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
```

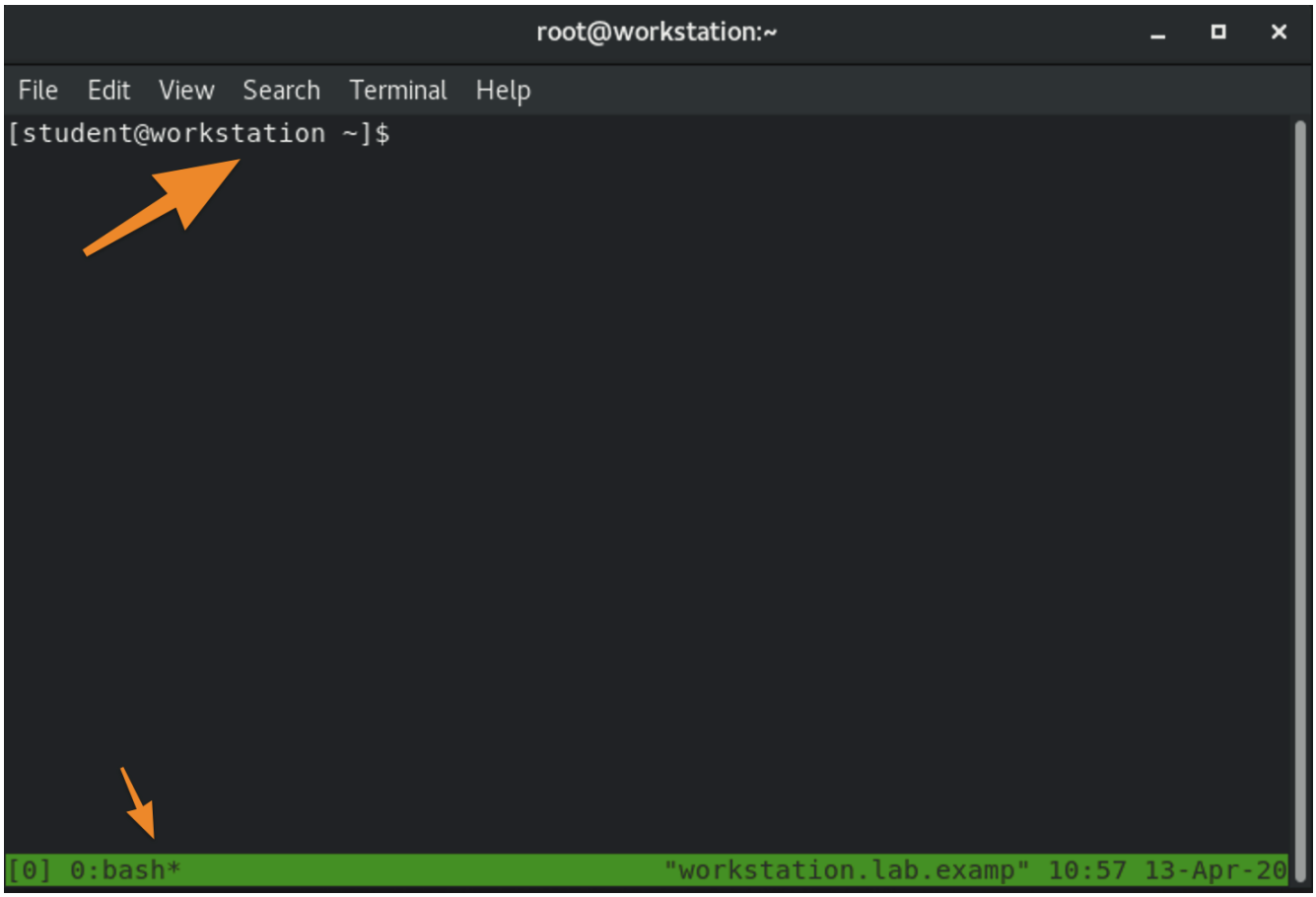

Figure 1. **tmux** *Terminal Window*

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
```

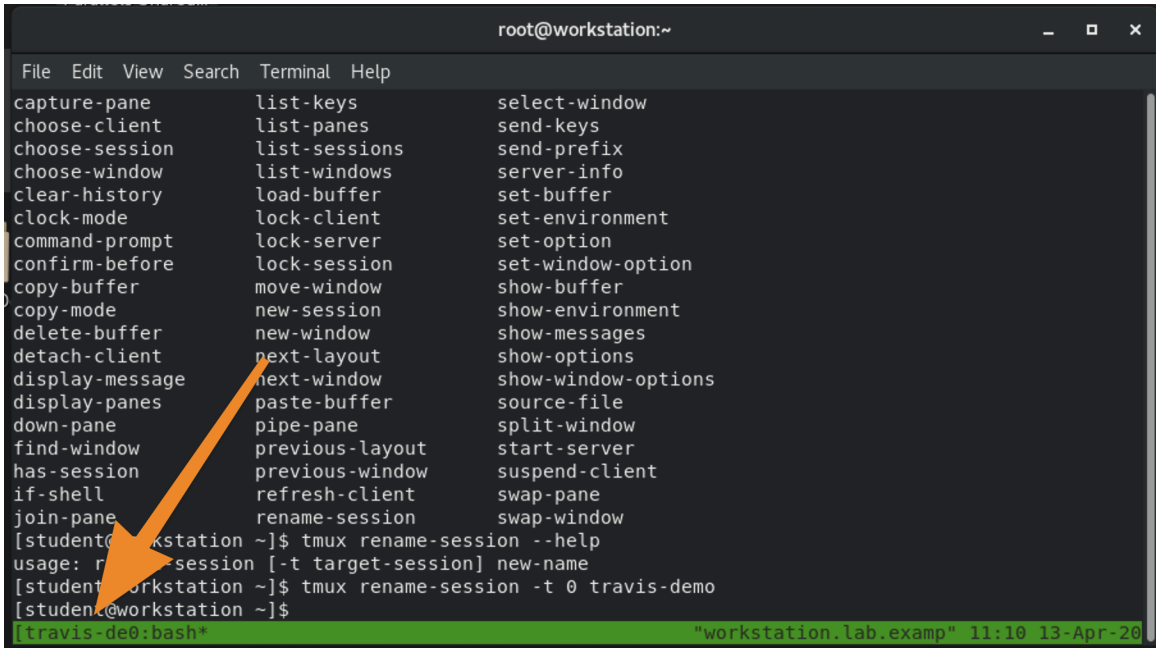

Figure 2. Renamed **tmux** *Session*

```
[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*
```

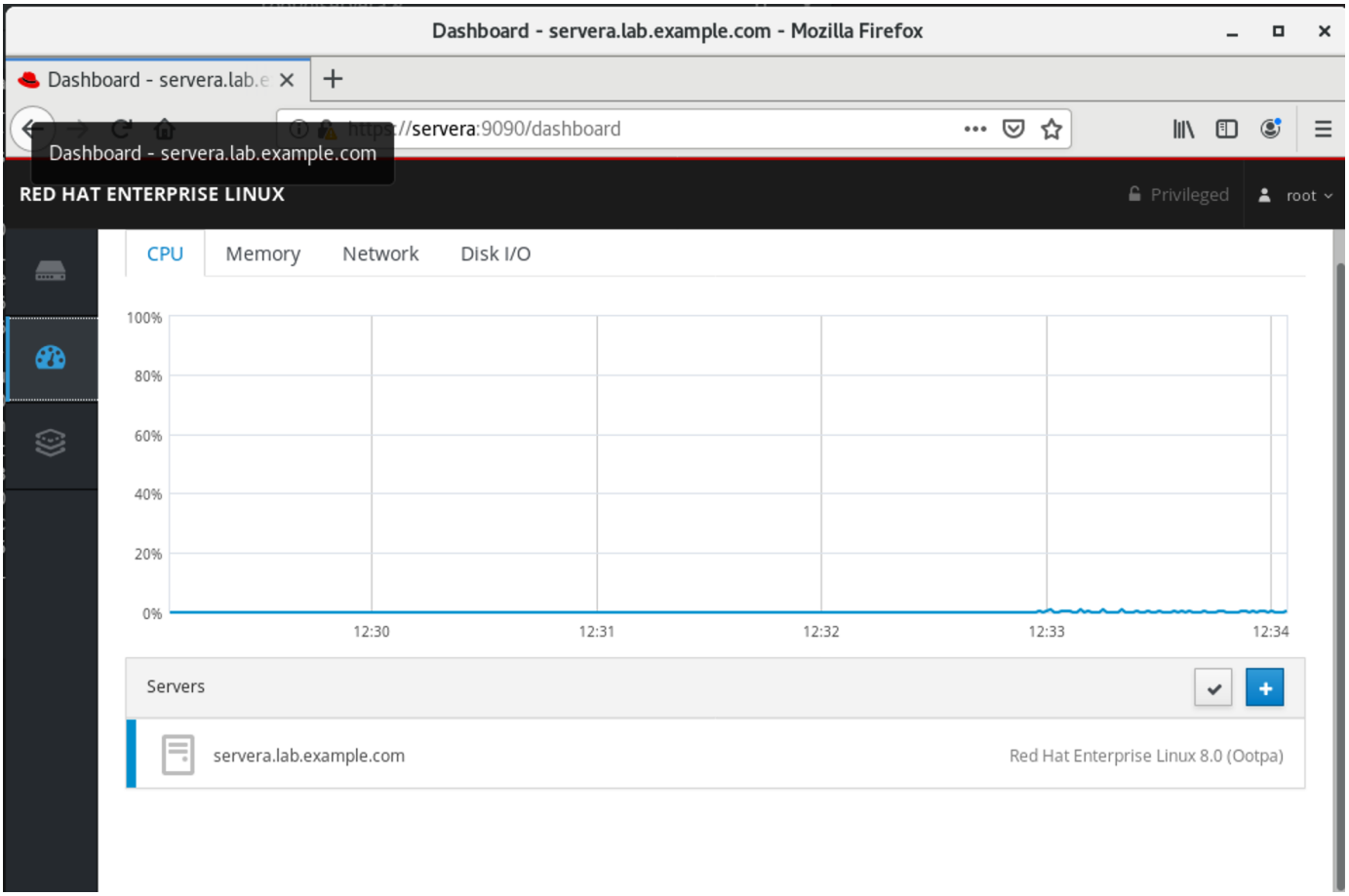

Figure 3. Cockpit Dashboard

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

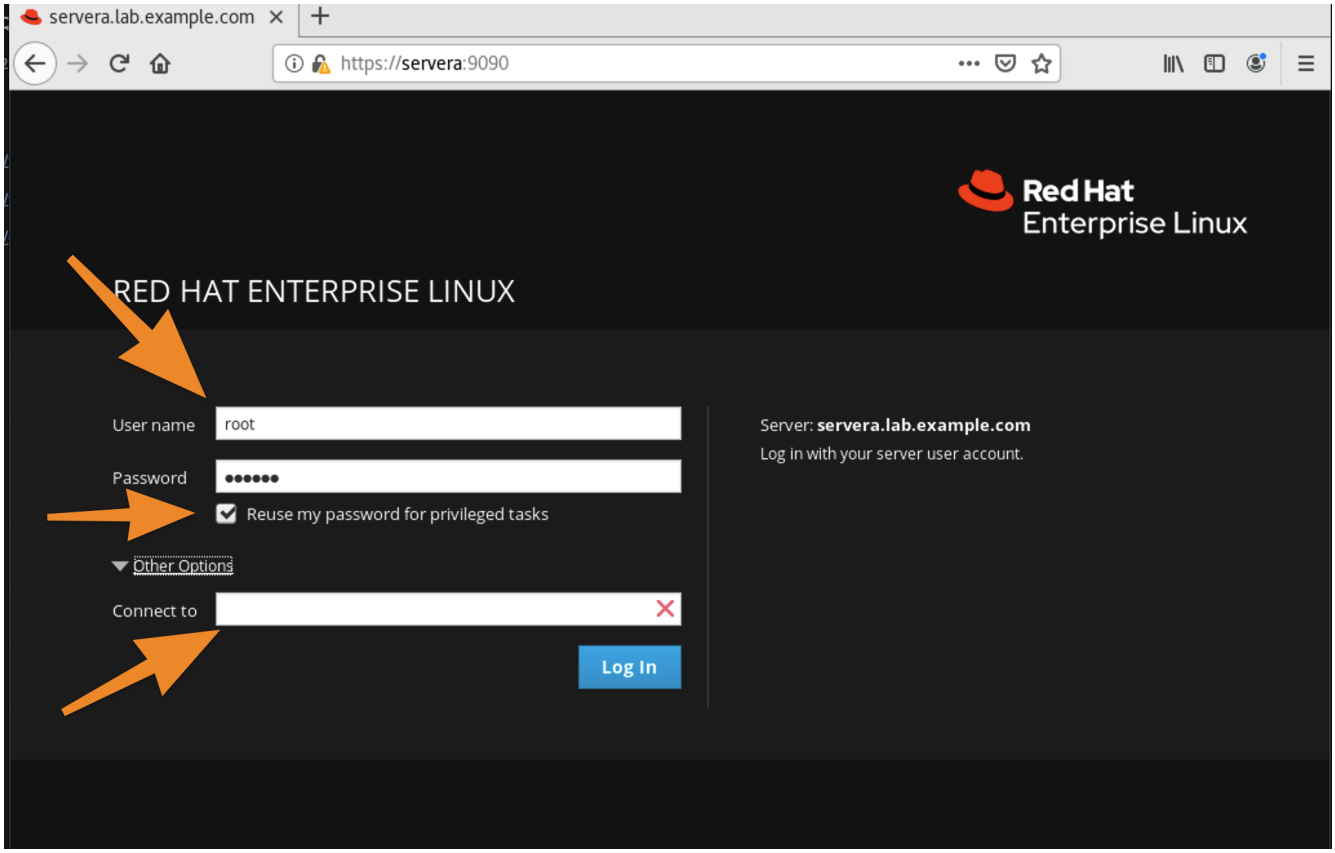

Figure 4. Red Hat Web Console (cockpit)

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

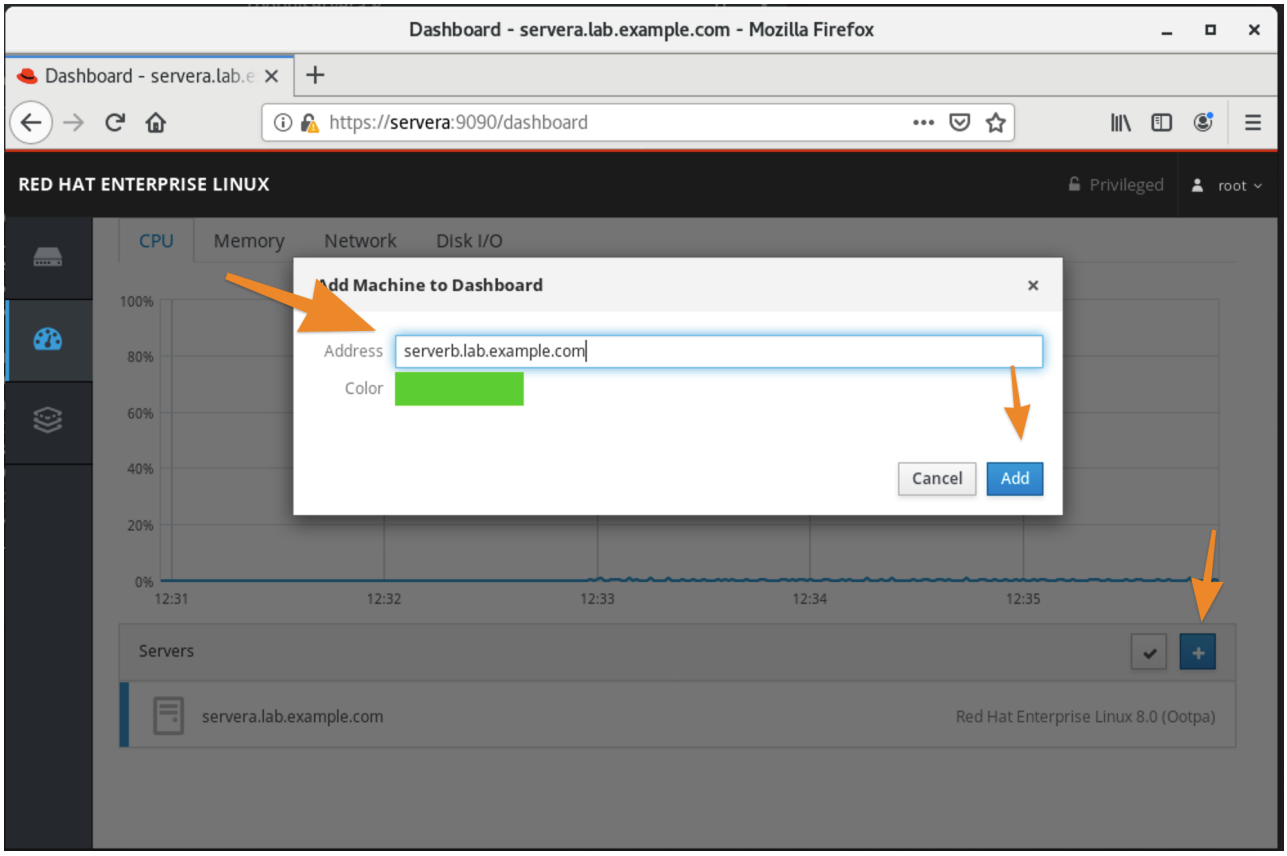

Figure 5. Adding Additional machines

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

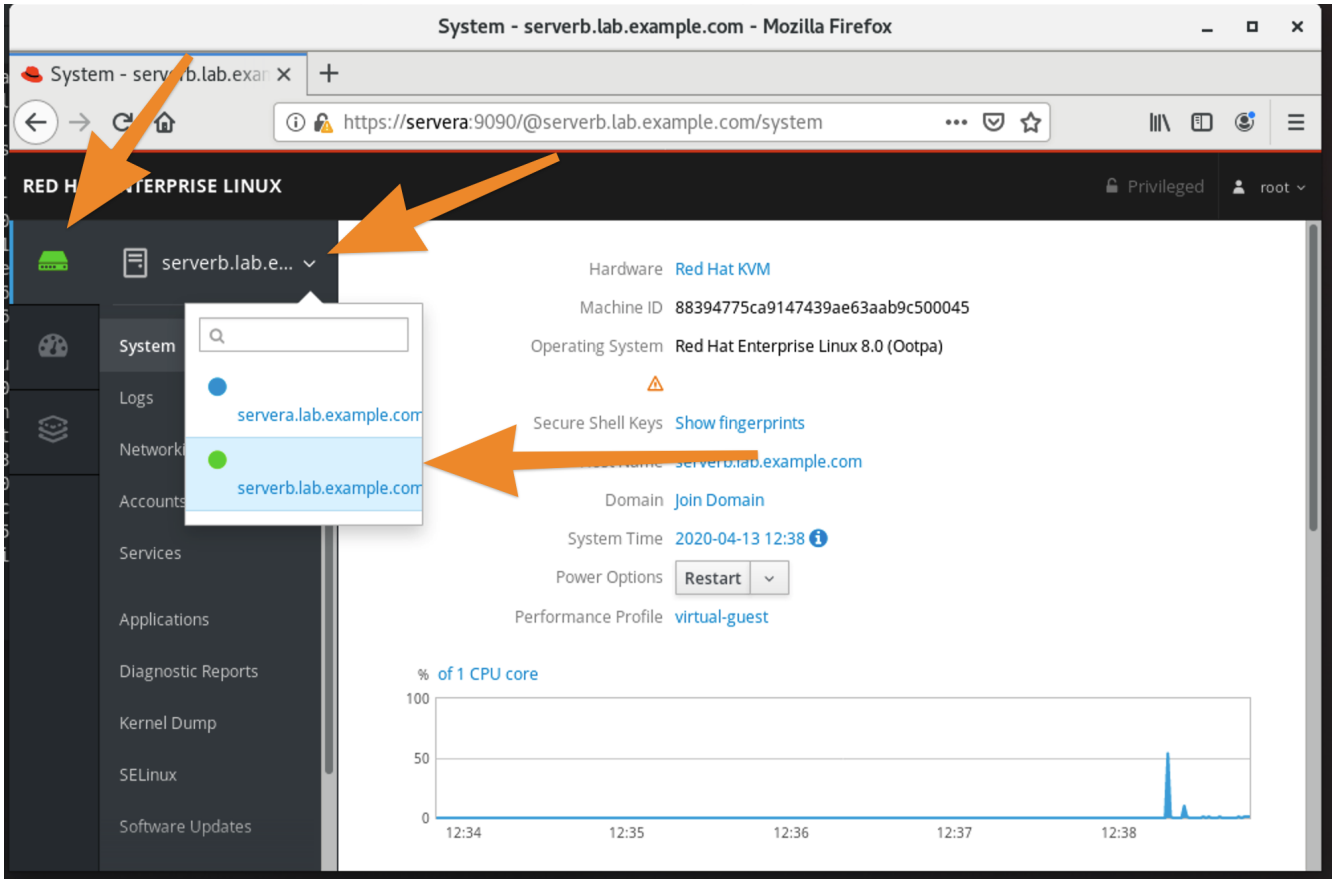

Figure 6. Switching Systems

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.

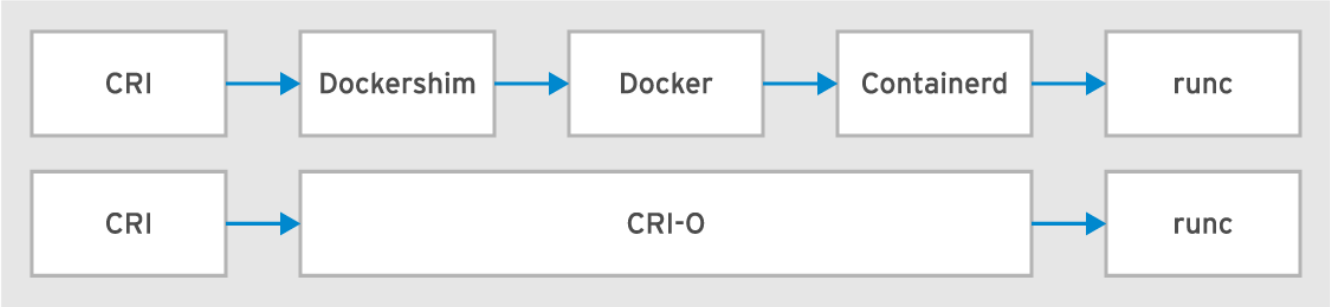

Figure 7. Docker to RHEL8 Container Runtime

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.

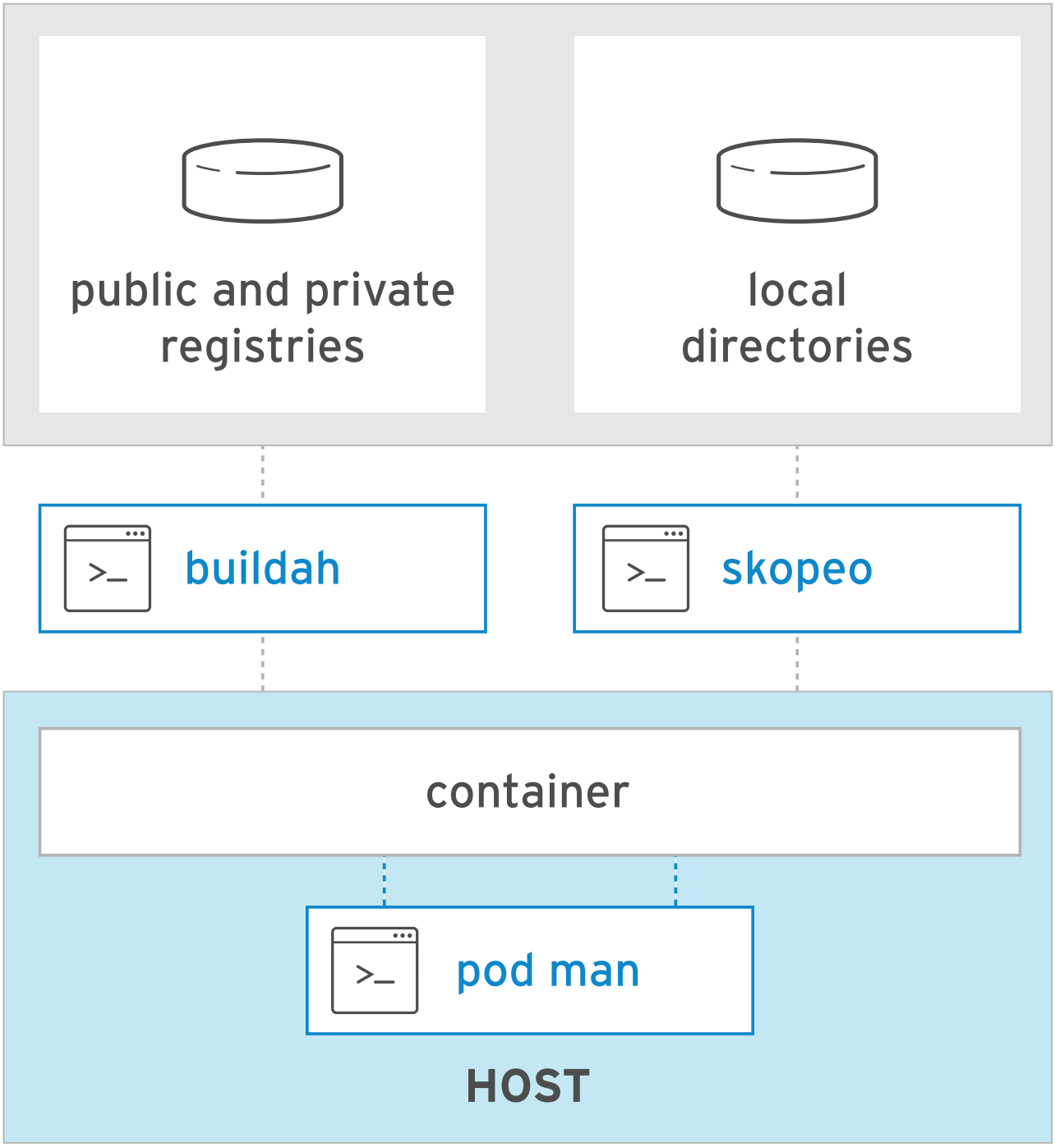

Figure 8. New Container Runtime

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 25. 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 *

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 26. 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 27. 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 28. **/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

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 29. Creating a Custom Container

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

Listing 30. Naming and Inspecting a Custom Container

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

Listing 31. 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 32. 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/
```

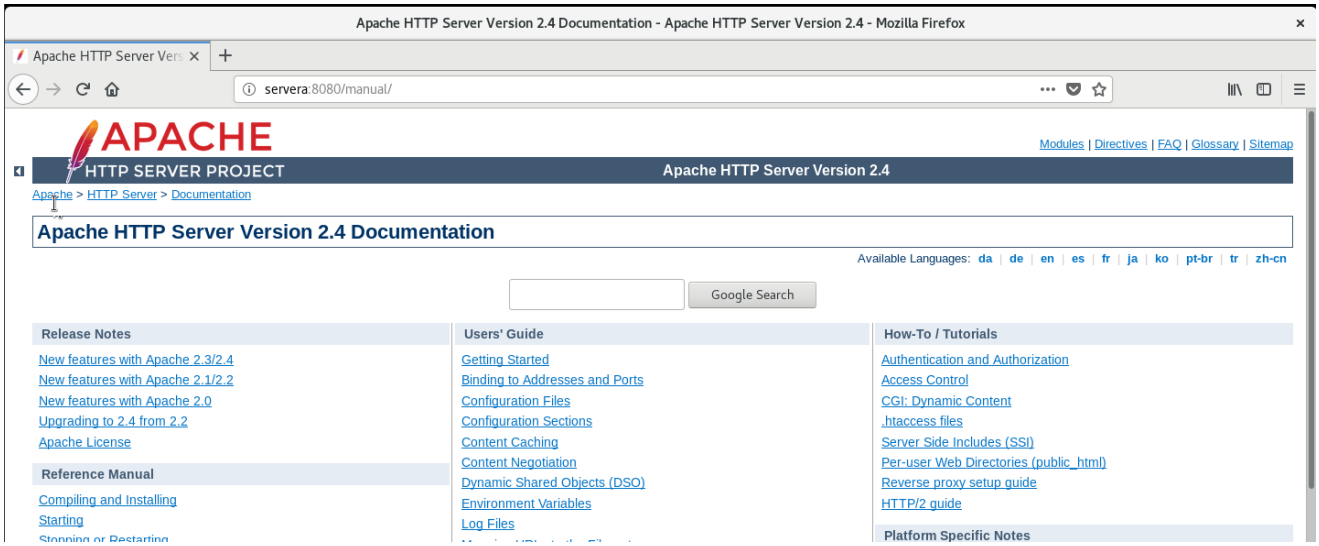

Figure 9. Testing Container

Listing 33. 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 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.

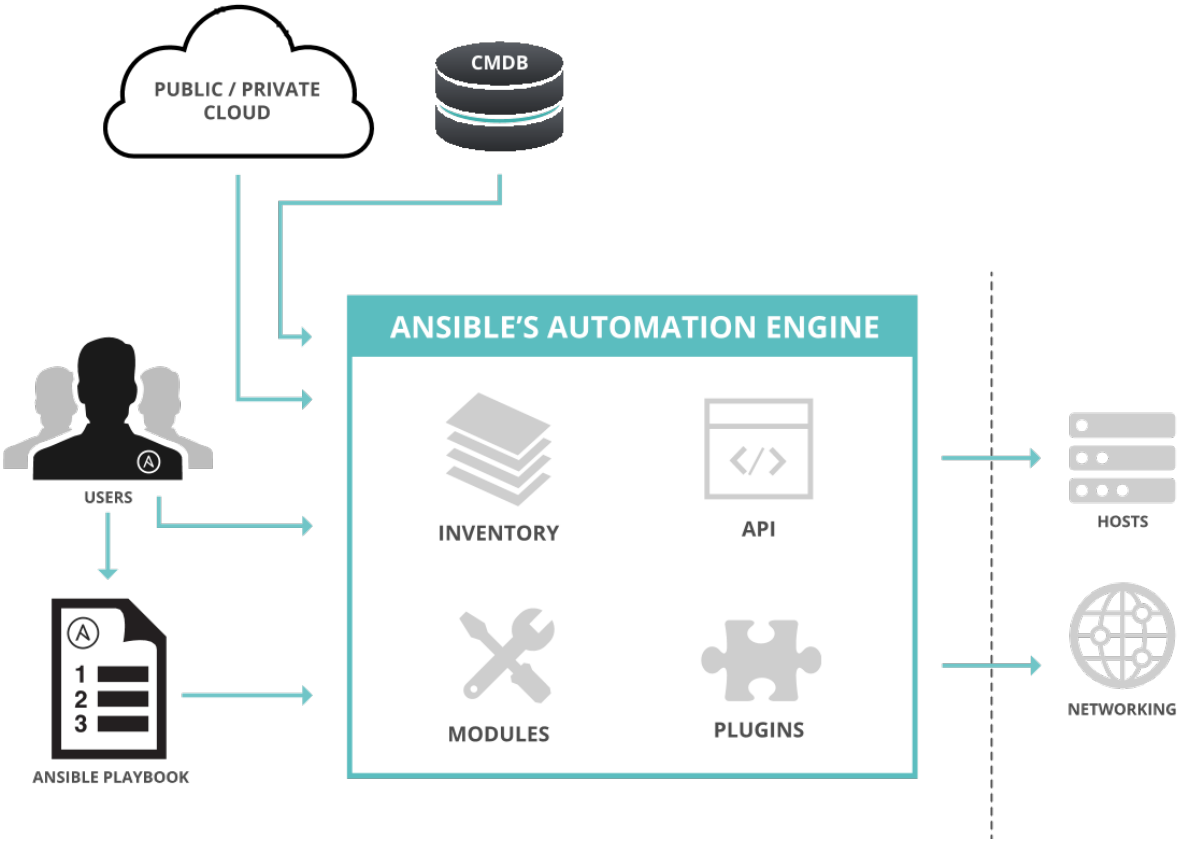

Figure 10. Ansible Architecture

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 34. 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 35. **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 36. 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 37. Podman Collection **requirements.yml** *File*

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

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

Listing 38. 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 39. 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 40. 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 41. **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 42. 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 43. 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 44. **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 45. 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 46. 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 47. 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 48. 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

7.1. Installing the Quay Image Registry

7.2. Using the Quay Image Registry

7.3. Inspecting Images with Skopeo on Remote Registries

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 2. Connecting to the Workstation VM

Listing 49. 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 3. Installing SCAP Workbench

Listing 50. 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 4. Launching SCAP Workbench

Listing 51. Using SCAP Workbench

```
# scap-workbench
```

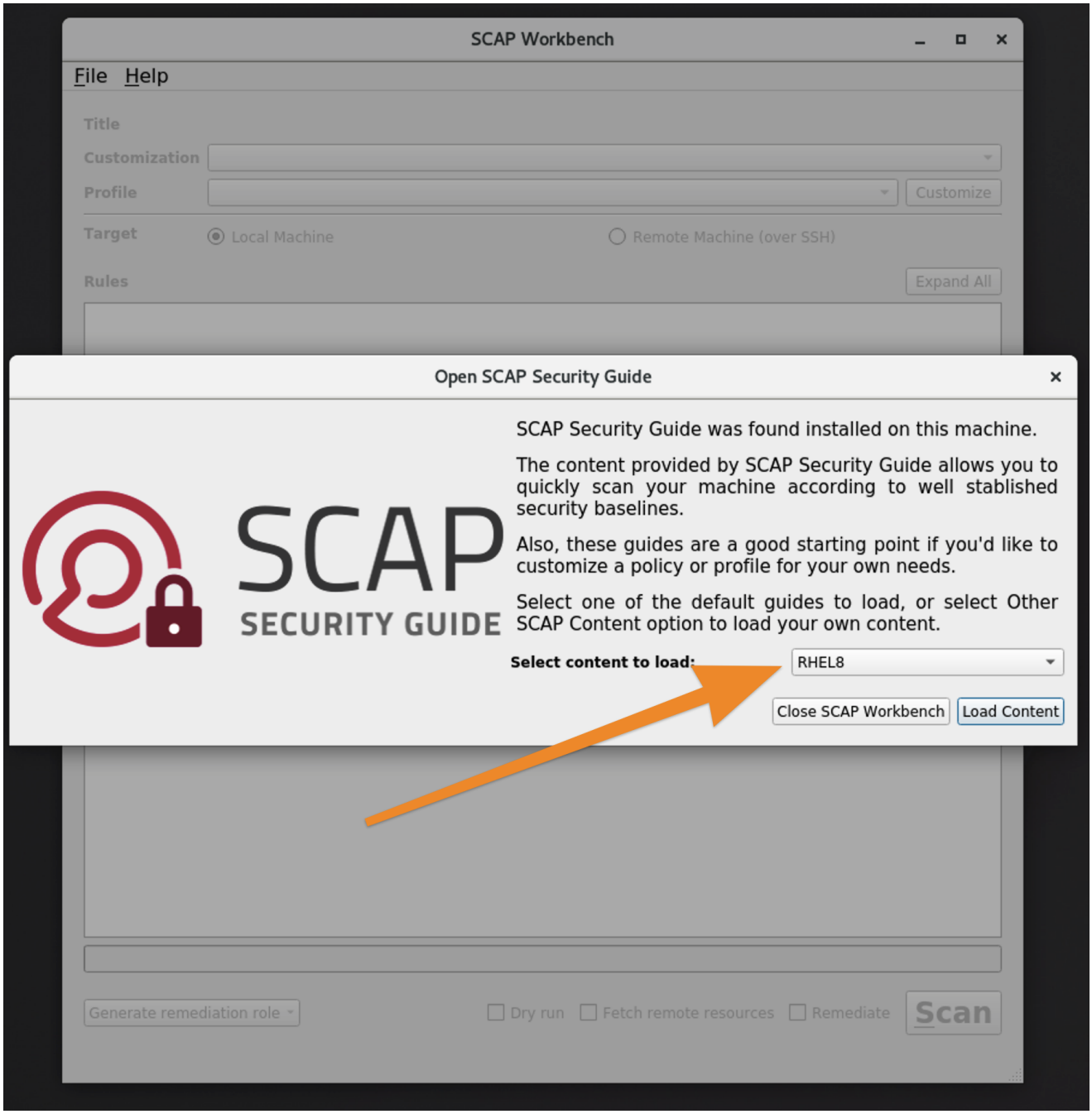

Figure 11. SCAP Workbench Startup

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 5. 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

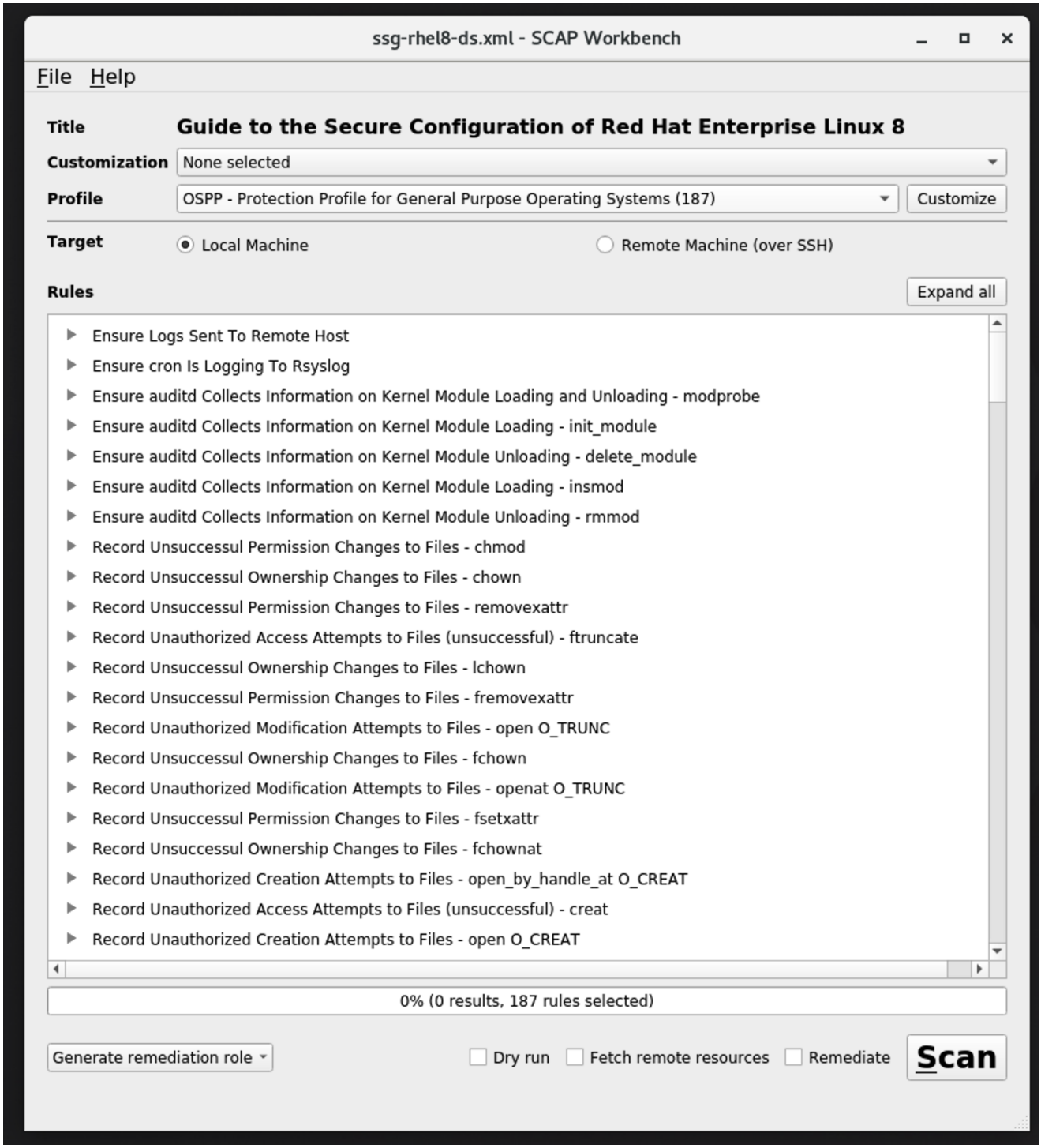

Figure 12. SCAP Workbench OSPP Profile

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

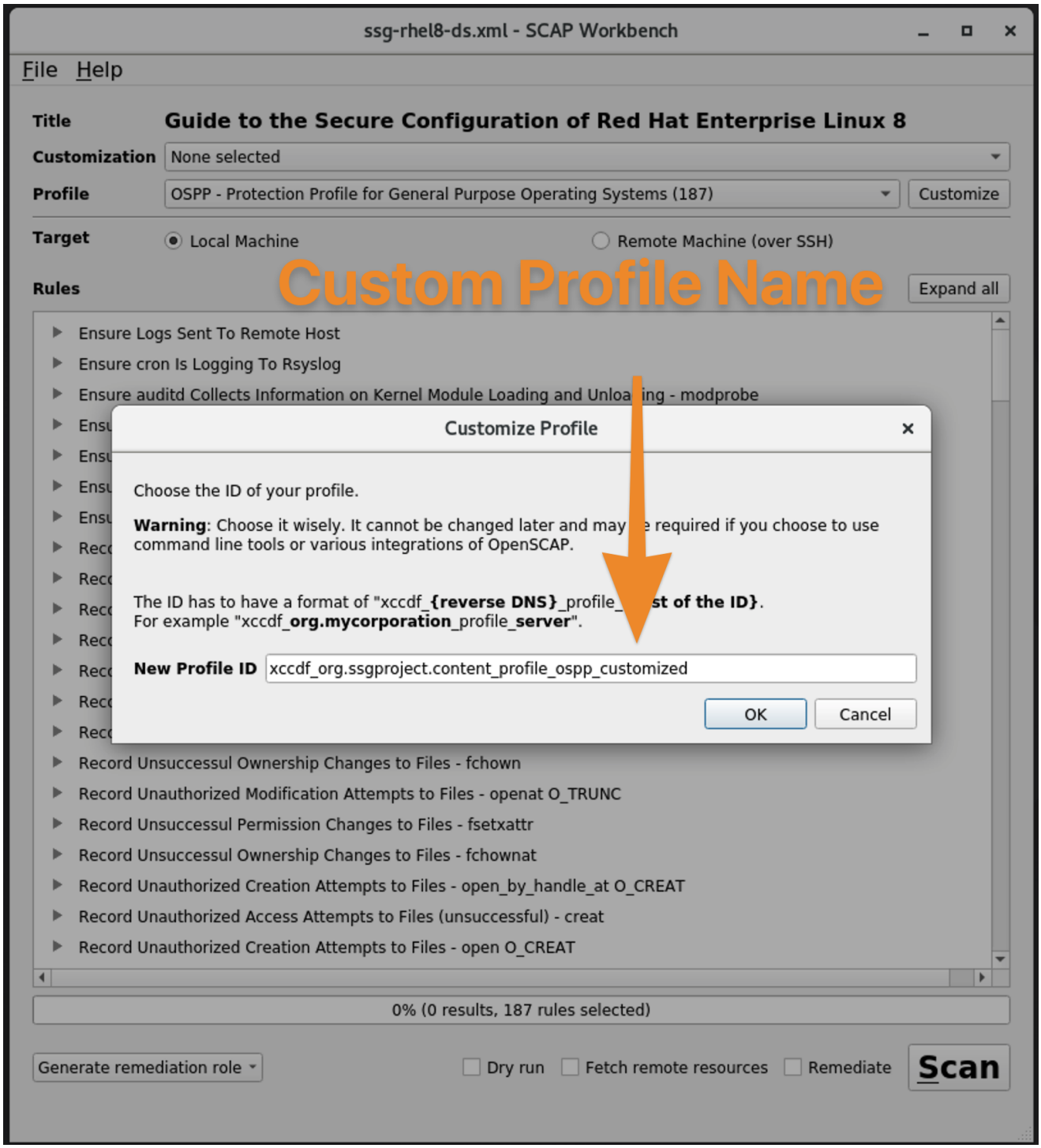

Figure 13. SCAP Custom OSPP Profile Creation

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.

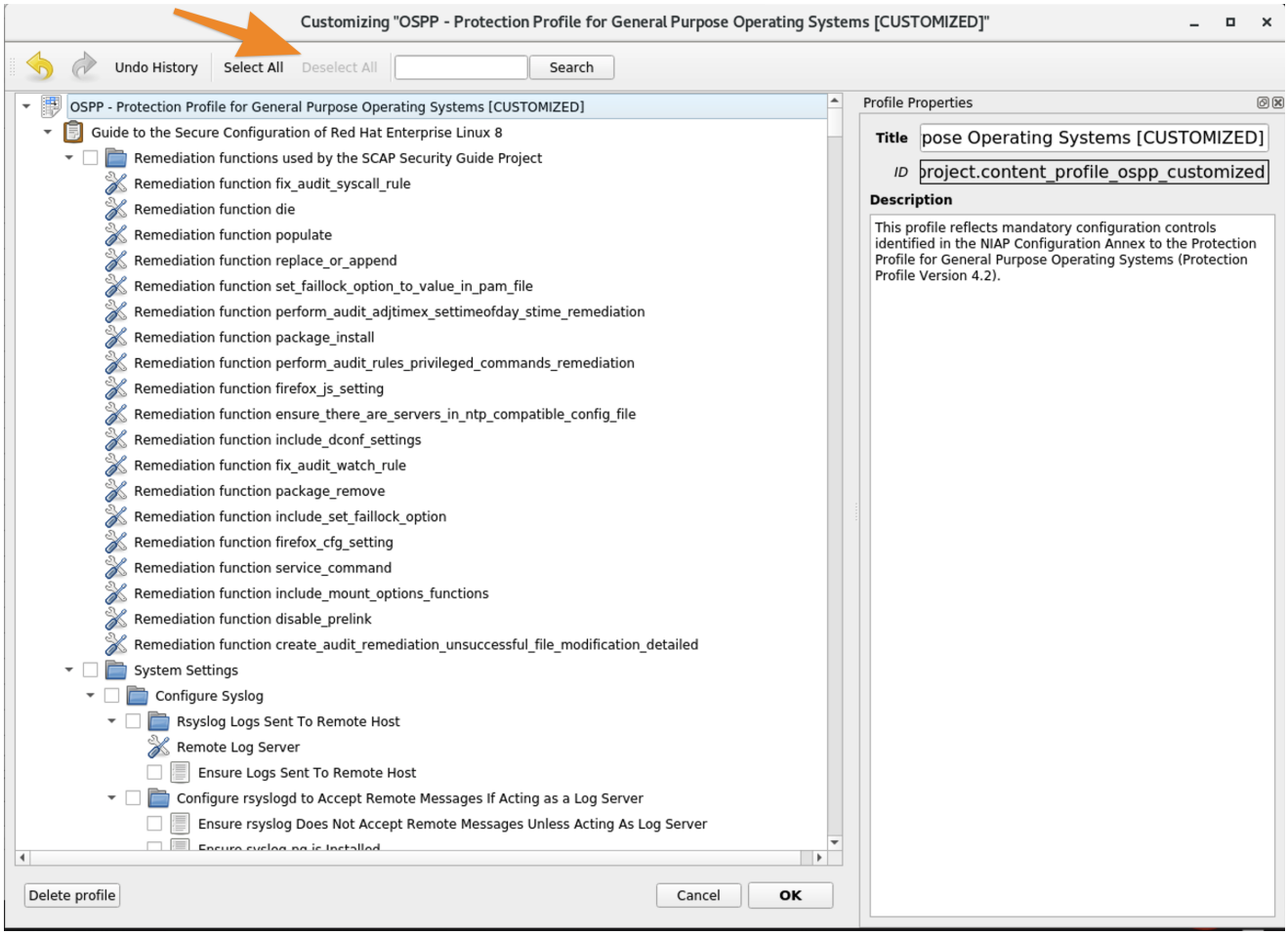

Figure 14. SCAP Custom Profile Selections

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**

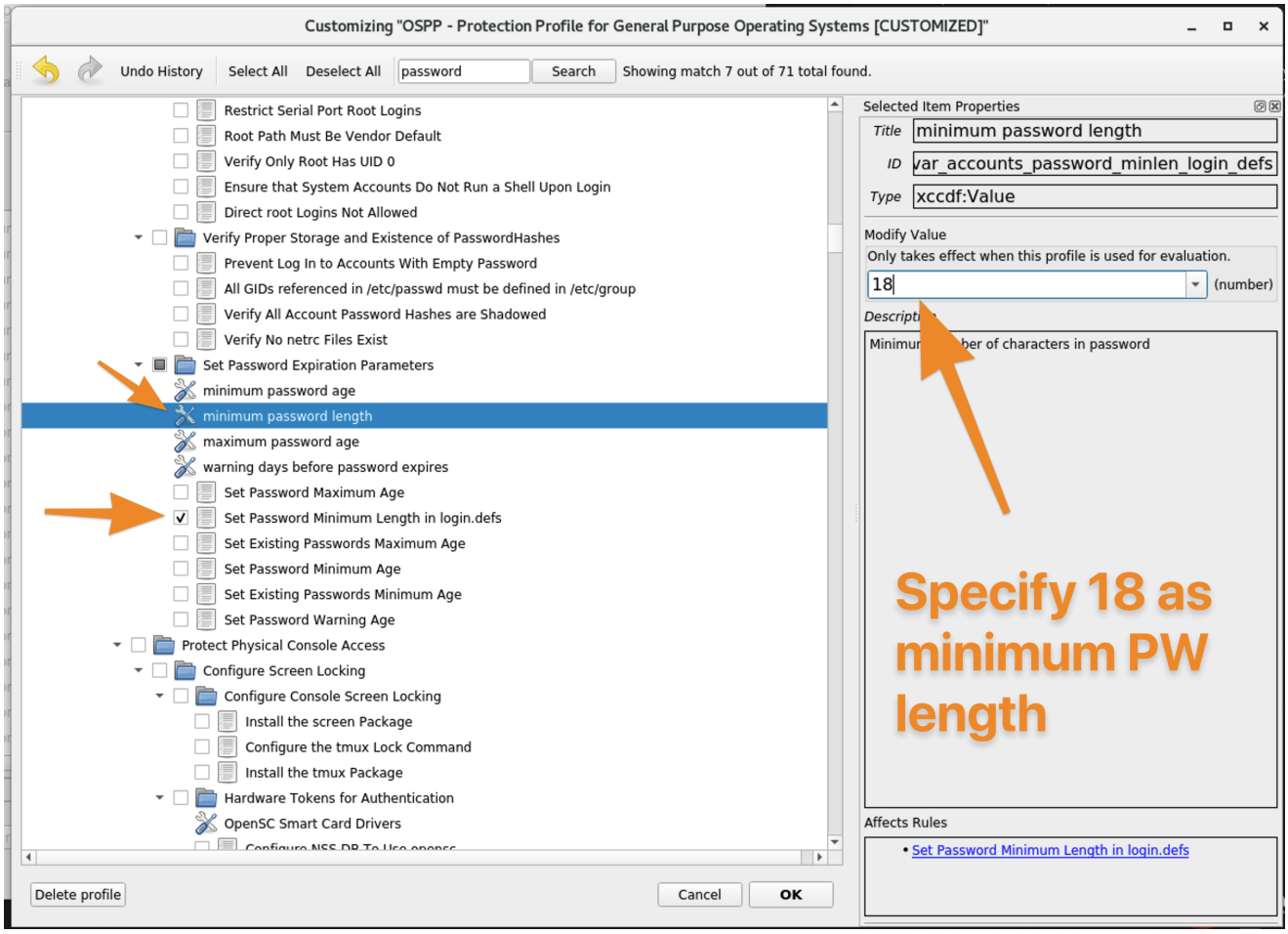

Figure 15. SCAP Custom Profile Password Settings for Login.Defs

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"

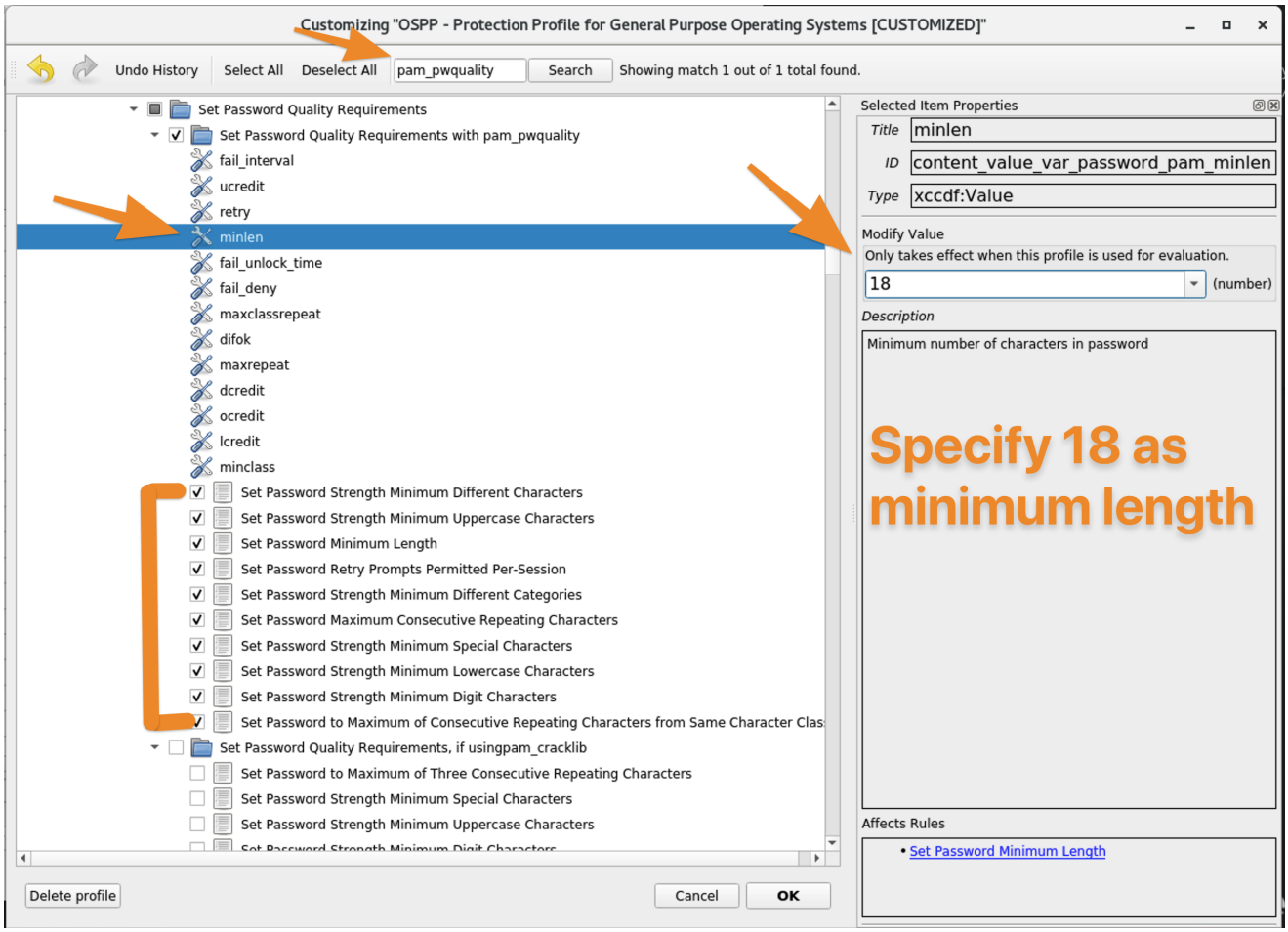

Figure 16. SCAP Custom Profile PAM Quality Requirements

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

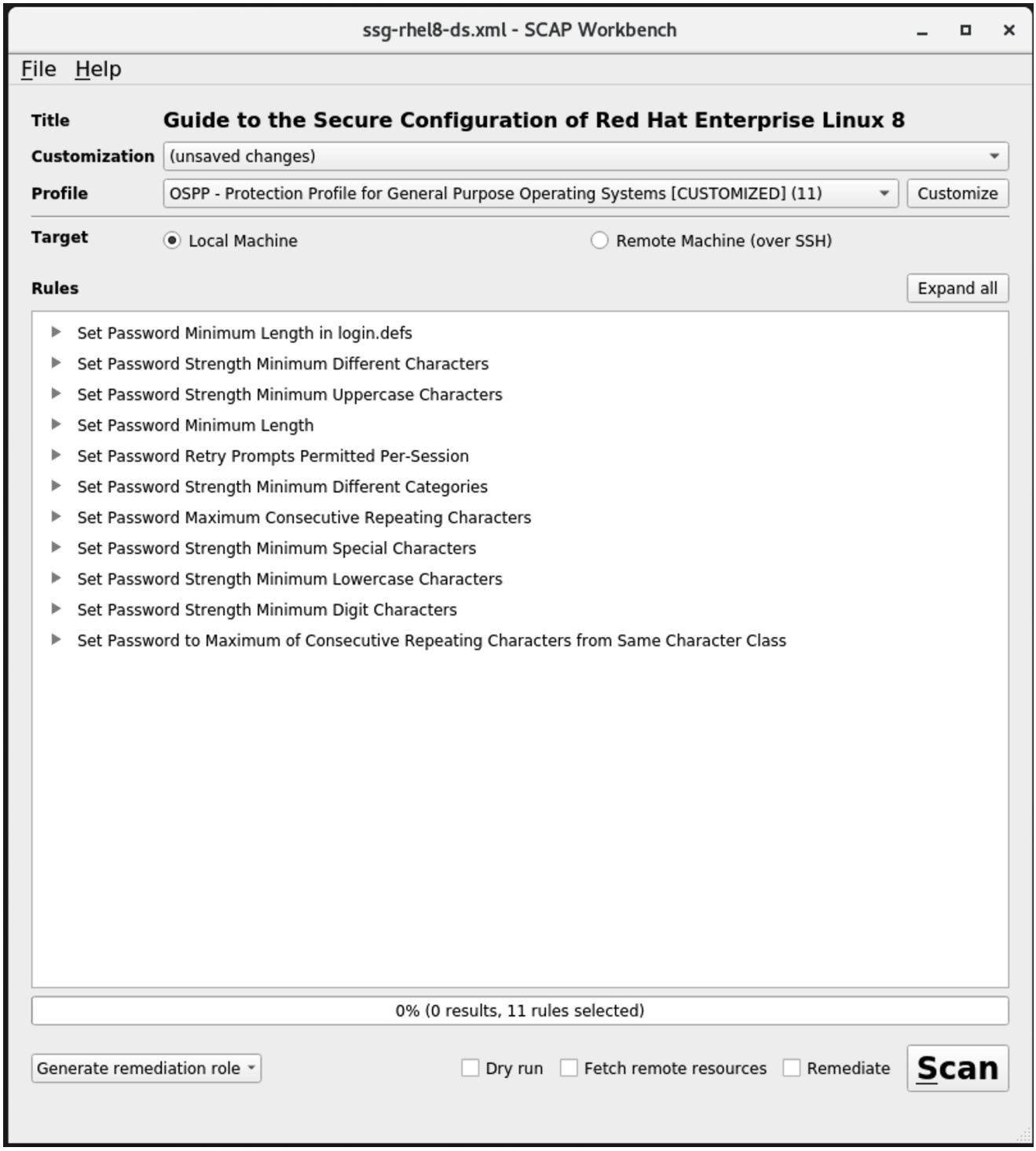

Figure 17. SCAP Custom Profile Selected Settings View

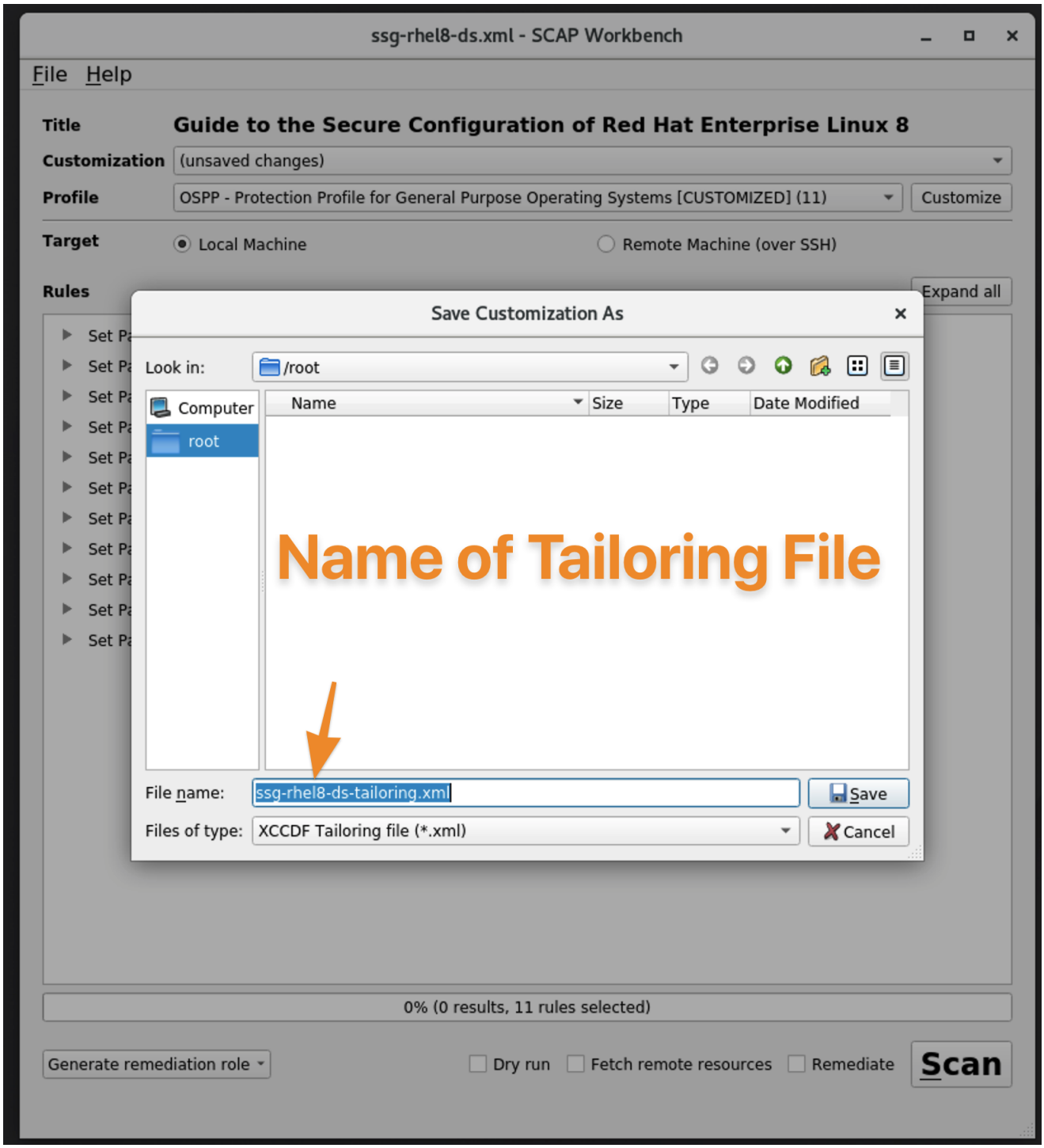

Figure 18. SCAP Custom Profile Creation Saving

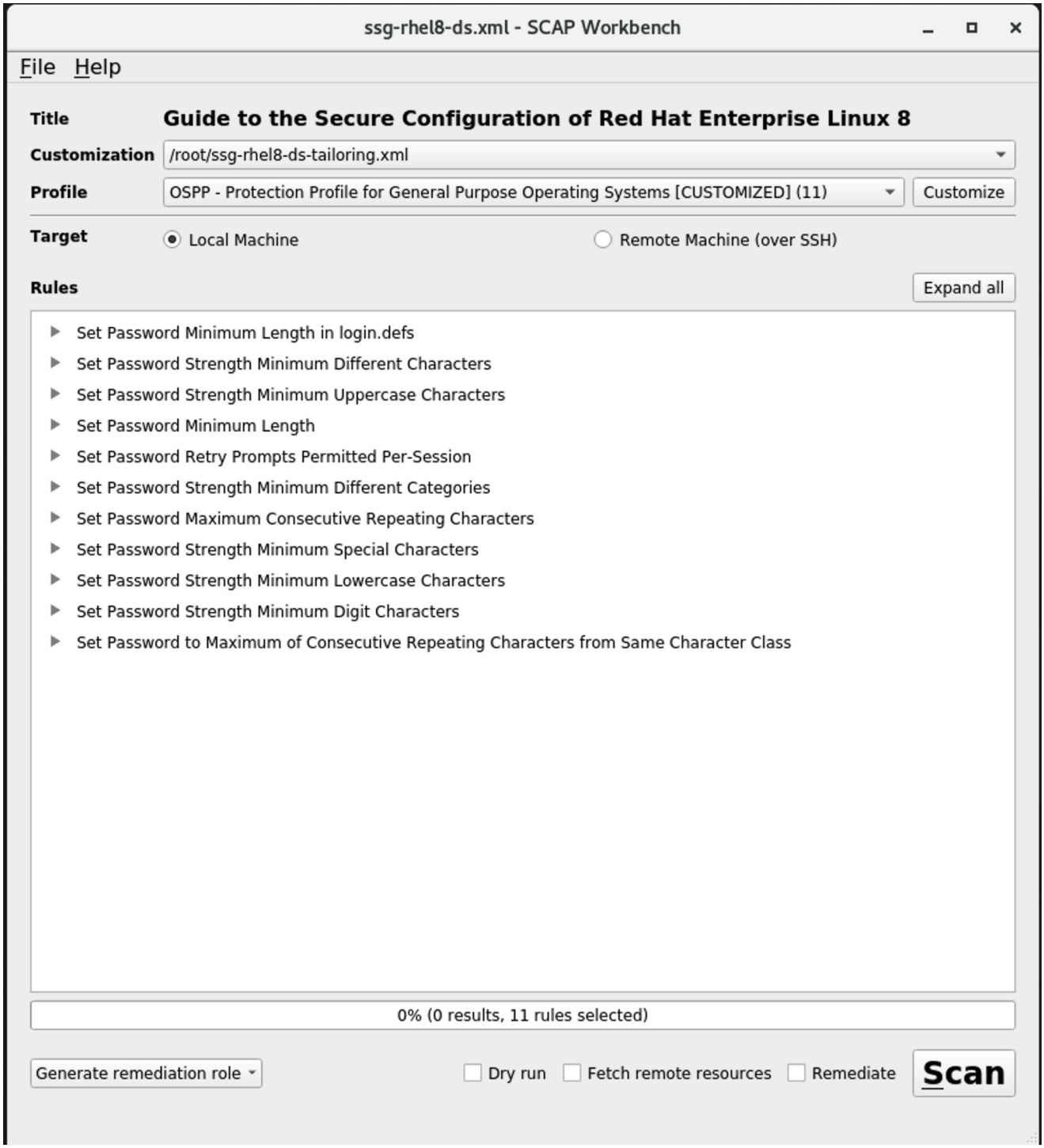

Figure 19. SCAP Custom Profile Final View

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 52. 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 6. Connecting to the **servera** *VM*

Listing 53. 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 7. Install software on **servera**

Listing 54. 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 55. 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 56. 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 8. Scanning **servera**

Listing 57. 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 9. Creating a SCAP Report from a Results File

Listing 58. 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 59. 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 10. Transferring Results

Listing 60. 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 11. Viewing the SCAP Report

Listing 61. Opening the SCAP HTML Report with Firefox

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

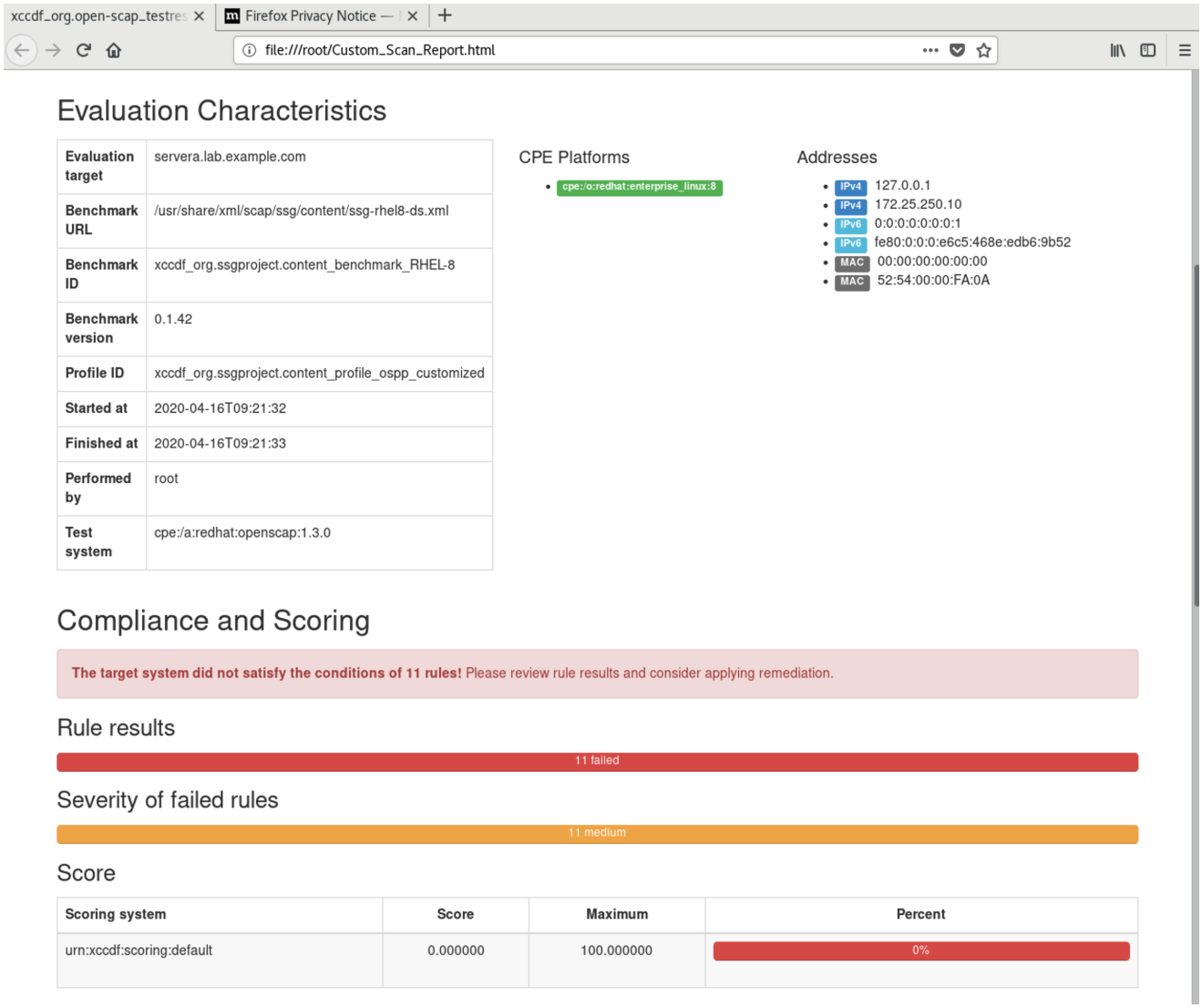

Figure 20. SCAP Scan Results Report in Firefox

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 12. Generating Ansible Playbook

Listing 62. 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 63. 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 13. Downloading Ansible Files

Listing 64. 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 14. Remediation of **serverc** with Ansible Playbook*

Listing 65. 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 15. Scanning System after Fixes and Verifying Results

Listing 66. 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 67. 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 68. Viewing Results on Workstation

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

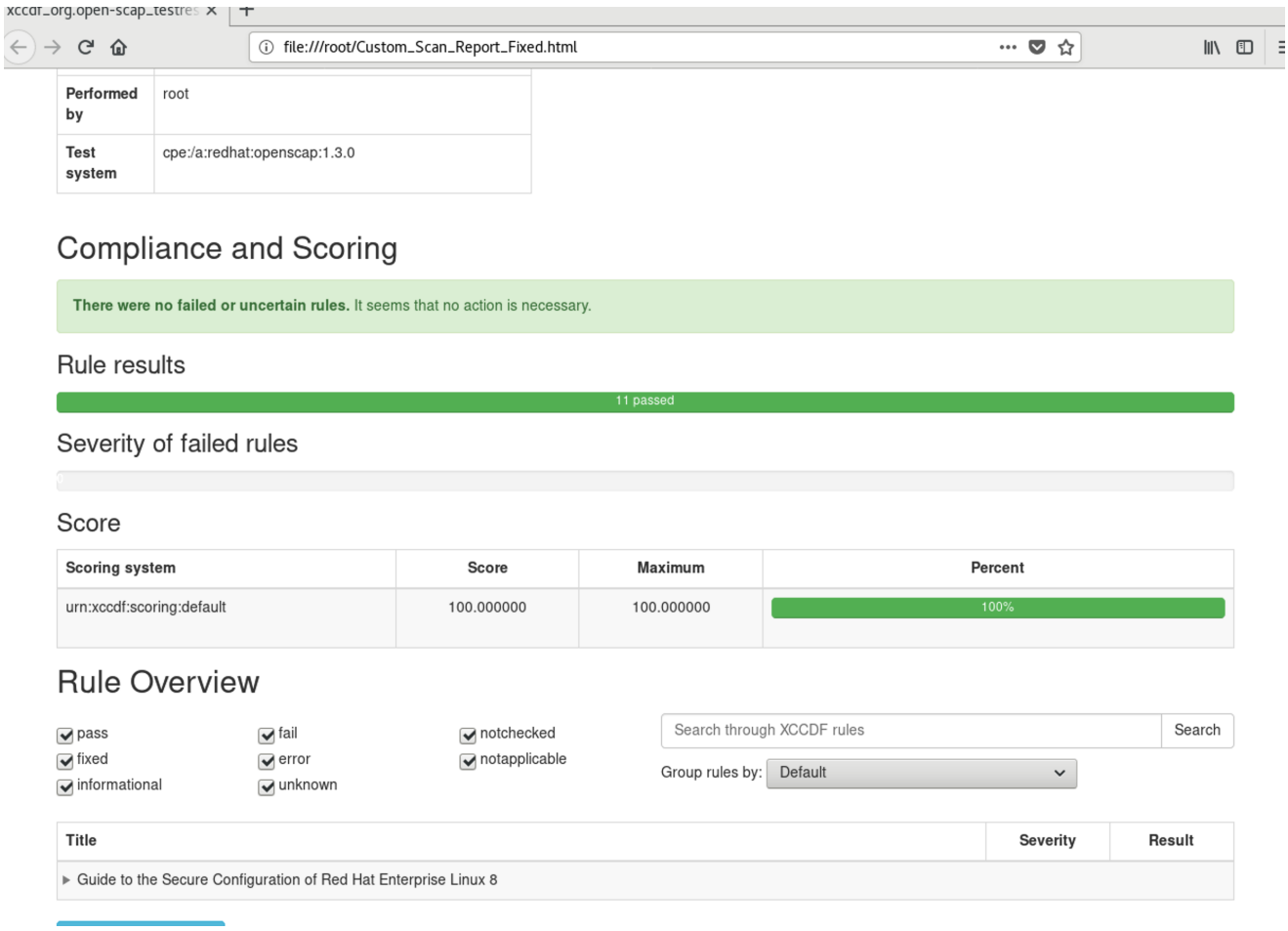

Figure 21. Fixed SCAP Scan Results Report in Firefox