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

1.1. 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 1. 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 2. Using **tmux**

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

[Chapter1 7d227] | *Chapter1-7d227.png*

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

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

*Listing 3. Detaching a **tmux** Session*

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

4. Re-attaching to **tmux**

Listing 4. 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 5. 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
```

[Chapter1 bf6b1] | *Chapter1-bf6b1.png*

*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
```

1.2. SystemD Overview

1.2.1. 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 6. Init Script to Run at Startup

```
[root@servera ~]# 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 7. Running **chmod** *on script*

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

3. Edit SystemD Service (Unit File)

Listing 8. Editing the .service File

```
[root@servera ~]# 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 9. Running **chmod** *on .service File*

```
[root@servera ~]# 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 10. Controlling a Custom Service with **systemctl**

```
[root@servera ~]# 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 11. Using **systemctl** *to Check Service Status*

```
[root@servera ~]# 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 servera.lab.example.com systemd[1]: Started UPS examp>
May 20 18:08:22 servera.lab.example.com bash[2136]: Looping..
```

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

Listing 12. Seaching Log file for Custom Service

```
[root@servera ~]# 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

1.3. 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.

1.3.1. 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/

1.3.2. 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 13. Allowing HTTP through the Firewall by Port/Protocol

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

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

Listing 14. Allowing HTTP through the Firewall by Service

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

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

[root@servera ~]# 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.

1.3.3. 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 15. FirewallD Configuration Files

```
[root@servera ~]# 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 16. FirewallD Default Configuration Files

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

1.3.4. 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 17. Using SSH Service File as a Starting Point

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

2. Edit the file and specify the new options

Listing 18. Edit the Custom SSH Service Definition

```
[root@servera ~]# 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 19. Getting Listing of FirewallD Services

```
[root@servera ~]# 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.

1.3.5. 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 20. FirewallD Configuration for Public

```
[root@servera ~]# 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 21. Adding our new Service

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

3. Verifying Firewall configuration for the Public Zone.

Listing 22. FirewallD Configuration for Public with Custom Service

```
[root@servera ~]# 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>
```

1.4. 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.

1.4.1. Installing Additional Cockpit Packages

Listing 23. Install all Base Cockpit Packages

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

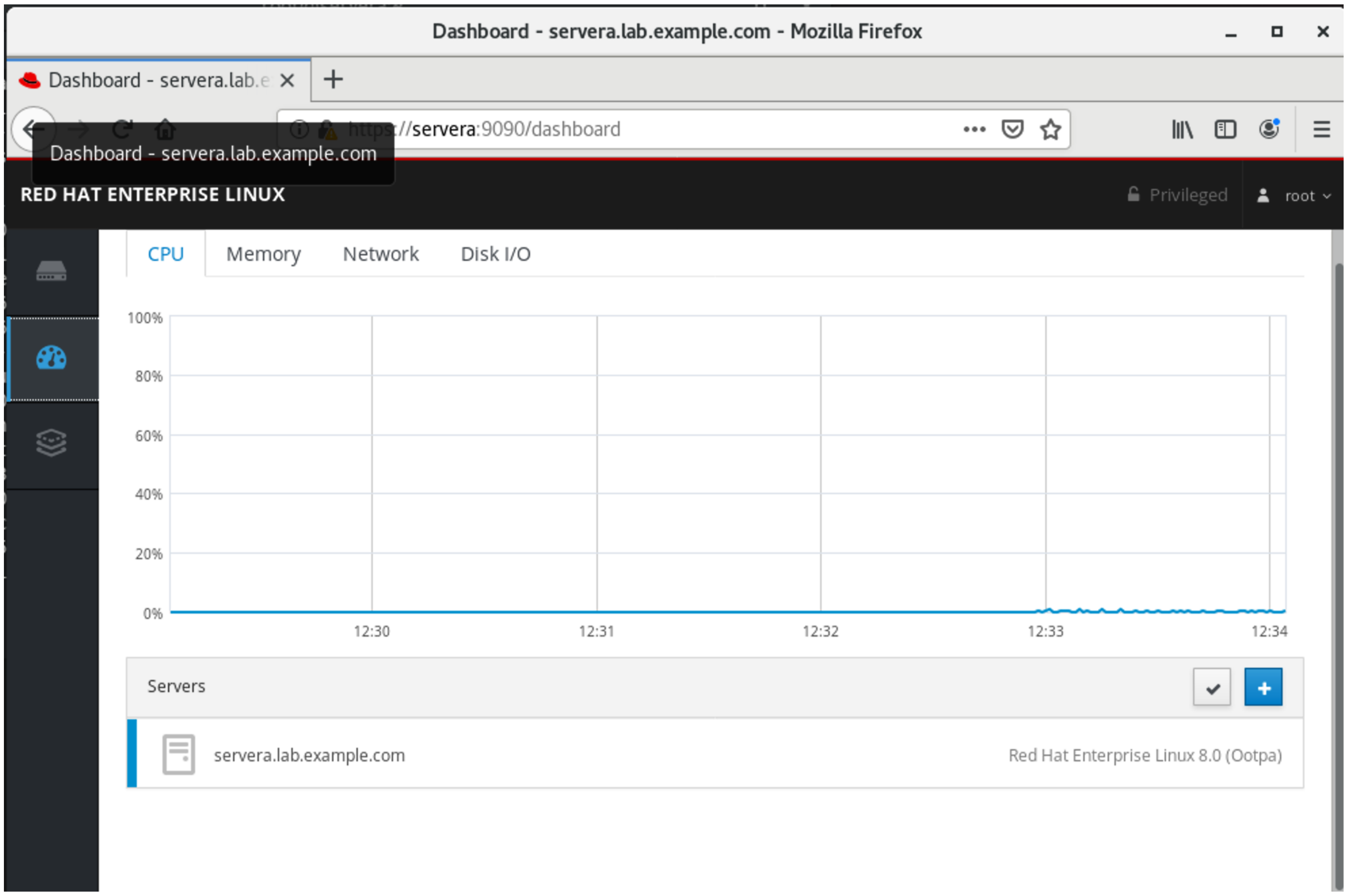

Dashboard - servera.lab.example.com - Mozilla Firefox
Dashboard - servera.lab.e
https://servera:9090/dashboard
Dashboard - servera.lab.example.com
RED HAT ENTERPRISE LINUX
Privileged
root
CPU
Memory
Network
Disk I/O
100%
80%
60%
40%
20%
0%
12:30
12:31
12:32
12:33
12:34
Servers
servera.lab.example.com
Red Hat Enterprise Linux 8.0 (Ootpa)

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.

1.4.1.1. Cockpit to Manage Multiple Systems

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

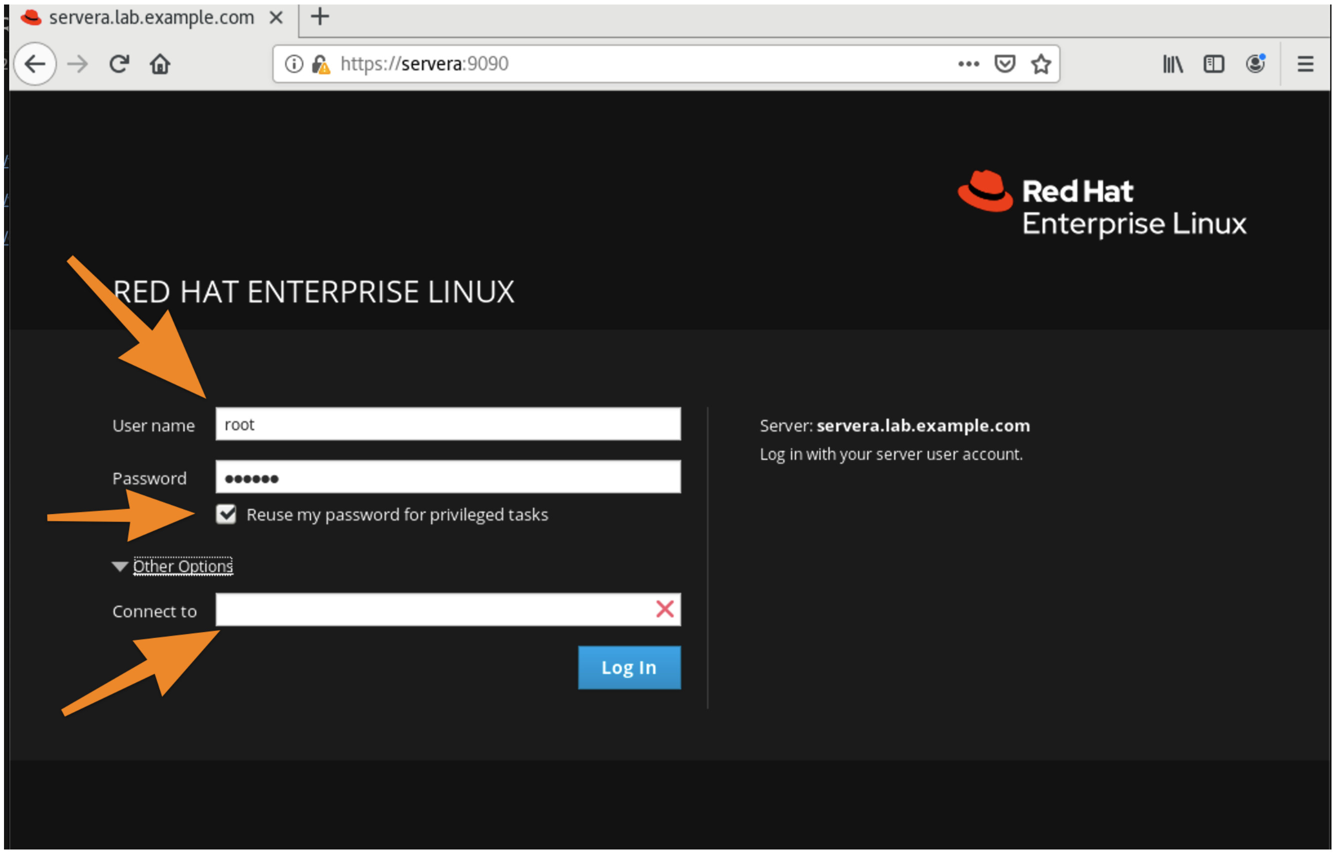
1. Ensure cockpit socket/service is running and configured on all systems.

Listing 24. Test and Enable Cockpit

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

[root@servera ~]# 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

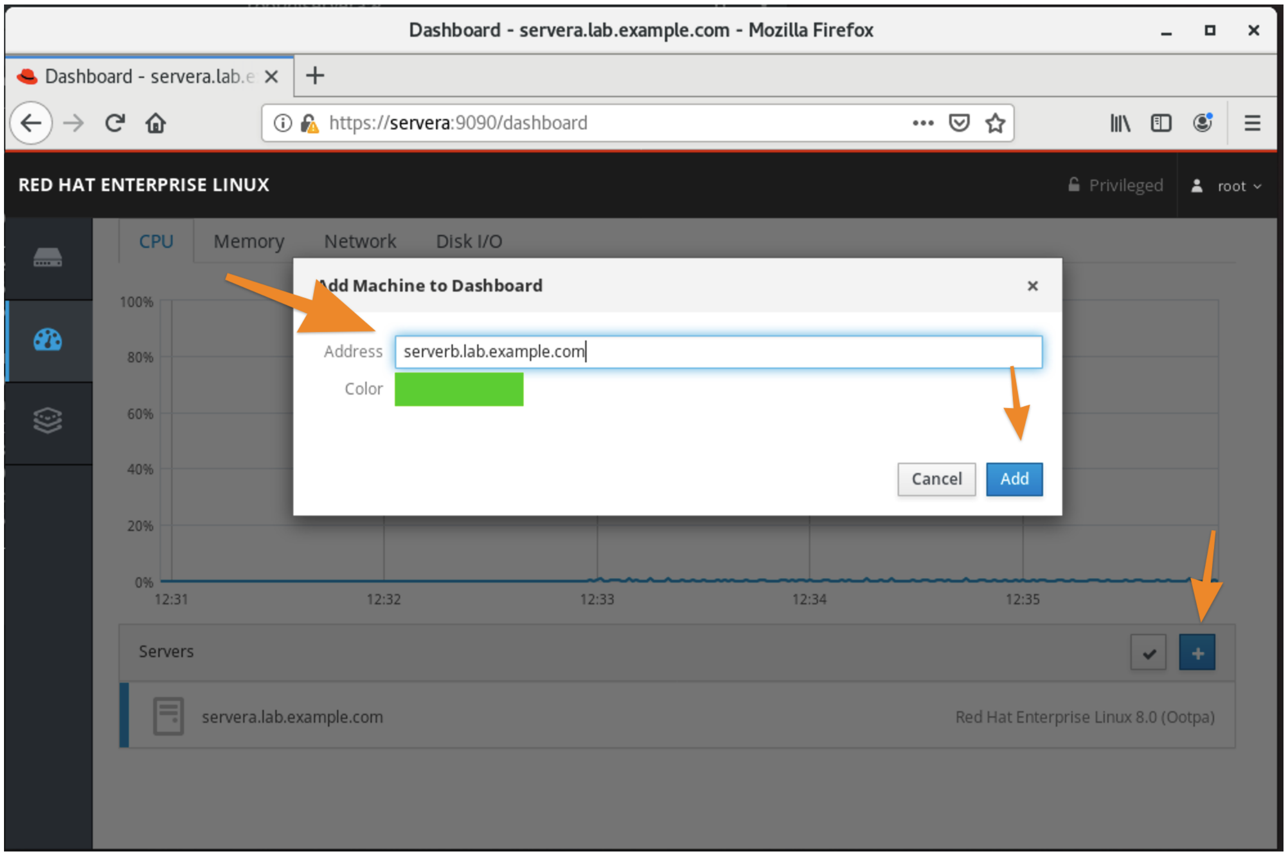


User name: root
Password: ●●●●●●
☑ Reuse my password for privileged tasks
Connect to:

Server: **servera.lab.example.com**
Log in with your server user account.

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


Dashboard - servera.lab.example.com - Mozilla Firefox
Dashboard - servera.lab.e
https://servera:9090/dashboard
RED HAT ENTERPRISE LINUX
Privileged
root
CPU
Memory
Network
Disk I/O
Servers
Cancel
Add

Add Machine to Dashboard

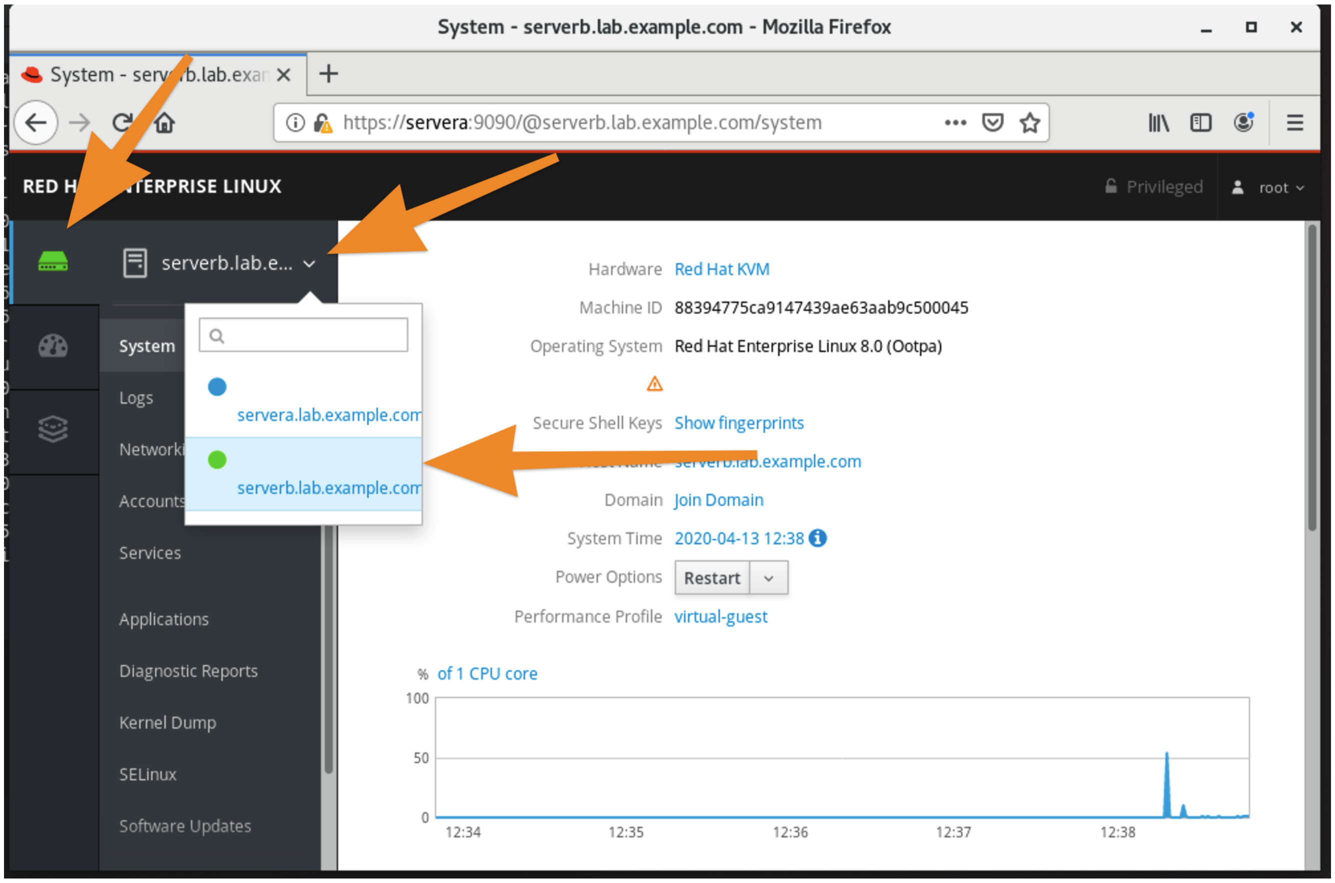
Address: serverb.lab.example.com

Color: 🟩

servera.lab.example.com — Red Hat Enterprise Linux 8.0 (Ootpa)

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



Hardware	Red Hat KVM
Machine ID	88394775ca9147439ae63aab9c500045
Operating System	Red Hat Enterprise Linux 8.0 (Ootpa)
Secure Shell Keys	Show fingerprints
Host Name	serverb.lab.example.com
Domain	Join Domain
System Time	2020-04-13 12:38
Power Options	Restart
Performance Profile	virtual-guest

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.

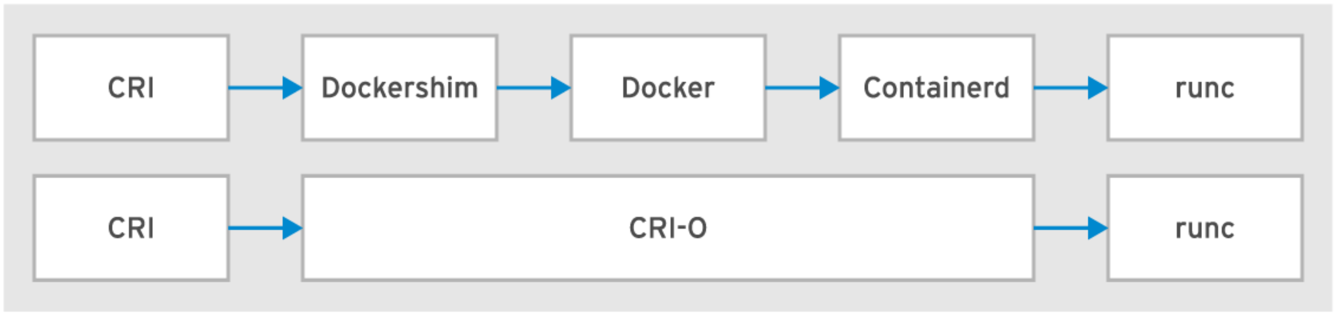


Figure 7. Docker to RHEL8 Container Runtime

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.

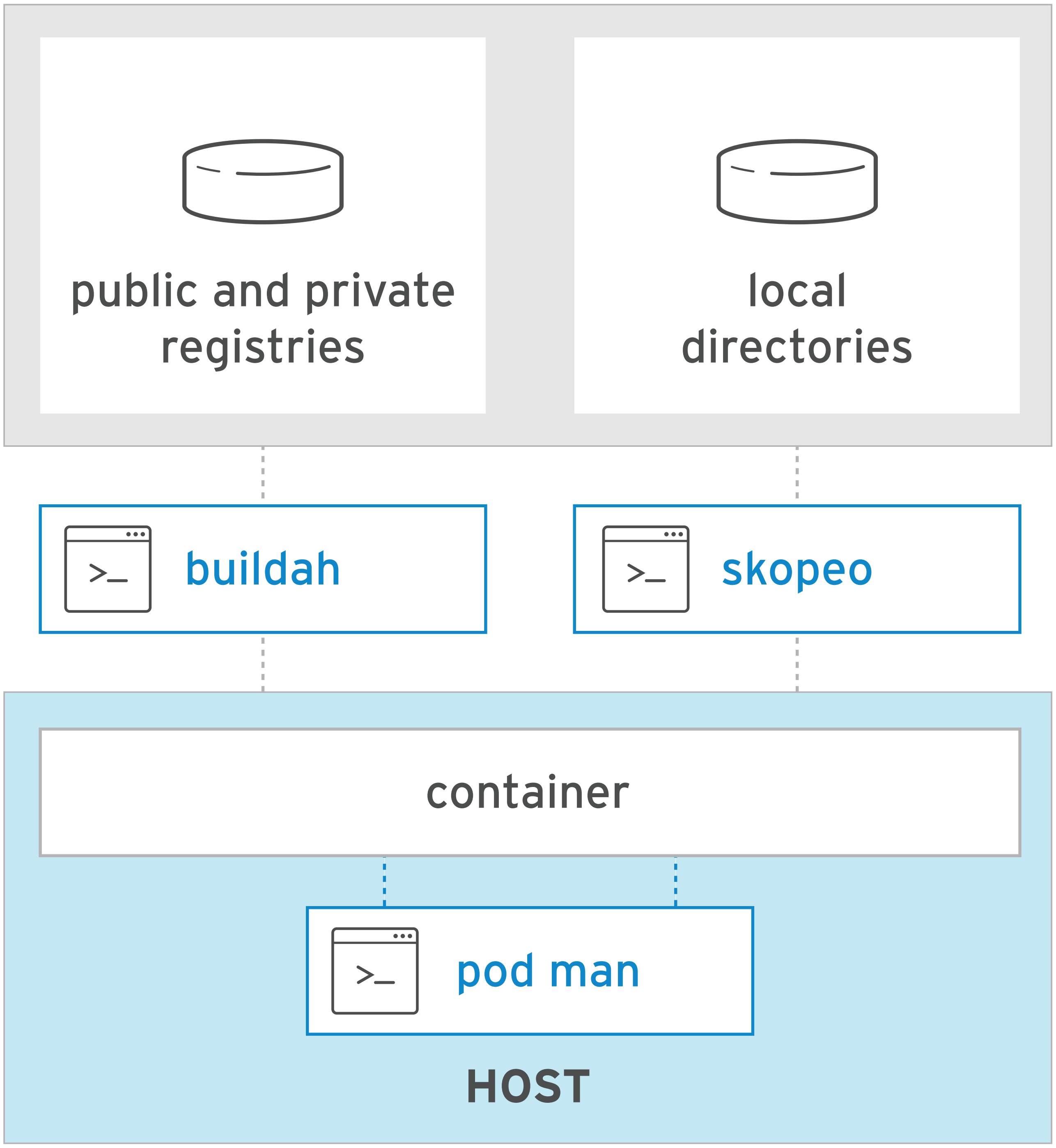


Figure 8. New Container Runtime

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 25. Installation of Container Tools

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

2.2. Podman Configuration Files

2.3. Container Image Storage

2.4. Managing Containers using the Red Hat Web Console

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/

Podman System Prune:

man pages

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

4. Building Containers with Buildah

Example 1. EXAMPLE - Creating a Custom Container Image Using Buildah

Listing 26. Creating a Custom Container

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

Listing 27. Naming and Inspecting a Custom Container

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

Listing 28. 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

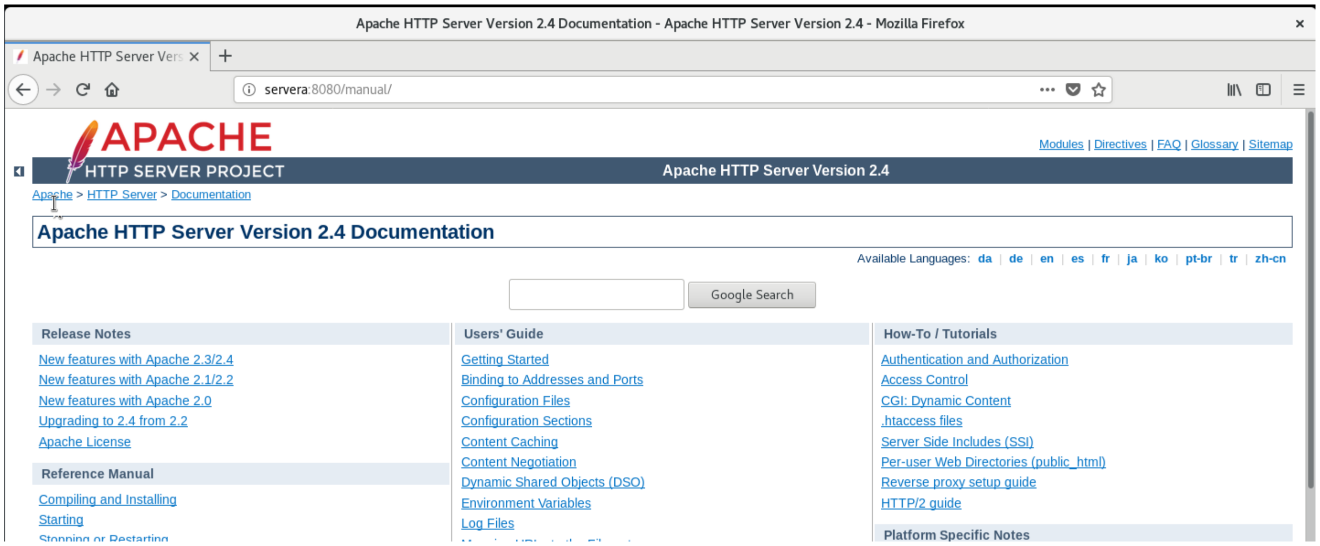
⑬ List container images

Listing 29. 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/
```



Apache HTTP Server Version 2.4 Documentation - Apache HTTP Server Version 2.4 - Mozilla Firefox

Apache HTTP Server Vers

servera:8080/manual/

APACHE HTTP SERVER PROJECT

Modules | Directives | FAQ | Glossary | Sitemap

Apache HTTP Server Version 2.4

Apache > HTTP Server > Documentation

Apache HTTP Server Version 2.4 Documentation

Available Languages: da | de | en | es | fr | ja | ko | pt-br | tr | zh-cn

Google Search

Release Notes
- New features with Apache 2.3/2.4
- New features with Apache 2.1/2.2
- New features with Apache 2.0
- Upgrading to 2.4 from 2.2
- Apache License

Reference Manual
- Compiling and Installing
- Starting
- Stopping or Restarting

Users' Guide
- Getting Started
- Binding to Addresses and Ports
- Configuration Files
- Configuration Sections
- Content Caching
- Content Negotiation
- Dynamic Shared Objects (DSO)
- Environment Variables
- Log Files

How-To / Tutorials
- Authentication and Authorization
- Access Control
- CGI: Dynamic Content
- .htaccess files
- Server Side Includes (SSI)
- Per-user Web Directories (public_html)
- Reverse proxy setup guide
- HTTP/2 guide

Platform Specific Notes

Figure 9. Testing Container

Listing 30. 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. Building an Image Using Buildah Rootless

5. Containers as System Services

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 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.

6.1.1. 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 31. 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

6.1.2. 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.

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 32. 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.

References

References for this chapter

7. Quay Image Registry

7.1. Installing the Quay Image Registry

7.2. Using the Quay Image Registry

7.3. Inspecting Images with Skopeo on Remote Registries

Unresolved directive in Containers.adoc - include::Chapters/AppendixA.adoc[]