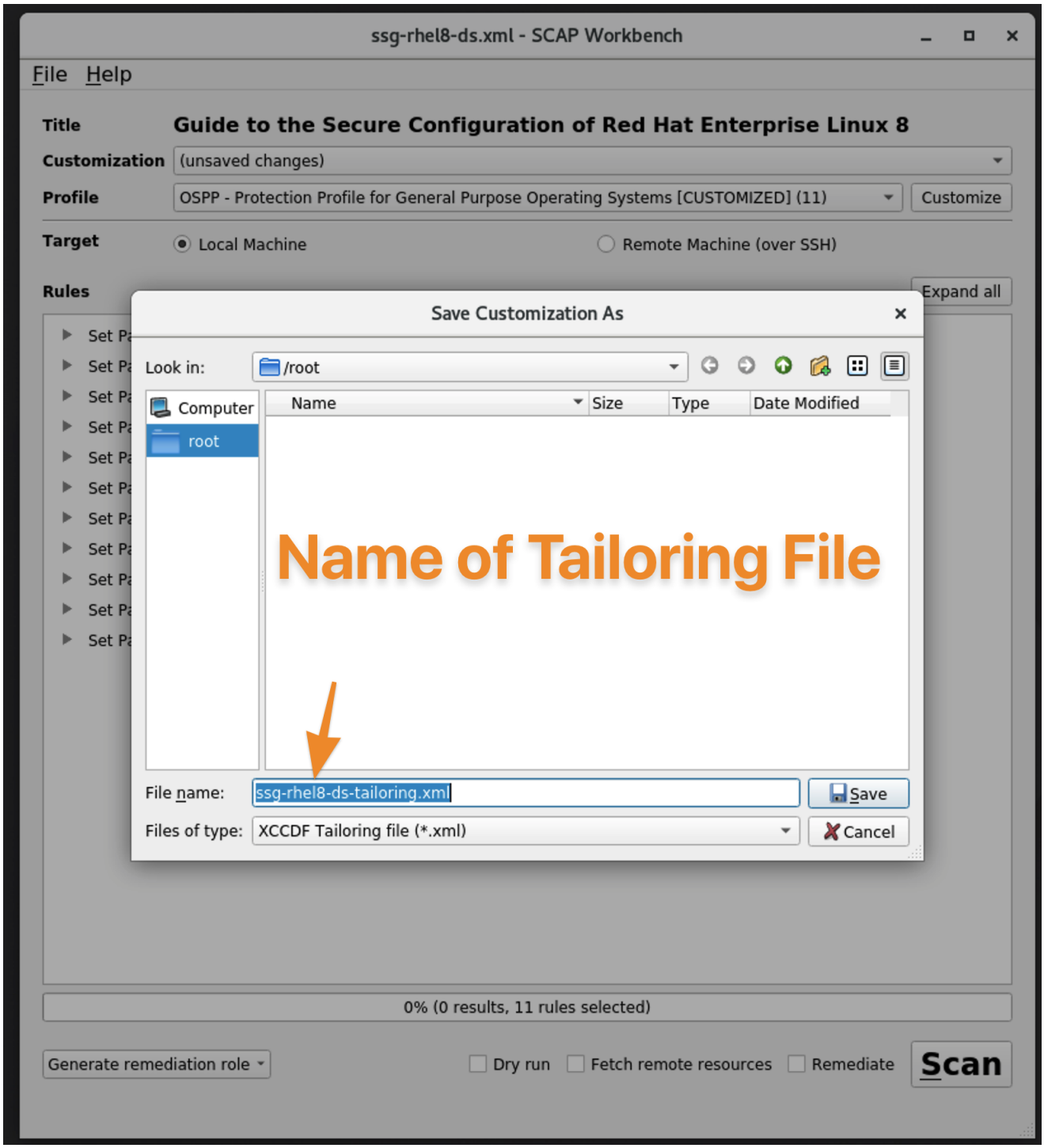


Figure 43. SCAP Custom Profile Creation Saving

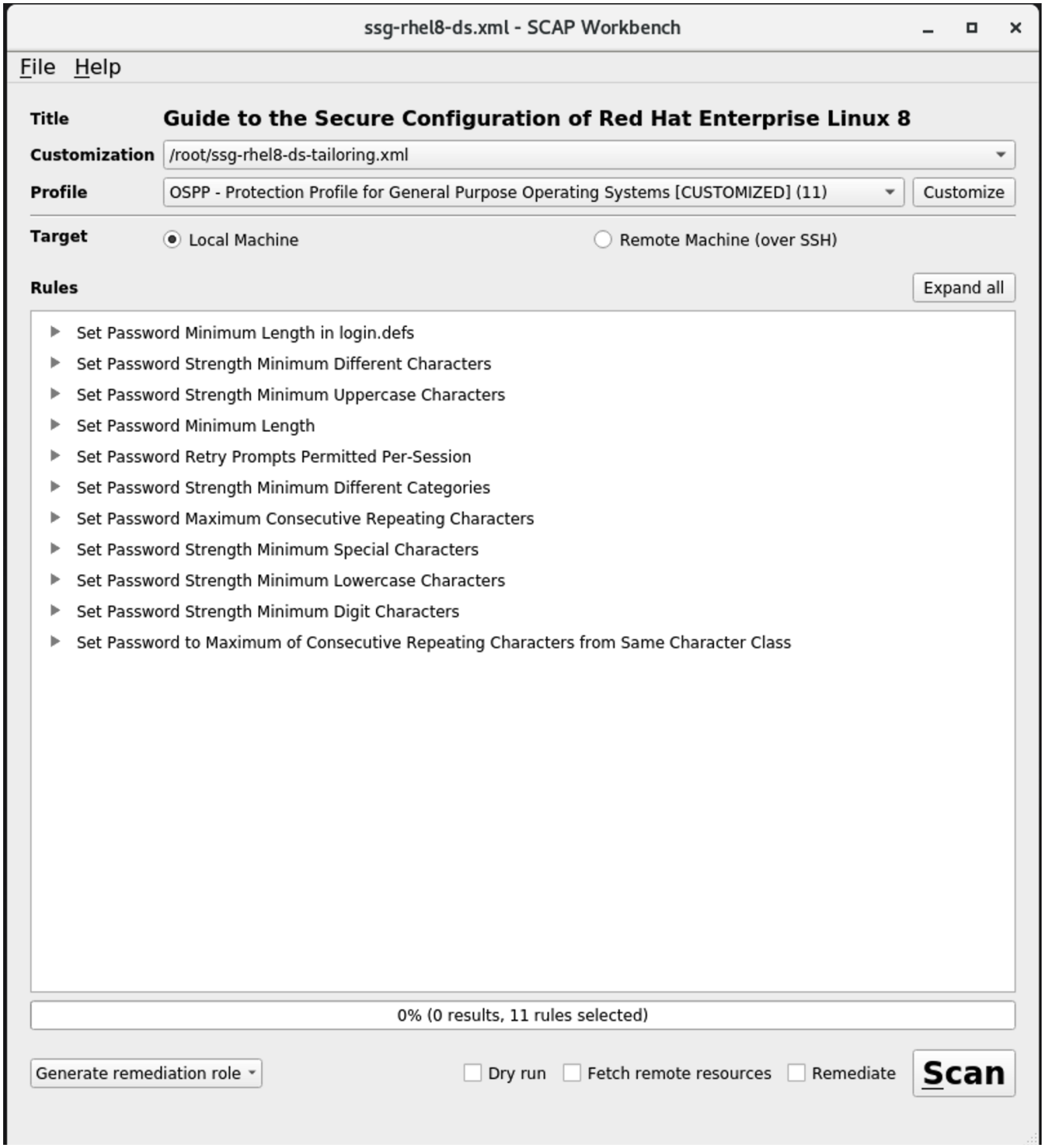


Figure 44. SCAP Custom Profile Final View

8. Copy the custom tailoring file to the server(s) being scanned. In this case, we will want to copy the file to **servera**

Listing 72. Copy custom content

```
[root@workstation ~]# scp ssg-rhel8-ds-tailoring.xml root@servera:
ssg-rhel8-ds-tailoring.xml                    100%   28KB  10.7MB/s   00:00
[root@workstation ~]#
```

A.2. Running a SCAP Scan with Custom Content

Servers to Configure

- servera

Packages to Install

- openscap-scanner
- scap-security-guide

Step 1 - SSH to serverc

The fist step is to connect to the server.

Example 7. Connecting to the **servera** *VM*

Listing 73. Connecting to **servera** *Using SSH*

```
# ssh root@servera
```

Step 2 - Install packages on servera

The second step is to install software on the server.

Example 8. Install software on **servera**

Listing 74. Installing Software on **servera**

```
[root@servera ~]# yum install scap-security-guide
Last metadata expiration check: 0:47:44 ago on Thu 16 Apr 2020 08:26:15 AM EDT.
Dependencies resolved.
================================================================================
 Package          Arch   Version       Repository                          Size
================================================================================
Installing:
 scap-security-guide
                  noarch 0.1.42-11.el8 rhel-8.0-for-x86_64-appstream-rpms 3.4 M
Installing dependencies:
 openscap         x86_64 1.3.0-7.el8   rhel-8.0-for-x86_64-appstream-rpms 3.3 M
 openscap-scanner x86_64 1.3.0-7.el8   rhel-8.0-for-x86_64-appstream-rpms  66 k
 xml-common       noarch 0.6.3-50.el8  rhel-8.0-for-x86_64-baseos-rpms     39 k

Transaction Summary
================================================================================
Install  4 Packages

Total download size: 6.9 M
Installed size: 132 M
Is this ok [y/N]: y


... output omitted ...

  Verifying        : openscap-1.3.0-7.el8.x86_64                              1/4
  Verifying        : openscap-scanner-1.3.0-7.el8.x86_64                      2/4
  Verifying        : scap-security-guide-0.1.42-11.el8.noarch                 3/4
  Verifying        : xml-common-0.6.3-50.el8.noarch                           4/4

Installed:
  scap-security-guide-0.1.42-11.el8.noarch    openscap-1.3.0-7.el8.x86_64
  openscap-scanner-1.3.0-7.el8.x86_64         xml-common-0.6.3-50.el8.noarch

Complete!
```

Learning about SCAP Commands

The SSG man page is a very good source of information for usage of the **oscap** tool as well as provides examples of how to use the SCAP SSG Guide profiles itself.

Listing 75. Looking at **SCAP Security Guide (SSG)** *Man Page*

```
# man scap-security-guide

scap-security-guide(8)      System Manager's Manual     scap-security-guide(8)

NAME
       SCAP  Security Guide - Delivers security guidance, baselines, and asso-
       ciated validation mechanisms utilizing the Security Content  Automation
       Protocol (SCAP).


... output omitted ...

EXAMPLES
       To  scan  your  system utilizing the OpenSCAP utility against the ospp-
       rhel7 profile:

       oscap xccdf eval --profile  ospp-rhel7  --results  /tmp/`hostname`-ssg-
       results.xml  --report  /tmp/`hostname`-ssg-results.html  --oval-results
       /usr/share/xml/scap/ssg/content/ssg-rhel7-xccdf.xml
```

Listing 76. Looking at ***oscap*** *Man Page*

```
# man oscap

OSCAP(8)                System Administration Utilities               OSCAP(8)

NAME
       oscap - OpenSCAP command line tool

SYNOPSIS
       oscap  [general-options]  module operation [operation-options-and-argu-
       ments]

DESCRIPTION
       oscap is Security Content Automation Protocol (SCAP) toolkit  based  on
       OpenSCAP  library.  It  provides  various  functions for different SCAP
       specifications (modules).

       OpenSCAP tool claims to provide capabilities of Authenticated  Configu-
       ration  Scanner  and  Authenticated Vulnerability Scanner as defined by
       The National Institute of Standards and Technology.

... output omitted ...

EXAMPLES
       Evaluate XCCDF content using CPE dictionary and produce html report. In
       this  case  we  use  United  States  Government  Configuration Baseline
       (USGCB) for Red Hat Enterprise Linux 5 Desktop.

              oscap xccdf eval --fetch-remote-resources --oval-results \
                      --profile united_states_government_configuration_baseline
 \
                      --report usgcb-rhel5desktop.report.html \
                      --results usgcb-rhel5desktop-xccdf.xml.result.xml \
                      --cpe usgcb-rhel5desktop-cpe-dictionary.xml \
                      usgcb-rhel5desktop-xccdf.xml
```

Step 3 - Running *oscap* scan

We will run the **oscap** utility to generate a report and a results file that can be sent back to the **workstation** system so that we can create an Ansible playbook for remediation and view the results of the report.

Be very careful about the name of the profile as this was selected during the creation of the custom profile/tailoring file portion when doing SCAP Workbench customizations.

Example 9. Scanning **servera**

Listing 77. Using **oscap** *and the tailoring profile to scan* **servera**

```
# [root@servera ~]# oscap xccdf eval \
--profile xccdf_org.ssgproject.content_profile_ospp_customized \
--tailoring-file ssg-rhel8-ds-tailoring.xml \
--results custom_scan_results.xml \
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml

Title   Set Password Minimum Length in login.defs
Rule    xccdf_org.ssgproject.content_rule_accounts_password_minlen_login_defs
Ident   CCE-80652-1
Result  fail

Title   Set Password Strength Minimum Different Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_difok
Ident   CCE-80654-7
Result  fail

Title   Set Password Strength Minimum Uppercase Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_ucredit
Ident   CCE-80665-3
Result  fail

Title   Set Password Minimum Length
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_minlen
Ident   CCE-80656-2
Result  fail

Title   Set Password Retry Prompts Permitted Per-Session
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_retry
Ident   CCE-80664-6
Result  fail

Title   Set Password Strength Minimum Different Categories
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_minclass
Result  fail

Title   Set Password Maximum Consecutive Repeating Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_maxrepeat
Result  fail

Title   Set Password Strength Minimum Special Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_ocredit
Ident   CCE-80663-8
Result  fail

Title   Set Password Strength Minimum Lowercase Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_lcredit
Ident   CCE-80655-4
Result  fail

Title   Set Password Strength Minimum Digit Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_dcredit
Ident   CCE-80653-9
Result  fail

Title   Set Password to Maximum of Consecutive Repeating Characters from Same Character Class
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_maxclassrepeat
Result  fail
```

Getting Custom Profile Name from Tailoring File

If you need to locate the profile used for the custom scanning content from the tailoring file, you can search for it with **grep**.

```
[root@servera ~]#  grep "Profile id" ssg-rhel8-ds-tailoring.xml
  <xccdf:Profile id="xccdf_org.ssgproject.content_profile_ospp_customized" extends
="xccdf_org.ssgproject.content_profile_ospp">
```

Step 4 - Creating a Results Report

You can create a results report file from the results file so you have a nice HTML file that is easy to ready with the results from the SCAP scan.

Example 10. Creating a SCAP Report from a Results File

Listing 78. Generating a Report

```
[root@servera ~]# oscap xccdf generate report \
custom_scan_results.xml > Custom_Scan_Report.html
```

Combining Steps 3 & 4

It is possible to perform a custom content scan which will generate the results file and the report for transfer back to the workstation for review.

Need to Specify

- **--results**
- **--report**

Listing 79. Creating a Results File and Report During Custom Content Scan

```
[root@servera ~]# oscap xccdf eval \
--profile xccdf_org.ssgproject.content_profile_ospp_customized \
--tailoring-file ssg-rhel8-ds-tailoring.xml \
--results custom_scan_results_2.xml \
--report Custom_Scan_Report_2.html \
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml

Title   Set Password Minimum Length in login.defs
Rule    xccdf_org.ssgproject.content_rule_accounts_password_minlen_login_defs
Ident   CCE-80652-1
Result  fail

Title   Set Password Strength Minimum Different Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_difok
Ident   CCE-80654-7
Result  fail

Title   Set Password Strength Minimum Uppercase Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_ucredit
Ident   CCE-80665-3
Result  fail

Title   Set Password Minimum Length
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_minlen
Ident   CCE-80656-2
Result  fail

Title   Set Password Retry Prompts Permitted Per-Session
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_retry
Ident   CCE-80664-6
Result  fail

Title   Set Password Strength Minimum Different Categories
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_minclass
Result  fail

Title   Set Password Maximum Consecutive Repeating Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_maxrepeat
Result  fail

Title   Set Password Strength Minimum Special Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_ocredit
Ident   CCE-80663-8
Result  fail

Title   Set Password Strength Minimum Lowercase Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_lcredit
Ident   CCE-80655-4
Result  fail

Title   Set Password Strength Minimum Digit Characters
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_dcredit
Ident   CCE-80653-9
Result  fail

Title   Set Password to Maximum of Consecutive Repeating Characters from Same Character Class
Rule    xccdf_org.ssgproject.content_rule_accounts_password_pam_maxclassrepeat
Result  fail
```

Step 5 - Transferring Results File and Report to Workstation

After you have the results files and the report, you should transfer it to your graphical workstation (**workstation**) for further analysis.

Example 11. Transferring Results

Listing 80. Transferring the Results and Report Files

```
[root@servera ~]# scp *.xml *.html root@workstation:
The authenticity of host 'workstation (<no hostip for proxy command>)' can't be established.
ECDSA key fingerprint is SHA256:pOQ1OJmyF2PFI+jxyFoOSCfi+1oWNsUruy2DZNjg+N0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'workstation' (ECDSA) to the list of known hosts.
root@workstation's password:
custom_scan_results_2.xml                     100% 4086KB  32.4MB/s   00:00
custom_scan_results.xml                       100% 4086KB  60.0MB/s   00:00
ssg-rhel8-ds-tailoring.xml                    100%   28KB  16.2MB/s   00:00
Custom_Scan_Report_2.html                     100%  332KB  44.3MB/s   00:00
Custom_Scan_Report.html                       100%  332KB  37.6MB/s   00:00
[root@servera ~]#
```

Step 6 - Viewing the SCAP scan report

After you have transferred the results file to **workstation** you can open the HTML report in a web browser. In this case we will use ***firefox*** to open the file.

Example 12. Viewing the SCAP Report

Listing 81. Opening the SCAP HTML Report with Firefox

```
[root@workstation ~]# firefox Custom_Scan_Report.html
```

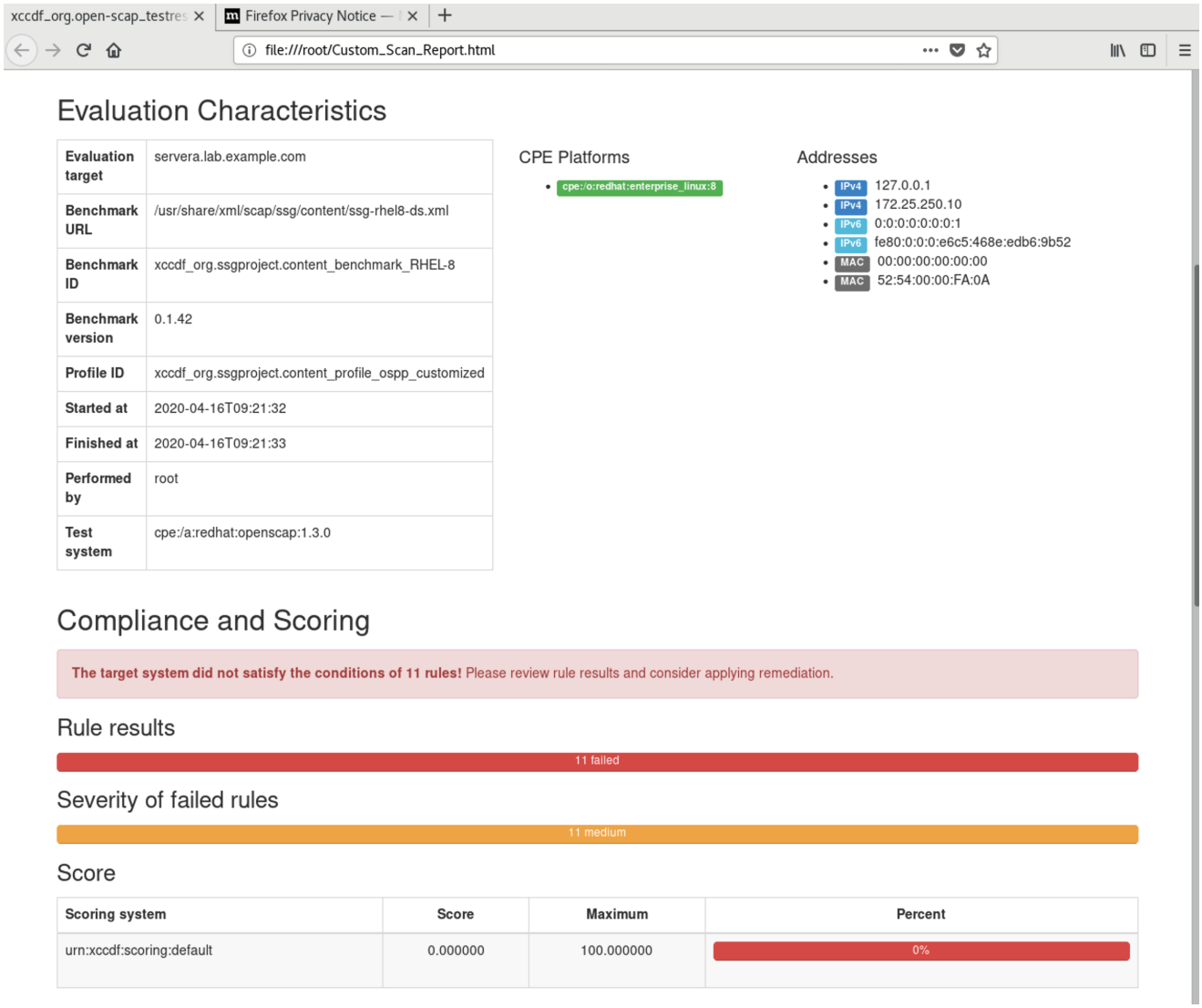


Figure 45. SCAP Scan Results Report in Firefox

Firefox may not open the file based on SELinux context triggers. In order to get around this you can use the command prompt and do **setenforce 0** to allow you to open the report.

A.3. Creating an Ansible Remediation Playbook Based on SCAP Scan Results

The OpenSCAP project and content created by Red Hat can automatically remediate findings from OpenSCAP scans. The findings can be remediated in many ways (**BASH**, **Ansible**, etc.). While things are mostly complete, there are some automated remediations that have not yet been developed.

There are multiple automatic remediation methods developed, but at this time, there isn't a script to fix everything.

Servers to Configure

- servera

We will continue to use **workstation** as our master SCAP system as it should have Ansible and SCAP Workbench installed.

Step 1 - Creating an Ansible Playbook from Results

The first step will be to generate an Ansible playbook from the SCAP scan results for system remediation.

Example 13. Generating Ansible Playbook

Listing 82. Ansible Playbook Generation

```
[root@workstation ~]# oscap xccdf generate fix \
--profile xccdf_org.ssgproject.content_profile_ospp_customized \
--tailoring-file ssg-rhel8-ds-tailoring.xml \
--fix-type ansible \
--result-id "" \
custom_scan_results.xml > Custom_Scan_Fix.yml
```

Viewing Remediation Playbook

It is also a good idea to view the created playbook for the system prior to running it.

```
[root@workstation ~]# cat Custom_Scan_Fix.yml
---
#################################################################################
#
# Ansible remediation role for the results of evaluation of profile
xccdf_org.ssgproject.content_profile_ospp_customized
# XCCDF Version:  unknown
#
# Evaluation Start Time:  2020-04-16T09:21:32
# Evaluation End Time:  2020-04-16T09:21:33
#
# This file was generated by OpenSCAP 1.3.0 using:
#   $ oscap xccdf generate fix --result-id xccdf_org.open-
scap_testresult_xccdf_org.ssgproject.content_profile_ospp_customized --template urn:xccdf:fix:script:ansible
xccdf-results.xml
#
# This script is generated from the results of a profile evaluation.
```

```
# It attempts to remediate all issues from the selected rules that failed the test.
#
# How to apply this remediation role:
# $ ansible-playbook -i "localhost," -c local playbook.yml
# $ ansible-playbook -i "192.168.1.155," playbook.yml
# $ ansible-playbook -i inventory.ini playbook.yml
#
################################################################################


 - hosts: all
   vars:
      var_accounts_password_minlen_login_defs: !!str 18
      var_password_pam_difok: !!str 8
      var_password_pam_ucredit: !!str -1
      var_password_pam_minlen: !!str 18
      var_password_pam_retry: !!str 3
      var_password_pam_minclass: !!str 3
      var_password_pam_maxrepeat: !!str 3
      var_password_pam_ocredit: !!str -1
      var_password_pam_lcredit: !!str -1
      var_password_pam_dcredit: !!str -1
      var_password_pam_maxclassrepeat: !!str 4
   tasks:


    - name: "Set Password Minimum Length in login.defs"
      lineinfile:
        dest: /etc/login.defs
        regexp: "^PASS_MIN_LEN *[0-9]*"
        state: present
        line: "PASS_MIN_LEN        {{ var_accounts_password_minlen_login_defs }}"
      tags:
        - accounts_password_minlen_login_defs
        - medium_severity
        - restrict_strategy
        - low_complexity
        - low_disruption
        - CCE-80652-1
        - NIST-800-53-IA-5(f)
        - NIST-800-53-IA-5(1)(a)
        - NIST-800-171-3.5.7
        - CJIS-5.6.2.1


... Output Omitted ...

    - name: Ensure PAM variable maxclassrepeat is set accordingly

      lineinfile:
        create: yes
        dest: "/etc/security/pwquality.conf"
        regexp: '^#?\s*maxclassrepeat'
        line: "maxclassrepeat = {{ var_password_pam_maxclassrepeat }}"

      tags:
        - accounts_password_pam_maxclassrepeat
        - medium_severity
        - restrict_strategy
        - low_complexity
        - low_disruption
        - NIST-800-53-IA-5
        - NIST-800-53-IA-5(c)
```

Ansible is not setup for the lab

Before we can do the next steps, we will download an Ansible config file and an inventory file so we can properly run the playbook.

Listing 83. Error Output Message

```
[root@workstation ~]# ansible-playbook Custom_Scan_Fix.yml
 [WARNING]: provided hosts list is empty, only localhost is available. Note
that the implicit localhost does not match 'all'


PLAY [all] *********************************************************************
skipping: no hosts matched

PLAY RECAP *********************************************************************
[root@workstation ~]#
```

Step 2 - Downloading Ansible Config and Ansible Inventory Files

This step is needed so that our Ansible system can be configured with various configuration options and the inventory files so we can run the given playbook.

Example 14. Downloading Ansible Files

Listing 84. Downloading Ansible Files

```
[root@workstation ~]# wget http://people.redhat.com/~tmichett/rh354/inventory
--2020-04-16 09:38:50--  http://people.redhat.com/~tmichett/rh354/inventory
Resolving people.redhat.com (people.redhat.com)... 209.132.183.19
Connecting to people.redhat.com (people.redhat.com)|209.132.183.19|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24
Saving to: 'inventory'

inventory           100%[===================>]      24  --.-KB/s    in 0s

2020-04-16 09:38:50 (2.69 MB/s) - 'inventory' saved [24/24]


[root@workstation ~]# wget http://people.redhat.com/~tmichett/rh354/ansible.cfg
--2020-04-16 09:39:34--  http://people.redhat.com/~tmichett/rh354/ansible.cfg
Resolving people.redhat.com (people.redhat.com)... 209.132.183.19
Connecting to people.redhat.com (people.redhat.com)|209.132.183.19|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 159
Saving to: 'ansible.cfg'

ansible.cfg         100%[===================>]     159  --.-KB/s    in 0s

2020-04-16 09:39:35 (13.8 MB/s) - 'ansible.cfg' saved [159/159]
```

Reviewing Ansible Configurations

The **inventory** file provided only has a single host **servera** in there. On real systems, you must be very cautious of running remediation playbooks against an inventory file as it could apply to unintended systems. Additionally the **ansible.cfg** file provided was created for use in this lab environment. Both of these items should be taken into account when doing going through the process on production systems.

```
[root@workstation ~]# cat inventory
servera.lab.example.com

[root@workstation ~]# cat ansible.cfg
[defaults]
roles_path = /etc/ansible/roles:/usr/share/ansible/roles
log_path   = /tmp/ansible.log
inventory  = ./inventory

[privilege_escalation]
become=True
[root@workstation ~]#
```

Step 3 - Run the Ansible Playbook

This step will utilize the **workstation** system which is configured as your Ansible management node and will run the playbook to remediate the results on the **servera** system.

*Example 15. Remediation of **serverc** with Ansible Playbook*

Listing 85. Running the Ansible Playbook

```
[root@workstation ~]# ansible-playbook Custom_Scan_Fix.yml

PLAY [all] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [servera.lab.example.com]

TASK [Set Password Minimum Length in login.defs] *******************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable difok is set accordingly] ****************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable ucredit is set accordingly] **************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable minlen is set accordingly] ***************************
changed: [servera.lab.example.com]

TASK [Set Password Retry Prompts Permitted Per-Session - system-auth (change)] ***
ok: [servera.lab.example.com]

TASK [Set Password Retry Prompts Permitted Per-Session - system-auth (add)] ****
changed: [servera.lab.example.com]

TASK [Ensure PAM variable minclass is set accordingly] *************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable maxrepeat is set accordingly] ************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable ocredit is set accordingly] **************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable lcredit is set accordingly] **************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable dcredit is set accordingly] **************************
changed: [servera.lab.example.com]

TASK [Ensure PAM variable maxclassrepeat is set accordingly] *******************
changed: [servera.lab.example.com]

PLAY RECAP *********************************************************************
servera.lab.example.com    : ok=13   changed=11   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

After running the playbook, you can see that there were 10 changes that were made to the system and exactly which parameters were changed. The next thing to do is perform another scan of the system to ensure that it is now fully compliant.

Step 4 - Rescan System and Review Results

Example 16. Scanning System after Fixes and Verifying Results

Listing 86. Performing SCAP Verification Scan

```
[root@servera ~]# oscap xccdf eval \
--profile xccdf_org.ssgproject.content_profile_ospp_customized \
--tailoring-file ssg-rhel8-ds-tailoring.xml \
--results custom_scan_results_fixed.xml \
--report Custom_Scan_Report_Fixed.html \
/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

Listing 87. Copying Results to Workstation

```
[root@servera ~]# scp custom_scan_results_fixed.xml Custom_Scan_Report_Fixed.html workstation:
root@workstation's password:
custom_scan_results_fixed.xml                 100% 4086KB  60.3MB/s   00:00
Custom_Scan_Report_Fixed.html                 100%  282KB  48.4MB/s   00:00
```

Listing 88. Viewing Results on Workstation

```
[root@workstation ~]# firefox Custom_Scan_Report_Fixed.html
```

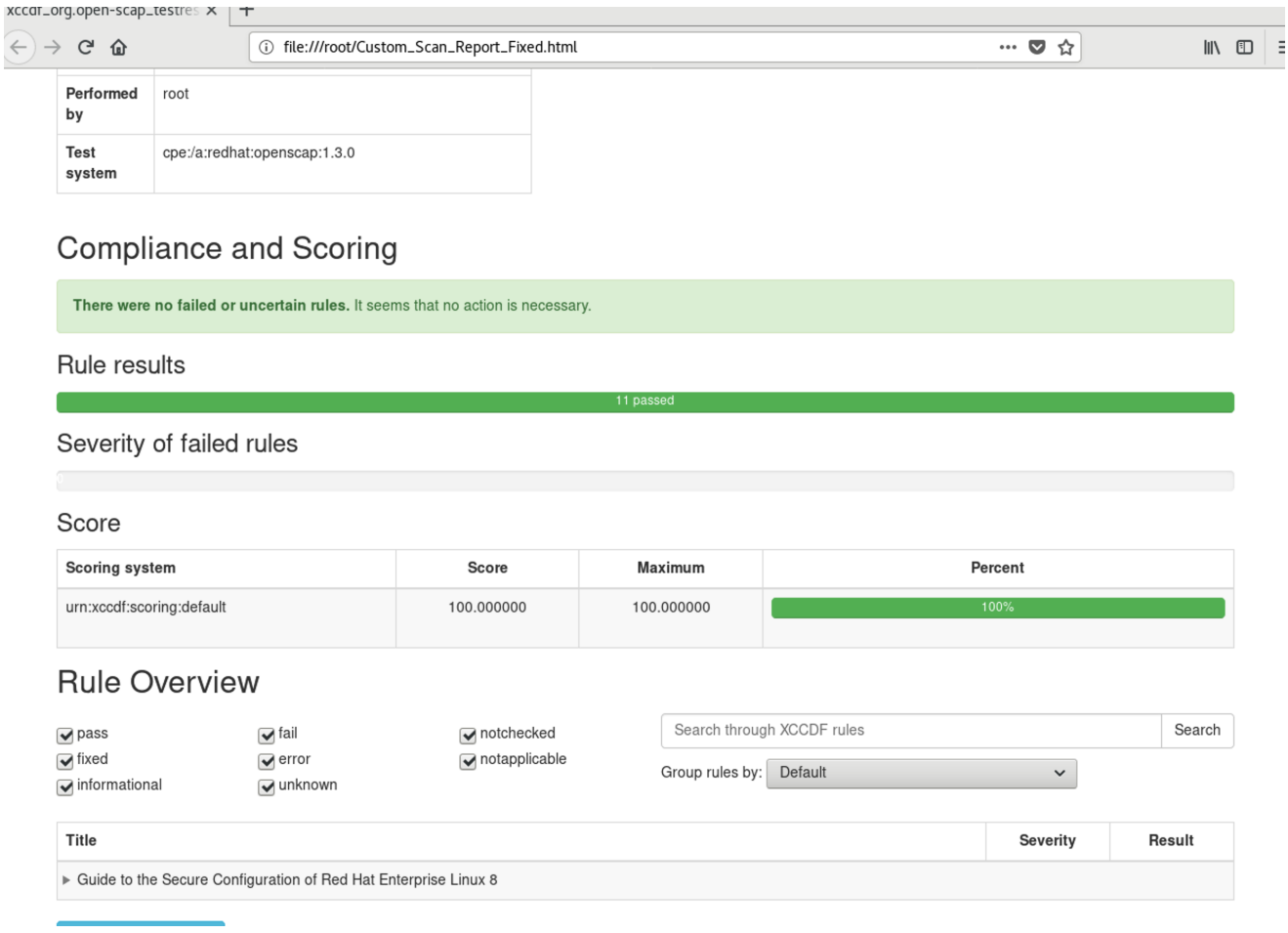


Figure 46. Fixed SCAP Scan Results Report in Firefox