Red Hat
Training and Certification

Student Workbook (ROLE)

Red Hat Enterprise Linux 8.0 RH354

RHEL 8 New Features for Experienced Linux Administrators

Edition 4

RH354-RHEL8.0-en-4-20200518

RHEL 8 New Features for Experienced Linux Administrators

Red Hat Enterprise Linux 8.0 RH354
RHEL 8 New Features for Experienced Linux Administrators
Edition 4 20200518
Publication date 20200518

Authors: Artur Glogowski, Morgan Weetman, Herve Quatremain, Trey Feagle, Adrian Andrade, Adolfo Vazquez
Editor: Philip Sweany, Steven Bonneville

Document Conventions

References

"References" describe where to find external documentation relevant to a subject.

Note

"Notes" are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

"Important" boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled "Important" will not cause data loss, but may cause irritation and frustration.

Warning

"Warnings" should not be ignored. Ignoring warnings will most likely cause data loss.

Introduction

RHEL 8 New Features for Experienced Linux Administrators

RHEL 8 New Features for Experienced Linux Administrators (RH354) is an introduction intended for experienced Linux system administrators to the key changes and new features in the upcoming Red Hat Enterprise Linux 8 release.

This course provides an orientation to Red Hat Enterprise Linux 8, to allow IT professionals with previous Red Hat Enterprise Linux 7 experience, including operators, managers, and principal system administrators, to be prepared for deployments and migrations to the upcoming major release of Red Hat Enterprise Linux.

Course Objectives

- Prepare for deployment of Red Hat Enterprise Linux 8 by learning about changes to key features of the operating system.
- Learn how to use key new features of Red Hat Enterprise Linux 8 in order to take advantage of new capabilities for your use cases.

Audience

- IT professionals experienced with Red Hat Enterprise Linux 7 and Linux system administration who are planning upgrades or deployment of Red Hat Enterprise Linux 8 systems.

Prerequisites

- Red Hat Certified System Administrator (EX200 / RHCSA) certification or equivalent Red Hat Enterprise Linux knowledge and experience. RHCE certification may be beneficial.

Orientation to the Classroom Environment

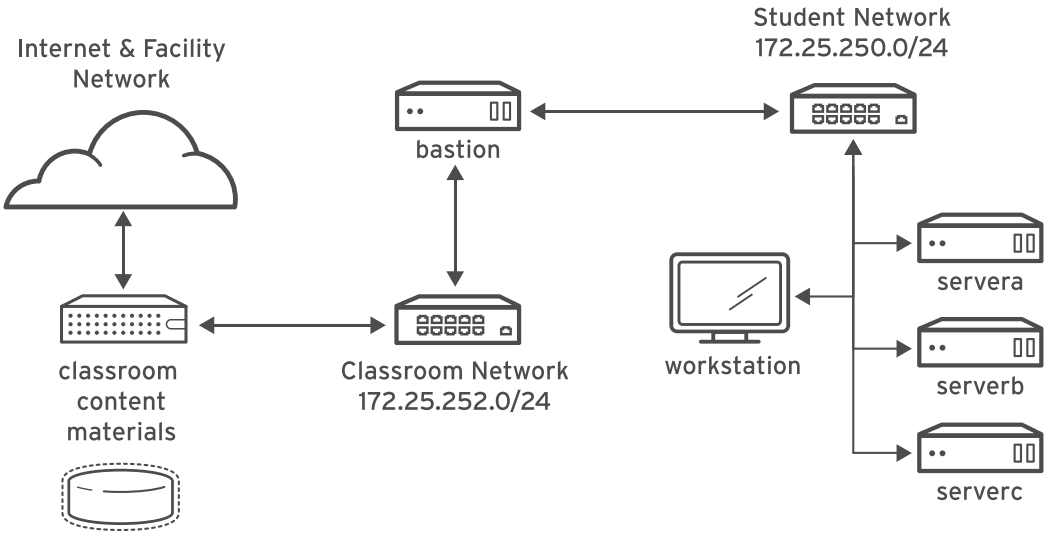


Figure 0.1: Classroom environment

In this course, the main computer system used for hands-on learning activities is **`workstation`**. Three other machines are also used by students for these activities: **`servera`**, **`serverb`**, and **`serverc`**. All four of these systems are in the **`lab.example.com`** DNS domain.

All student computer systems have a standard user account, **`student`**, which has the password **`student`**. The **`root`** password on all student systems is **`redhat`**.

Classroom Machines

Machine name	IP addresses	Role
bastion.lab.example.com	172.25.250.254	Gateway system to connect student private network to classroom server (must always be running)
workstation.lab.example.com	172.25.250.9	Graphical workstation used for system administration
servera.lab.example.com	172.25.250.10	Managed server "A"
serverb.lab.example.com	172.25.250.11	Managed server "B"
serverc.lab.example.com	172.25.250.12	Managed server "C"

The primary function of **`bastion`** is that it acts as a router between the network that connects the student machines and the classroom network. If **`bastion`** is down, other student machines will only be able to access systems on the individual student network.

Several systems in the classroom provide supporting services. Two servers, **content.example.com** and **materials.example.com**, are sources for software and lab materials used in hands-on activities. Information on how to use these servers is provided in the instructions for those activities. These are provided by the **classroom.example.com** virtual machine. Both **classroom** and **bastion** should always be running for proper use of the lab environment.

Note

When logging on to **servera** or **serverb** you might see a message concerning the activation of **cockpit**. The message can be ignored.

```
[student@workstation ~]$ ssh student@serverb
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of
 known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

[student@serverb ~]$
```

Controlling Your Systems

Students are assigned remote computers in a Red Hat Online Learning classroom. They are accessed through a web application hosted at rol.redhat.com [http://rol.redhat.com]. Students should log in to this site using their Red Hat Customer Portal user credentials.

Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through a web page. The state of each virtual machine in the classroom is displayed on the page under the **Online Lab** tab.

Machine States

Virtual machine state	Description
STARTING	The virtual machine is in the process of booting.
STARTED	The virtual machine is running and available (or, when booting, soon will be).
STOPPING	The virtual machine is in the process of shutting down.
STOPPED	The virtual machine is completely shut down. Upon starting, the virtual machine boots into the same state as when it was shut down (the disk will have been preserved).
PUBLISHING	The initial creation of the virtual machine is being performed.
WAITING_TO_START	The virtual machine is waiting for other virtual machines to start.

Depending on the state of a machine, a selection of the following actions is available.

Classroom/Machine Actions

Button or action	Description
PROVISION LAB	Create the ROL classroom. Creates all of the virtual machines needed for the classroom and starts them. This can take several minutes to complete.
DELETE LAB	Delete the ROL classroom. Destroys all virtual machines in the classroom. **Caution: Any work generated on the disks is lost.**
START LAB	Start all virtual machines in the classroom.
SHUTDOWN LAB	Stop all virtual machines in the classroom.
OPEN CONSOLE	Open a new tab in the browser and connect to the console of the virtual machine. Students can log in directly to the virtual machine and run commands. In most cases, students should log in to the **`workstation`** virtual machine and use **`ssh`** to connect to the other virtual machines.
ACTION → Start	Start (power on) the virtual machine.
ACTION → Shutdown	Gracefully shut down the virtual machine, preserving the contents of its disk.
ACTION → Power Off	Forcefully shut down the virtual machine, preserving the contents of its disk. This is equivalent to removing the power from a physical machine.
ACTION → Reset	Forcefully shut down the virtual machine and reset the disk to its initial state. **Caution: Any work generated on the disk is lost.**

At the start of an exercise, if instructed to reset a single virtual machine node, click **ACTION → Reset** for only the specific virtual machine.

At the start of an exercise, if instructed to reset all virtual machines, click **ACTION → Reset**

If you want to return the classroom environment to its original state at the start of the course, you can click **DELETE LAB** to remove the entire classroom environment. After the lab has been deleted, you can click **PROVISION LAB** to provision a new set of classroom systems.

Warning
The **DELETE LAB** operation cannot be undone. Any work you have completed in the classroom environment up to that point will be lost.

The Autostop Timer

The Red Hat Online Learning enrollment entitles students to a certain amount of computer time. To help conserve allotted computer time, the ROL classroom has an associated countdown timer, which shuts down the classroom environment when the timer expires.

To adjust the timer, click **MODIFY** to display the **New Autostop Time** dialog box. Set the number of hours and minutes until the classroom should automatically stop. Note that there is a maximum time of ten hours. Click **ADJUST TIME** to apply this change to the timer settings.

Internationalization

Per-user Language Selection

Your users might prefer to use a different language for their desktop environment than the system-wide default. They might also want to use a different keyboard layout or input method for their account.

Language Settings

In the GNOME desktop environment, the user might be prompted to set their preferred language and input method on first login. If not, then the easiest way for an individual user to adjust their preferred language and input method settings is to use the Region & Language application.

You can start this application in two ways. You can run the command **`gnome-control-center region`** from a terminal window, or on the top bar, from the system menu in the right corner, select the settings button (which has a crossed screwdriver and wrench for an icon) from the bottom left of the menu.

In the window that opens, select Region & Language. Click the **Language** box and select the preferred language from the list that appears. This also updates the **Formats** setting to the default for that language. The next time you log in, these changes will take full effect.

These settings affect the GNOME desktop environment and any applications such as **`gnome-terminal`** that are started inside it. However, by default they do not apply to that account if accessed through an **`ssh`** login from a remote system or a text-based login on a virtual console (such as **`tty5`**).

Note

You can make your shell environment use the same **`LANG`** setting as your graphical environment, even when you log in through a text-based virtual console or over **`ssh`**. One way to do this is to place code similar to the following in your **`~/.bashrc`** file. This example code will set the language used on a text login to match the one currently set for the user's GNOME desktop environment:

```
i=$(grep 'Language=' /var/lib/AccountsService/users/${USER} \
  | sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Japanese, Korean, Chinese, and other languages with a non-Latin character set might not display properly on text-based virtual consoles.

Individual commands can be made to use another language by setting the **`LANG`** variable on the command line:

```
[user@host ~]$ LANG=fr_FR.utf8 date
jeu. avril 25 17:55:01 CET 2019
```

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to determine the current value of **LANG** and other related environment variables.

Input Method Settings

GNOME 3 in Red Hat Enterprise Linux 7 or later automatically uses the IBus input method selection system, which makes it easy to change keyboard layouts and input methods quickly.

The Region & Language application can also be used to enable alternative input methods. In the Region & Language application window, the **Input Sources** box shows what input methods are currently available. By default, **English (US)** may be the only available method. Highlight **English (US)** and click the **keyboard** icon to see the current keyboard layout.

To add another input method, click the **+** button at the bottom left of the **Input Sources** window. An **Add an Input Source** window will open. Select your language, and then your preferred input method or keyboard layout.

When more than one input method is configured, the user can switch between them quickly by typing **Super**+**Space** (sometimes called **Windows**+**Space**). A *status indicator* will also appear in the GNOME top bar, which has two functions: It indicates which input method is active, and acts as a menu that can be used to switch between input methods or select advanced features of more complex input methods.

Some of the methods are marked with gears, which indicate that those methods have advanced configuration options and capabilities. For example, the Japanese **Japanese (Kana Kanji)** input method allows the user to pre-edit text in Latin and use **Down Arrow** and **Up Arrow** keys to select the correct characters to use.

US English speakers may also find this useful. For example, under **English (United States)** is the keyboard layout **English (international AltGr dead keys)**, which treats **AltGr** (or the right **Alt**) on a PC 104/105-key keyboard as a "secondary shift" modifier key and dead key activation key for typing additional characters. There are also Dvorak and other alternative layouts available.

Note

Any Unicode character can be entered in the GNOME desktop environment if you know the character's Unicode code point. Type **Ctrl**+**Shift**+**U**, followed by the code point. After **Ctrl**+**Shift**+**U** has been typed, an underlined **u** will be displayed to indicate that the system is waiting for Unicode code point entry.

For example, the lowercase Greek letter lambda has the code point U+03BB, and can be entered by typing **Ctrl**+**Shift**+**U**, then **03BB**, then **Enter**.

System-wide Default Language Settings

The system's default language is set to US English, using the UTF-8 encoding of Unicode as its character set (**en_US.utf8**), but this can be changed during or after installation.

From the command line, the **root** user can change the system-wide locale settings with the **localectl** command. If **localectl** is run with no arguments, it displays the current system-wide locale settings.

To set the system-wide default language, run the command **`localectl set-locale LANG=`*`locale`***, where *locale* is the appropriate value for the **LANG** environment variable from the "Language Codes Reference" table in this chapter. The change will take effect for users on their next login, and is stored in **`/etc/locale.conf`**.

```
[root@host ~]# localectl set-locale LANG=fr_FR.utf8
```

In GNOME, an administrative user can change this setting from Region & Language by clicking the **Login Screen** button at the upper-right corner of the window. Changing the **Language** of the graphical login screen will also adjust the system-wide default language setting stored in the **`/etc/locale.conf`** configuration file.

Important

Text-based virtual consoles such as **`tty4`** are more limited in the fonts they can display than terminals in a virtual console running a graphical environment, or pseudoterminals for **`ssh`** sessions. For example, Japanese, Korean, and Chinese characters may not display as expected on a text-based virtual console. For this reason, you should consider using English or another language with a Latin character set for the system-wide default.

Likewise, text-based virtual consoles are more limited in the input methods they support, and this is managed separately from the graphical desktop environment. The available global input settings can be configured through **`localectl`** for both text-based virtual consoles and the graphical environment. See the **`localectl`**(1) and **`vconsole.conf`**(5) man pages for more information.

Language Packs

Special RPM packages called *langpacks* install language packages that add support for specific languages. These langpacks use dependencies to automatically install additional RPM packages containing localizations, dictionaries, and translations for other software packages on your system.

To list the langpacks that are installed and that may be installed, use **`yum list langpacks-*`**:

```
[root@host ~]# yum list langpacks-*
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Installed Packages
langpacks-en.noarch      1.0-12.el8        @AppStream
Available Packages
langpacks-af.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-am.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-ar.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-as.noarch      1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
langpacks-ast.noarch     1.0-12.el8        rhel-8-for-x86_64-appstream-rpms
...output omitted...
```

To add language support, install the appropriate langpacks package. For example, the following command adds support for French:

```
[root@host ~]# yum install langpacks-fr
```

Use **`yum repoquery --whatsupplements`** to determine what RPM packages may be installed by a langpack:

```
[root@host ~]# yum repoquery --whatsupplements langpacks-fr
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:33 ago on Wed 06 Feb 2019 10:47:24 AM CST.
glibc-langpack-fr-0:2.28-18.el8.x86_64
gnome-getting-started-docs-fr-0:3.28.2-1.el8.noarch
hunspell-fr-0:6.2-1.el8.noarch
hyphen-fr-0:3.0-1.el8.noarch
libreoffice-langpack-fr-1:6.0.6.1-9.el8.x86_64
man-pages-fr-0:3.70-16.el8.noarch
mythes-fr-0:2.3-10.el8.noarch
```

Important

Langpacks packages use RPM *weak dependencies* in order to install supplementary packages only when the core package that needs it is also installed.

For example, when installing *langpacks-fr* as shown in the preceding examples, the *mythes-fr* package will only be installed if the *mythes* thesaurus is also installed on the system.

If *mythes* is subsequently installed on that system, the *mythes-fr* package will also automatically be installed due to the weak dependency from the already installed *langpacks-fr* package.

References

`locale`(7), **`localectl`**(1), **`locale.conf`**(5), **`vconsole.conf`**(5), **`unicode`**(7), and **`utf-8`**(7) man pages

Conversions between the names of the graphical desktop environment's X11 layouts and their names in **`localectl`** can be found in the file **`/usr/share/X11/xkb/rules/base.lst`**.

Language Codes Reference

Note

This table might not reflect all langpacks available on your system. Use **`yum info langpacks-`*`SUFFIX`*** to get more information about any particular langpacks package.

Language Codes

Language	Langpacks Suffix	$LANG value
English (US)	en	en_US.utf8

Language	Langpacks Suffix	$LANG value
Assamese	as	as_IN.utf8
Bengali	bn	bn_IN.utf8
Chinese (Simplified)	zh_CN	zh_CN.utf8
Chinese (Traditional)	zh_TW	zh_TW.utf8
French	fr	fr_FR.utf8
German	de	de_DE.utf8
Gujarati	gu	gu_IN.utf8
Hindi	hi	hi_IN.utf8
Italian	it	it_IT.utf8
Japanese	ja	ja_JP.utf8
Kannada	kn	kn_IN.utf8
Korean	ko	ko_KR.utf8
Malayalam	ml	ml_IN.utf8
Marathi	mr	mr_IN.utf8
Odia	or	or_IN.utf8
Portuguese (Brazilian)	pt_BR	pt_BR.utf8
Punjabi	pa	pa_IN.utf8
Russian	ru	ru_RU.utf8
Spanish	es	es_ES.utf8
Tamil	ta	ta_IN.utf8
Telugu	te	te_IN.utf8

Chapter 1

Previewing Red Hat Enterprise Linux 8

Goal

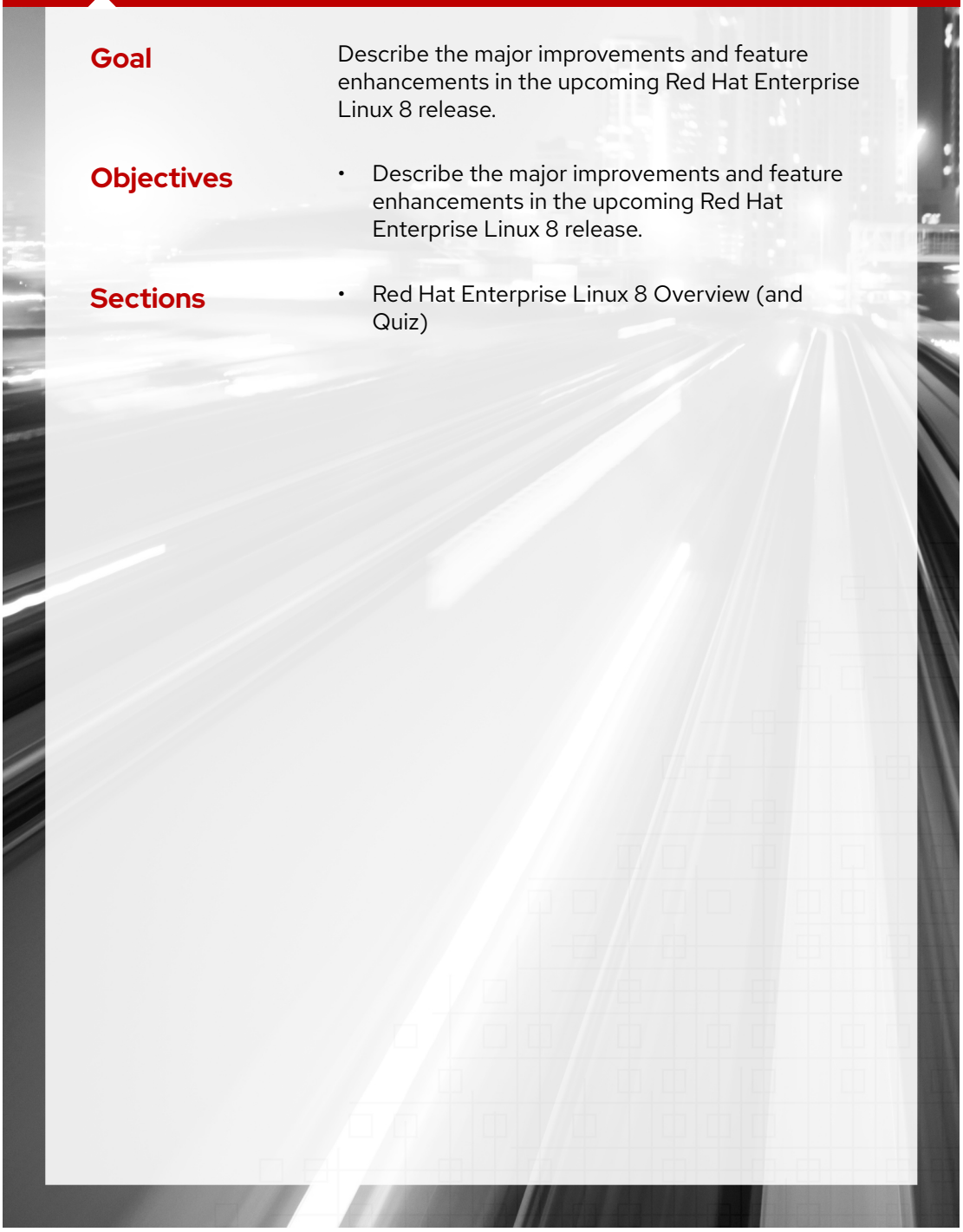
Describe the major improvements and feature enhancements in the upcoming Red Hat Enterprise Linux 8 release.

Objectives

- Describe the major improvements and feature enhancements in the upcoming Red Hat Enterprise Linux 8 release.

Sections

- Red Hat Enterprise Linux 8 Overview (and Quiz)



Red Hat Enterprise Linux 8 Overview

Objectives

After completing this section, you should be able to provide a basic overview of key features in Red Hat Enterprise Linux 8.

About Red Hat Enterprise Linux 8

Red Hat Enterprise Linux 8 (RHEL 8) is the latest major release of Red Hat's enterprise operating system. Deployable on physical hardware, virtual machines, in the cloud, and in containers, Red Hat Enterprise Linux 8 delivers advanced features required for next-generation IT architectures.

This course provides a hands-on opportunity to work with Red Hat Enterprise Linux 8, and explores selected features of this new version. The course authors have chosen these features based on their likely relevance to a wide audience. However, not every new feature in Red Hat Enterprise Linux 8 will be covered. For more information, review the product documentation and articles on http://access.redhat.com/, the Red Hat Customer Portal.

Supported Architectures

Red Hat Enterprise Linux 8 is supported on the following processor architectures:

Supported Processor Architectures

- 64-bit AMD/Intel (x86_64)
- 64-bit ARM (aarch64)
- IBM POWER, little endian (ppc64le)
- IBM Z (s390x)

Red Hat Enterprise Linux is no longer just about **x86_64** servers. Although **x86_64** dominates datacenter deployments, there are reasons to deploy Linux on alternative architectures and new applications that benefit from non-traditional computing architectures.

Red Hat's multi-architecture initiative provides software support for a variety of processor and machine architectures, while still powered by a common operating platform based on Red Hat Enterprise Linux. Customers can deploy systems on a variety of server designs while taking advantage of commonality of management and application compatibility.

ARMv8 Support

Support for 64-bit ARM was piloted in Red Hat Enterprise Linux 7 and is now supported as a core component of Red Hat Enterprise Linux 8:

RHEL 8 for ARM64 Support

- Red Hat Enterprise Linux for ARM (aarch64).
- 64-bit ARMv8 processors are supported.
- Targeted at server-optimized SoCs for cloud, hyperscale, telco, HPC, and edge computing.
- Goal is a single operating platform across multiple ARM suppliers.

Red Hat has been driving open standards in the ARM processor ecosystem for many years. Our goal has been to develop a single operating platform across multiple 64-bit ARMv8-A server-class *system-on-chip* (SoC) products designed for cloud, hyperscale, telco and edge computing, as well as high-performance computing (HPC) applications.

Note that 32-bit ARM processors (such as the armv7l architecture) are not supported.

Little Endian IBM POWER Support

Processor support for Red Hat Enterprise Linux 8 for IBM Power Systems has some changes:

RHEL 8 for IBM POWER

- RHEL 8 for IBM Power Systems supports little endian mode only (ppc64le).
- POWER8 and POWER9 processors are supported.
- Can be a KVM guest on Red Hat Virtualization for Power, PowerVM, and PowerNV (bare metal).

Older versions of RHEL supported IBM Power Systems using the ppc64 *big endian* architecture. Red Hat Enterprise Linux 8 is no longer being built or delivered for big endian. Only the *little endian* ppc64le mode is supported.

A little endian system stores data in memory ordered least-significant byte first and most-significant byte last, opposite to the behavior of a big-endian system. The x86_64 architecture is also little endian. The advantage to running POWER processors in little endian mode is that it makes it simpler for programmers to port applications from x86_64 to POWER. Using the same endian mode for all processors supported by Red Hat Enterprise Linux helps programmers avoid bugs that would cause errors in the representation of data in the ported application, or when one application exchanges data with another.

New Features and Changes

There are too many enhancements, changes, and new features in Red Hat Enterprise Linux to cover them all in this course. The overview that follows discusses many of them, some of which will be investigated in detail later in this course. This list is not exhaustive. For more information, visit http://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/ and review the *Release Notes* and other documentation for Red Hat Enterprise Linux 8.

Selected Changes in RHEL 8

- The following discussion is a review of selected changes in RHEL 8.
- Not every change or new feature in RHEL 8 will be covered in this course.
- For more information, see the *Release Notes* on https://access.redhat.com/.

Installation and Deployment

Some installer changes are discussed in detail later in this course. To summarize:

Overview of Installation Changes

- Restructure of channels into "BaseOS" and "AppStream".
- "System Purpose" to indicate planned purpose for entitlement and support level.
- Enhancements to Kickstart directives.
- Can install and boot from NVDIMM devices (using either Anaconda and Kickstart).

System Startup and Management

Changes to bootloader management:

BOOM Boot Manager

- Simplifies process of creating boot entries.
- Adds entries, does not modify them.
- Simplified CLI and API.

BOOM is a Linux boot manager that simplifies the boot entry creation. BOOM can boot LVM based snapshot system images. Existing boot loader configuration is not modified, since BOOM only inserts additional entries. The existing configuration is maintained and functions as before. BOOM has both a command-line interface and an API for administrators to create boot entries.

Enabling Secure-boot Guests

- RHEL 8 supports secure-boot guests which use cryptographically signed images.
- RHEL 8 relies on the Open Virtual Machine Firmware (OVMF) using the **`edk2`** codebase (*edk2-ovmf*) to provide support for secure firmware to virtual machines.
- Images are signed by trusted third-party organizations to ensure integrity.

Kernel

The current kernel supports 52-bit physical addressing for the 64-bit ARM architecture, 5-level paging, Control Group v2 mechanism, early kdump, the deadline process scheduler, and configuration of separate time namespaces.

Packaging Changes

- *kernel-core* provides the core kernel.
- *kernel-modules* and *kernel-modules-extra* contain kernel modules matching the *kernel-core* package version.
- *kernel* is now a meta-package that ensures *kernel-core* and *kernel-modules* are installed.

This new approach to kernel packaging allows for more granular and modular kernel configuration. It is especially important in today's cloud-based deployments, where space is of high importance.

Memory Management

- New 5-level paging model.
- 57-bit virtual memory addressing (128 PiB usable address space).
- 52-bit physical memory addressing (theoretically up to 4 PiB RAM).
- Actual physical support limits might vary depending on hardware.

Memory addressing capacity has been extended to 57/52-bit of virtual/physical memory. To handle the expanded address range, memory management was extended to use a 5-level page table implementation.

Enabling Early Kdump at Boot

- RHEL 7, and previous RHEL releases start the kdump.service as part of **`multi-user.target`** of the boot process. Problems that occur prior to this service event may not be able to be captured.
- RHEL 8 provides early Kdump, by storing the **`vmlinuz`** and **`initramfs`** of the crash kernel inside the **`initramfs`** of the booting kernel. These components are loaded directly into

reserved memory (crashkernel) during the early **`initramfs`** stage, allowing kernel crash dump capture during all phases of booting.

Enhancing the Process Scheduler

- The CFS process scheduler remains the default process scheduler in RHEL 8.
- CFS in RHEL 8 provides a new process scheduling class, **`SCHED_DEADLINE`**, which enables predictable task scheduling based on application deadlines.
- **`SCHED_DEADLINE`** is based on the Earliest Deadline First (EDF) and Constant Bandwidth Server (CBS) algorithms.
- **`SCHED_DEADLINE`** is suitable for real-time applications, such as multimedia or industrial control, and provides improved performance on machines with NUMA capabilities.
- Under **`SCHED_DEADLINE`**, processes use specific system calls to inform the scheduler of their estimated **`runtime`**, **`deadline`**, and **`period`**.

Networking

Enhancements and changes to networking features in RHEL 8 include:

Firewall Changes

- nftables is the default firewall backend.
- nftables is the successor to iptables, ip6tables, arptables, ebtables, and ipset.
- Still recommend using **`firewall-cmd`** to manage firewall; use **`nft`** directly only for complex configurations.
- iptables compatibility tools are available.

The iptables and ebtables toolsets are replaced by nftables in RHEL 8. The nftables framework, and its core tool **`nft`**, provides significant improvements in convenience, features, and performance. It unifies IPv4 and IPv6 tools, uses fast lookup tables instead of linear rule processing, supports debugging and tracing in the rule set, and has a more consistent and compact syntax. Compatibility tools based on the old command names are available, and tools to migrate rulesets from **`iptables`** and **`ip6tables`** to **`nft`** are documented in the xtables-translate(8) man page.

The **`firewalld`** system is still the recommended way to manage local firewall rules for typical use cases. In RHEL 8, it uses nftables as its default backend rule system. If the backend rule system changes again in a later Red Hat Enterprise Linux version, **`firewalld`** is designed to easily add and select a default amongst available rule back ends.

Firewall enhancements are covered in detail later in this course.

NetworkManager and Network Scripts

- **`nmcli`** is the preferred tool to manage network configuration through NetworkManager.
- New versions of **`ifup`** and **`ifdown`** require NetworkManager.
- Legacy network scripts like **`ifup-local`** are deprecated and not available by default.

NetworkManager enhancements are discussed in detail later in this course.

NTP Time Synchronization

- Chrony (**`chronyd`**) is the default NTP service implementation.
- Chrony performs better in a wide range of conditions, and synchronizes faster with better accuracy.
- Migration tools are available in **`/usr/share/doc/chrony/`**.

In RHEL 8, Chrony is the default NTP implementation provided by the operating system. The ntpd package is no longer available.

Chrony performs better than **`ntpd`** in many real-world scenarios: when access to the time reference is intermittent, when the network is frequently congested, or when the system clock is subject to sudden changes in tick rate (due to changes in the temperature of the crystal oscillator). It has a number of other enhancements as well.

A **`/usr/share/doc/chrony/ntp2chrony.py`** migration script is available to convert configurations from **`ntpd`** to Chrony. Details on how to use it are provided in the RHEL 8 documentation at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#migrating-to-chrony_using-chrony-to-configure-ntp.

Chrony is discussed in detail later in this course.

Enhancements to TCP

- Update to version 4.16 of the TCP stack.
- Improved performance for TCP server with high ingress connection rate.
- New BBR and NV congestion control algorithms.

Removed Network Drivers

- Certain obsolete network drivers have been removed or are no longer supported in RHEL 8.
- The **`e1000`** driver is no longer supported, but **`e1000e`** is still supported.
- The **`tulip`** driver has been removed, impacting "Generation 1" VMs on Microsoft Hyper-V.

Obsolete network drivers have been removed from RHEL 8 and are no longer supported. Review the *Removed hardware support* section of the release notes to see which ones are affected. Two particular drivers were widely used and may impact old hardware or virtualization environments.

The **`e1000`** driver for PCI-based Intel Gigabit Ethernet devices is no longer supported. This common family of network cards was the default driver for KVM virtual machines using the **`i440fx`** machine type. Red Hat now recommends the **`q35`** machine type in RHEL 8.

Important

The related **`e1000e`** driver for PCIe-based Intel Gigabit Ethernet devices is still available and supported.

The **`tulip`** driver for the DEC "Tulip" chip Fast Ethernet device family has been removed. Microsoft Hyper-V "Generation 1" virtual machines emulate this device for their virtual network adapter. Since RHEL 8 does not provide a driver for that device, PXE installation of RHEL 8 on a Hyper-V "Generation 1" virtual machine will fail. The workaround is to use "Generation 2" virtual machines, or to install the virtual machine using an ISO image.

Software Management

A new Yum feature is the package grouping method called *modules*.

Modules

- Modules are installed independently of the underlying Operating System major version.
- The module system supports multiple versions of an application simultaneously.
- A module is tied to an application stream.

Modularity and management of module streams with **`yum`** are covered in detail later in this course.

Updates to RPM and YUM

- DNF is a technology rewrite of Yum and is the new standard package management functionality for RPM packages in RHEL 8.
- The **`yum`** command (v4) is retained as the recommended command-line utility, symbolically linked to **`dnf`** to facilitate backward script and operator compatibility.
- Yum v4 supports modules that enable software modularity.
- Yum v4 now understands weak and boolean dependencies.
- Yum v4 provides a broad collection of plug-ins and add-on tooling.

Storage

New local storage manager provides volume-managed file system using shared pools of storage.

Stratis Storage Manager

- Ability to create pools of one or multiple block devices.
- Create dynamic and flexible file systems within those pools.
- Stratis supports file system snapshotting. Snapshots are independent from the source file systems.

Stratis Storage Manager is discussed in detail later in this course.

Virtual Data Optimizer (VDO)

- VDO reduces data footprint on storage on three phases: zero-block elimination, deduplication of redundant blocks, and data compression.
- VDO removes blocks which only include zeros, and keeps their metadata.
- Virtual disks for virtual machines are a good use case of VDO volumes.

VDO is discussed in detail later in this course.

XFS Copy-on-Write Extents

- *Copy-on-Write* (CoW) is enabled by default when file system is created.
- Adds support to XFS to allow two or more files to share the same data blocks.
- If one file changes, sharing is broken and separate blocks are tracked.
- Efficient file cloning, per-file snapshots, and operation of NFS and Overlayfs.
- RHEL 7 can only mount XFS with CoW extents in read-only mode.

The shared copy-on-write data extents functionality allows multiple files to share a common set of data blocks.

Supporting SCSI-3 Persistent Reservations with Virtio-SCSI

- On RHEL 8, both qemu and libvirt support SCSI-3 persistent reservations on storage-devices presented to VMs through Virtio-SCSI backed by direct-attached LUNs.
- Virtual machines can share Virtio-SCSI storage devices and use SCSI-3 PRs to control access.
- Storage devices managed with device-mapper-multipath can be passed through to VMs to use SCSI-3 PRs, and the host manages PR actions across all paths.

Other Storage Features

- New LUKS2 on-disk format for encrypted storage replaces LUKS1.

- Block devices now use multiqueue scheduling and the **`scsi-mq`** driver is enabled by default for better SSD performance.

Removed Storage Features

- Some old storage drivers have been entirely removed.
- Some old storage drivers are no longer supported, but they are still available.
- The Btrfs file system has been removed.
- Software-managed *Fibre Channel over Ethernet* (FCoE) support has been removed.

Important

Offloading FCoE adapters that appear as Fibre Channel adapters in the operating system and do not use the fcoe-utils management tools continue to be supported. This applies to select adapters from the **`lpfc`** and **`qla2xxx`** drivers.

Offloading FCoE adapters that do use the fcoe-utils management tools, but have their own drivers instead of **`fcoe.ko`** and manage DCBX configuration in their drivers or firmware, are also still supported unless Red Hat notes otherwise. This includes the **`fnic`**, **`bnx2fs`**, and **`qedf`** drivers.

Security Features

Many new or enhanced Security related features are introduced in RHEL 8. Important changes include:

System-wide Cryptographic Policy

- System-wide crypto policy for TLS/IPSec/SSH/DNSSEC/Kerberos.
- Allows admin to update list of protocols and algorithms to follow recommended practice for many services consistently.
- Several policies are provided and may be applied using the **`update-crypto-policies`** command.
- **`DEFAULT`** policy provides reasonable default compatible with PCI-DSS.

For more information on system-wide cryptographic policies, see Consistent security by crypto policies in Red Hat Enterprise Linux 8 [https://www.redhat.com/en/blog/consistent-security-crypto-policies-red-hat-enterprise-linux-8] and the **`update-crypto-policies`**(8) man page.

Improving sosreport Capabilities

- RHEL 8 includes the 3.6 version of the sosreport tool, which supports new profiles, for example containers, user and policy defined command line presets, and size limits for external commands
- This version of sosreport also supports a large collection of new plugins, for example **`ansible`**, **`buildah`**, and **`runc`**
- The sos-collector utility is available in RHEL 8, and collects sosreports from multi-host environments like a RHV cluster, or a RHEL High Availability cluster

Other Security Changes

- Audit subsystem updated to version 3.0.
- rsyslog update to 8.37.0 with additional new features and fixes.
- OpenSSH 7.8p1 rebase, removed support for weak ciphers and the obsolete SSH version 1 protocol.

- OpenSCAP CLI improvements, draft OSPP profile version 4.2 for RHEL 8.
- **`tcp_wrappers`** support has been removed.

User Environment

GNOME is now the only available display manager. KDE has been completely removed from the distribution. Wayland is the default display server, providing multiple advantages and improvements, including a stronger security model, improved multi-monitor handling, and improved user interface scaling.

Graphical Desktop Changes

- Wayland is the default display server; Xorg is still available.
- Updated to GNOME 3.28.
- KDE removed from the distribution.

Virtualization

In this release, QEMU can now emulate the Intel Q35 motherboard chipset, which offers a better hardware platform for modern virtualized operating systems.

Virtual Machine Management

- Packages for virtualization are in the **`virt`** module stream.
- New interface in the web console to manage virtual machines.
- **`virt-manager`** is deprecated but still available.

Using the web console to create and manage virtual machines is discussed in detail later in this course.

Updated KVM Hardware Model

- KVM now defaults to Q35 hardware model (newer hardware emulation).
- Better support of PCI Express passthrough, supports secure boot.
- The previous Intel 440FX emulation is still available for older operating systems.
- Similar concept to "Generation 1" and "Generation 2" virtual machines in Microsoft Hyper-V.

Selecting the KVM hardware model to use for virtual machines is discussed in detail later in this course.

Updates to HA Clustering

- Pacemaker upgraded to version 2.0.0
- **`pcs`** has new features, Corosync 3 support, some syntax changes
- A detailed list of changes is available at https://access.redhat.com/articles/3681151/.

Linux Containers

RHEL 8 includes a new package module which provides a new container engine, named Podman, to replace Docker and Moby. In contrast to Docker, which depends on daemons to build and run containers, this new toolset and container engine allow building and running containers without daemons.

New Container Tools

- The Podman container engine is daemonless and supports the execution of containers.

- Buildah supports the build of containers images, including building those images from scratch, or from a Dockerfile.
- You can copy and inspect container images in registries with Skopeo.

Containers are discussed in detail later in this course.

References

sched(7) man page.

`/usr/share/doc/edk2-ovmf/README`

For more information, refer to the *RHEL 8.0 Release Notes* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/8.0_release_notes/

Knowledgebase: "What is early kdump support and how do I configure it?"
https://access.redhat.com/solutions/3700611

Knowledgebase: "Pacemaker 2.0 upgrade in Red Hat Enterprise Linux 8"
https://access.redhat.com/articles/3681151

Consistent security by crypto policies in Red Hat Enterprise Linux 8
https://www.redhat.com/en/blog/consistent-security-crypto-policies-red-hat-enterprise-linux-8

Five-level page tables
https://lwn.net/Articles/717293

Zero-copy networking
https://lwn.net/Articles/726917

Boom Boot Manager
https://media.readthedocs.org/pdf/boom/latest/boom.pdf

Deadline Task Scheduling
https://www.kernel.org/doc/Documentation/scheduler/sched-deadline.txt

chrony: Comparison of NTP implementations
https://chrony.tuxfamily.org/comparison.html

BBR: Congestion-Based Congestion Control
https://queue.acm.org/detail.cfm?id=3022184

TCP-NV: An Update to TCP-Vegas
https://docs.google.com/document/d/1o-53jbO_xH-m9g2YCgjaf5bK8vePjWP6Mk0rYiRLK-U

Linux Multi-Queue Block IO Queueing Mechanism (blk-mq)
https://www.thomas-krenn.com/en/wiki/Linux_Multi-Queue_Block_IO_Queueing_Mechanism_(blk-mq)

Sysreport sos-3.6
https://github.com/sosreport/sos/releases

Quiz

Red Hat Enterprise Linux 8 Overview

Choose the correct answers to the following questions:

1. Which two of the following statements are true? (Choose two)

a. nftables is the default firewalld backend.
b. Btrfs is the default file system, and has been updated and enhanced.
c. Yum v4 supports modules that enable software modularity.
d. The e1000 driver for PCI-based Intel Gigabit Ethernet devices has been backported into Red Hat Enterprise Linux 8.

2. What four processor architectures does Red Hat Enterprise Linux 8 support? (Choose four)

a. 64-bit AMD/Intel (x86_64)
b. 32-bit AMD/Intel (x86)
c. 64-bit ARM (aarch64)
d. 32-bit ARM (aarch32)
e. IBM POWER, little endian (ppc64le)
f. IBM Z (s390x)

3. Which two statements are true about the new memory management in Red Hat Enterprise Linux 8? (Choose two)

a. Memory addressing capacity has been extended to 52/57-bit of virtual/physical memory.
b. The theoretical maximum physical memory addressing is up to 8 PiB RAM.
c. Memory addressing capacity has been extended to 57/52-bit of virtual/physical memory.
d. The theoretical maximum physical memory addressing is up to 4 PiB RAM.

4. Which of the following three deadline process scheduler parameters are supported in RHEL 8? (Choose three)

a. deadline
b. period
c. deadline_time
d. runtime

Solution

Red Hat Enterprise Linux 8 Overview

Choose the correct answers to the following questions:

1. Which two of the following statements are true? (Choose two)

a. nftables is the default firewalld backend.
b. Btrfs is the default file system, and has been updated and enhanced.
c. Yum v4 supports modules that enable software modularity.
d. The e1000 driver for PCI-based Intel Gigabit Ethernet devices has been backported into Red Hat Enterprise Linux 8.

2. What four processor architectures does Red Hat Enterprise Linux 8 support? (Choose four)

a. 64-bit AMD/Intel (x86_64)
b. 32-bit AMD/Intel (x86)
c. 64-bit ARM (aarch64)
d. 32-bit ARM (aarch32)
e. IBM POWER, little endian (ppc64le)
f. IBM Z (s390x)

3. Which two statements are true about the new memory management in Red Hat Enterprise Linux 8? (Choose two)

a. Memory addressing capacity has been extended to 52/57-bit of virtual/physical memory.
b. The theoretical maximum physical memory addressing is up to 8 PiB RAM.
c. Memory addressing capacity has been extended to 57/52-bit of virtual/physical memory.
d. The theoretical maximum physical memory addressing is up to 4 PiB RAM.

4. Which of the following three deadline process scheduler parameters are supported in RHEL 8? (Choose three)

a. deadline
b. period
c. deadline_time
d. runtime

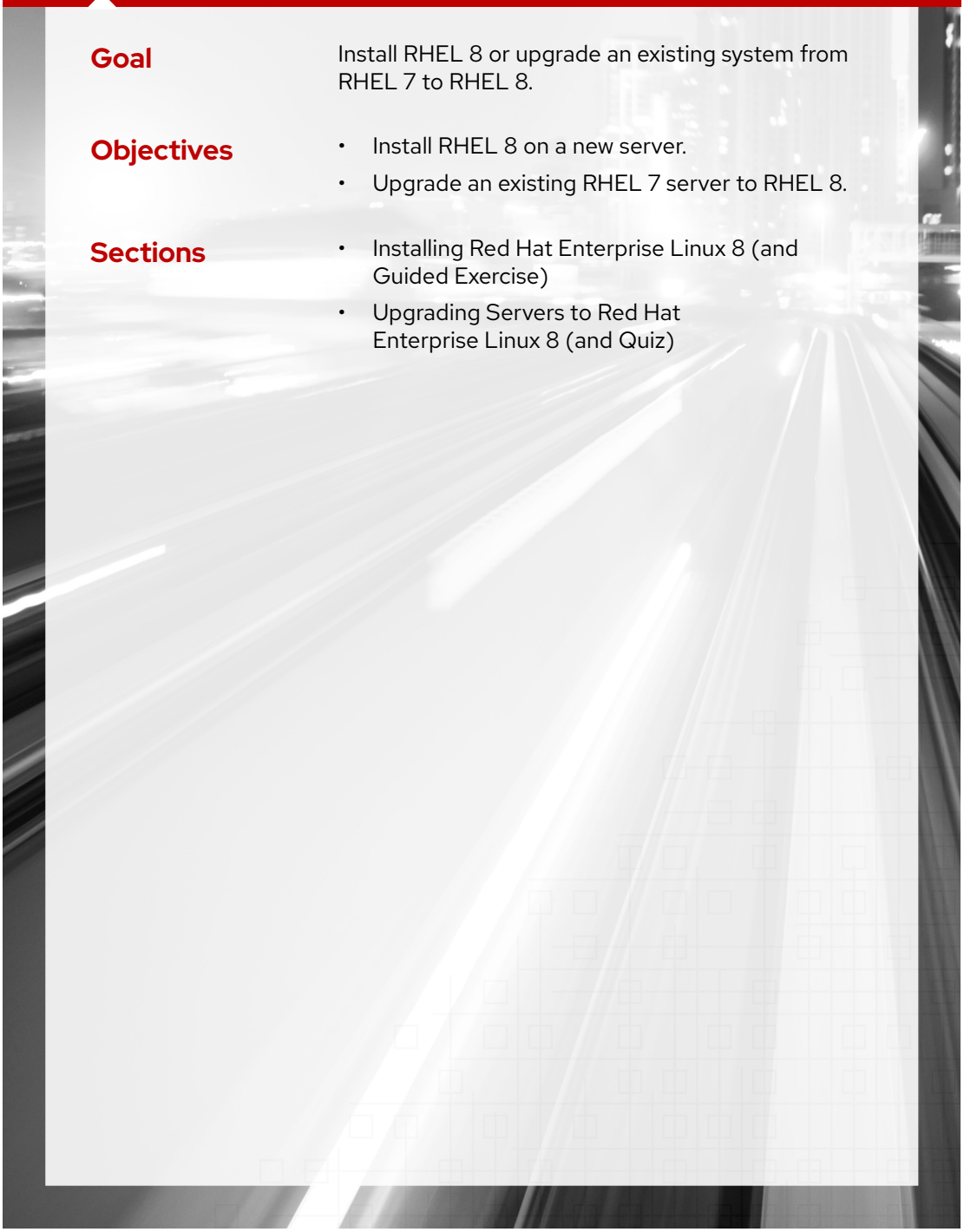
Summary

In this chapter, you learned:

- Red Hat's multi-architecture initiative is about providing software support for a variety of processor and machine architectures powered by a common operating platform built on Red Hat Enterprise Linux.
- The updated kernel supports, among others, 52-bit physical addressing for the 64-bit ARM architecture, 5-level paging, and Control Group v2 mechanism.
- nftables replaces iptables as the firewalld back end.
- **`nmcli`** is the preferred tool to manage network configuration.
- Chrony (**`chronyd`**) completely replaces **`ntpd`** as the NTP implementation.
- A number of obsolete network drivers have been removed from RHEL 8.
- One of the new features for Yum is the package grouping method called **`modules`**.
- KVM now defaults to the Q35 hardware model (providing newer hardware emulation).

Chapter 2

Installing or Upgrading to Red Hat Enterprise Linux 8



Goal

Install RHEL 8 or upgrade an existing system from RHEL 7 to RHEL 8.

Objectives

- Install RHEL 8 on a new server.
- Upgrade an existing RHEL 7 server to RHEL 8.

Sections

- Installing Red Hat Enterprise Linux 8 (and Guided Exercise)
- Upgrading Servers to Red Hat Enterprise Linux 8 (and Quiz)

Installing Red Hat Enterprise Linux 8

Objectives

After completing this section, students should be able to install RHEL 8 on a new server.

Repository Structure

Red Hat Enterprise Linux has changed the way packages are divided into channels or repositories. Packages are reorganized into Base Operating System (BaseOS) and Application Stream (AppStream) repositories. Special purpose repositories are also restructured to support the new AppStream and modular contexts.

RHEL 8 Repositories and Contents

- BaseOS: packages required for a minimal operating system installation.
- AppStream: most other packages, except those requiring add-on entitlements.
- CodeReady Linux Builder: packages providing programming language build-time dependencies; used by application developers.
- The Extras and Optional repositories have been removed. AppStream design makes additional repositories unnecessary. Packages were moved to AppStream or dropped. Appropriate dropped packages might end up in EPEL.
- Supplemental: exists and retains the same role in RHEL 8.
- There are add-on repositories for High Availability, Resilient Storage, Real Time, and Real Time for NFV. Add-on products require additional entitlements.

Installation Media Structure

The installation media choices determine the method for initiating an installation. The media choices remain the same as in previous RHEL versions. The repository changes determine from where packages are installed. The modularity of AppStream changes how packages are grouped and layered on the Base OS.

Available Installation Media Formats

- Netboot ISO: contains only the Anaconda installer. This method requires a configured network to access package repositories available by HTTP, FTP, or NFS.
- Binary DVD: contains Anaconda, and both the BaseOS and AppStream repositories.
- QCOW2 image: prebuilt system disk ready to deploy as a VM in cloud or virtual environments.

Package Modularity

Red Hat Enterprise Linux 8 brings some additions in package management capabilities.

Modules

Yum uses a new package grouping method called *modules*. A module is a set of RPM packages that form an application profile.

Module Features

- Modules can be installed independently of the underlying Operating System major version.
- The module system design supports multiple application versions simultaneously.
- Modules can contain several streams to make multiple application versions available for installation.
- A stream represents an application version substantially different from previous versions.
- A stream may be created when a newer application version is not backward compatible.
- Installing a stream wholly replaces the previously installed stream, whether as an upgrade or downgrade. Contrast this with a legacy package upgrade which adds or removes only changed package files.

For example, you could install the Perl 5.24 stream, and even though Perl 5.26 packages are available, no updates will be listed or applied. You could switch to the Perl 5.26 stream at any time. The module feature replaces Red Hat Software Collections.

The full module naming specification takes the form **`name`**:**`stream`**:**`version`**:**`context`**:**`architecture`**/**`profile`**. Shorter specifications can be used when the default values are acceptable.

Module Naming Specification

- **`name`**
- **`name`**:**`stream`**
- **`name`**:**`stream`**:**`version`**
- Any of the above specifications with ::**`architecture`** appended.
- **`name`**:**`stream`**:**`version`**:**`context`**:**`architecture`**
- Any of the above specifications with /**`profile`** appended.

Describing the System Purpose Configuration

The System Purpose configuration dialog box is a new installation feature that allocates accurate system entitlements to match the intended system use.

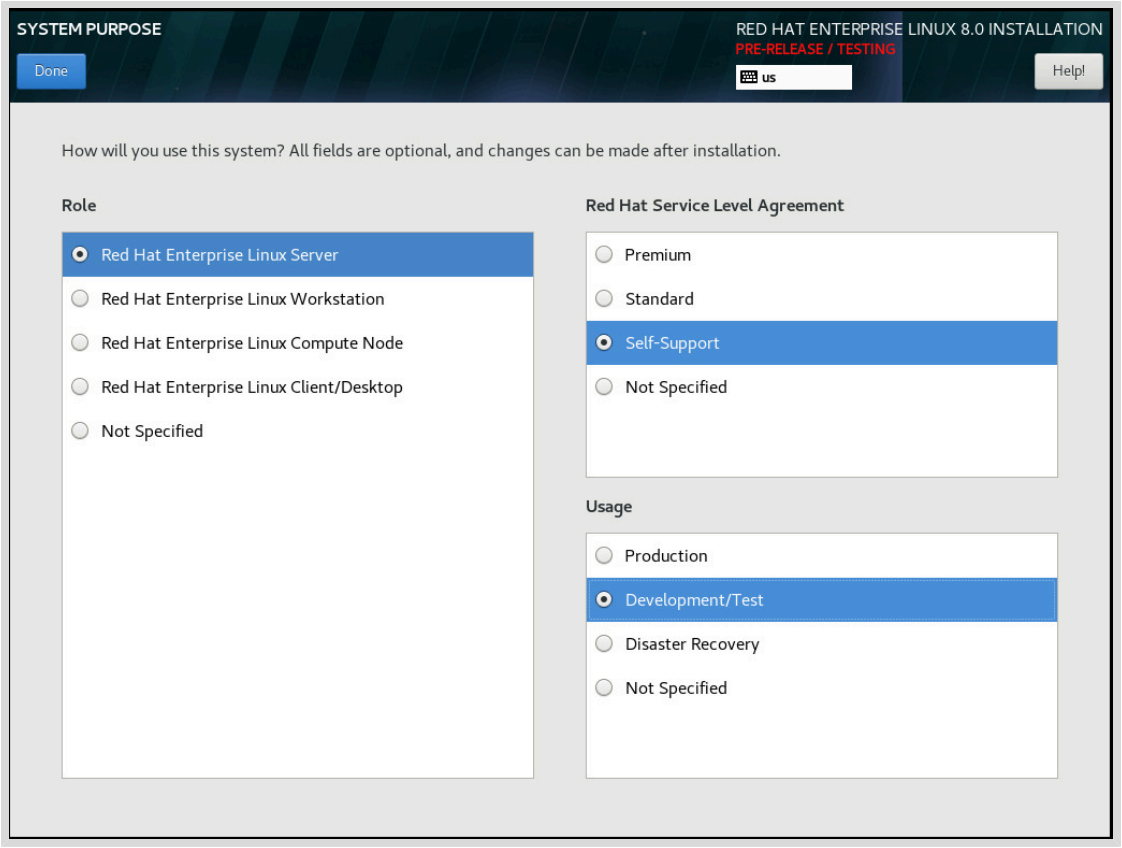


Figure 2.1: The System Purpose dialog box

System Purpose Dialog Box

- **`Role`**: choice of server, workstation, compute node, or client.
- **`Service Level Agreement`**: the level of support this system needs.
- **`Usage`**: choice of production, development, or disaster recovery.
- These choices determine how subscription attachments and entitlement consumption are handled during installation.

System Purpose Kickstart Directives

- **`syspurpose --role=`**
- **`syspurpose --sla=`**
- **`syspurpose --usage=`**

References

For more information, refer to the *Performing a standard RHEL installation* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/performing_a_standard_rhel_installation

Introducing CodeReady Linux Builder
https://developers.redhat.com/blog/2018/11/15/introducing-codeready-linux-builder/

Introducing Application Streams in RHEL 8
https://developers.redhat.com/blog/2018/11/15/rhel8-introducing-appstreams/

Guided Exercise

Installing Red Hat Enterprise Linux 8

In this exercise, you will manually install RHEL 8 on a clean virtual machine using Anaconda.

Outcomes

You should be able to manually install Red Hat Enterprise Linux 8.

Before You Begin

The installation media ISO has already been attached to **`serverc`**.

Log in to **`workstation`** as **`student`** using **`student`** as the password.

On **`workstation`**, run the **`lab install-installation start`** command.

```
[student@workstation ~]$ lab install-installation start
```

1. Locate the icon for the **`serverc`** console, as appropriate for your classroom environment. Power on **`serverc`** and open the console for it. The server is uninstalled, and is preconfigured to boot from a virtual installation boot ISO.

2. From the installation media boot menu, use the arrow keys to select **Install Red Hat Enterprise Linux 8.0**.

3. Select an appropriate language from those available, and then click **Continue**.

Important

The installation source for this exercise is over the network from the **`content.example.com`** system. Network parameters must be configured properly before the installation source will become available in a later step.

4. Use automatic partitioning on the default disk.

   4.1. Click **Installation Destination**.

   4.2. Click **Done** to use the default option of automatic partitioning.

5. Set the server host name to **`serverc.lab.example.com`**, and enable and configure the only network interface.

   5.1. Click **Network & Host Name**.

   5.2. In the **Host Name** field enter **`serverc.lab.example.com`**.

   5.3. Click **Apply**.

   5.4. Click **Configure** and then click the **IPv4 Settings** tab.

5.5. Select **Manual**.

5.6. Click **Add**.

5.7. In the **Address** field enter **`172.25.250.12`**.

5.8. In the **Netmask** field enter **`255.255.255.0`**. As an alternative, you can specify the mask in the CIDR notation: **24**.

5.9. In the **Gateway** field enter **`172.25.250.254`**.

5.10. In the **DNS servers** field enter **`172.25.250.254`**.

5.11. Click **Save**.

5.12. Enable the network interface by setting **ON/OFF** to **`ON`**.

5.13. Click **Done**.

6. Set the **Installation Source** field to **`http://content.example.com/rhel8.0/x86_64/dvd`**.

6.1. Click **Installation Source**.

6.2. In the **http://** field type **`content.example.com/rhel8.0/x86_64/dvd`**.

6.3. Click **Done**.

7. Disable Kdump.

7.1. Click **KDUMP**.

7.2. Clear **Enable kdump**.

7.3. Click **Done**.

8. Select the software required to run a basic web server.

8.1. Click **Software Selection**.

8.2. Select **Server** from the Base Environment list.

8.3. Select **Basic Web Server** from the Add-ons list.

8.4. Click **Done**.

9. Configure the system's purpose.

9.1. Click **System Purpose**.

9.2. Select the **Red Hat Enterprise Linux Server** role.

9.3. Select an SLA level of **Self-Support**.

9.4. Select a usage of **Development/Test**.

9.5. Click **Done**.

10. Click **Begin Installation**.

11. While the installation progresses, set the password for **root** to **redhat**.

    11.1. Click **Root Password**.

    11.2. Enter **redhat** in the **Root Password** field.

    11.3. Enter **redhat** in the **Confirm** field.

    11.4. The password is weak so you will need to click **Done** twice.

12. While the installation progresses, add the **student** user.

    12.1. Click **User Creation**.

    12.2. Enter **student** in the **Full Name** field.

    12.3. Check **Make this user administrator**.

    12.4. Enter **student** in the **Password** field.

    12.5. Enter **student** in the **Confirm** field.

    12.6. The password is weak so you will need to click **Done** twice.

13. When the installation is complete, click **Reboot**.

14. If the system boots from the installation media again, press **Esc**, then type **local** and press **Enter**. Alternatively, select **Troubleshooting**, and then **Boot from local drive**.

15. When the system displays the login prompt, log in as **student** with a password of **student**.

Finish

On **workstation**, run **lab install-installation finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab install-installation finish
```

This concludes the guided exercise.

Upgrading Servers to Red Hat Enterprise Linux 8

Objectives

After completing this section, students should be able to upgrade an existing RHEL 7 server to RHEL 8.

Describing Leapp

Leapp is an extendable framework designed to assist administrators with application modernization. It supports Red Hat Enterprise Linux, CentOS, and Fedora, and is the preferred method for in-place upgrades from RHEL 7 to RHEL 8.

Leapp Modernization Framework

- Enables users to modernize existing workloads without disrupting them.
- Three methods: upgrade in place, migrate to new place, or containerize.
- Designed modular architecture to replace the preupgrade-assistant tool.
- Various migration strategies and application-specific logic are kept in independent modules or plug-ins.
- Leapp is message-driven, for passing data between actors. The execution of actors is dependent on the data produced by other actors running before them.

Leapp is the newly designed modular architecture to replace the preupgrade-assistant tool, whose architecture is too restrictive for the complexity of modern application upgrade requirements. Actors will specify what types of data they require and provide on their input and output ports respectively, and a dependency solver will produce a dependency graph to satisfy those requirements, that is, connect output ports to appropriate input ports. The implementation of actors is done in a shell, with a Python wrapper to present the proper interface for the (Python-based) framework. The shell parts are executed using Ansible (as Ansible modules) and use JSON for communication.

Information about various migration strategies and application-specific logic is kept in independent modules or plug-ins instead of single source files. Modules and plug-ins can be maintained by experts on a particular application, independently of the maintenance of the core Leapp application. Leapp solves preupgrade-assistant limitations by adding module dependencies and methods for information passing between modules. Leapp is message-driven, for passing data between actors. The execution of actors is dependent on the data produced by other actors running before them. This is in a contrast with Ansible where parameters have to be specified in the playbooks before execution.

Leapp Components

- **`Workflow`**: describes an entire upgrade process.
- **`Phase`**: a section of the workflow dedicated to a specific part of the upgrade.
- **`Stage`**: phases are broken into stages, Before, Main, and After.
- **`Actor`**: a step in the workflow that performs a task. Actors can be tagged which allows them to be included in a workflow automatically by the framework.
- **`Tag`**: allows the framework to locate and execute actors.
- **`Message`**: used to transfer information between actors.
- **`Model`**: defines the structure of the data sent in messages. Models use Topics to group the data.

- **`Topic`**: defines the subject area for Model data.
- **`Repository`**: holds the definitions for all Actors, Tags, Topics, and Workflows.

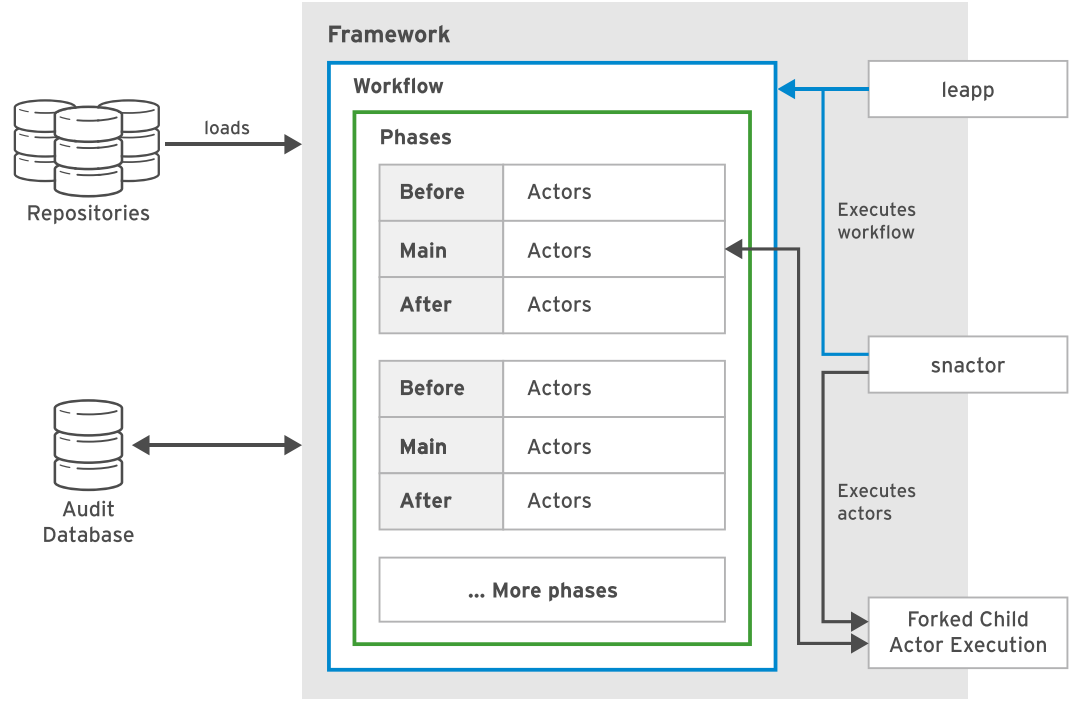


Figure 2.2: The Leapp architecture overview

Leapp Status

Leapp is under heavy development, and is not currently suitable for all upgrades.

Leapp Limitations

- Currently Leapp only works on minimal Red Hat Enterprise Linux 7 installations.
- Leapp only supports systems with single network interfaces; no bridges, bonds, or teams.
- Leapp only supports systems with simple storage configurations; no RAID, multipath, UEFI, or encryption.
- Leapp has no support for layered products; but modules can be written by any expert.
- The only supported architecture is x86_64.

Demonstration: System upgrade using Leapp

Leapp Demonstration

- You can view the video from **`workstation`** in both physical and online classrooms.
- Copy and paste this URL into the Firefox browser running on your **`workstation`** system:
- `http://materials.example.com/video/RH354-RHEL8.0-upgrade-using-leapp-mweetman-201902.webm`
- Watch this video as an instructor performs an upgrade from RHEL 7 to RHEL 8 using Leapp.

References

For more information, refer to the *Upgrading to RHEL 8* guide at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/upgrading_to_rhel_8

Leapp Documentation
https://leapp.readthedocs.io/en/latest/index.html

Leapp: OS and Application modernization framework
https://oamg.github.io/leapp/

Quiz

Upgrading Servers to Red Hat Enterprise Linux 8

Choose the correct answers to the following questions:

1. **Which three of the following are currently limitations of Leapp? (Choose three.)**
   a. Does not support bonded network interfaces.
   b. Does not support encrypted volumes.
   c. Only supports x86_64 and IBM Z series.
   d. No support for layered products.
   e. Only supports systems using packages from the rhel-7-server-rpms and rhel-7-server-optional-rpms channels.

2. **Which five of the following are components of Leapp? (Choose five.)**
   a. Subscriber
   b. Topic
   c. Stage
   d. Phase
   e. Consumer
   f. Actor
   g. Workflow

3. **Which three of the following are operating systems supported by Leapp? (Choose three)**
   a. Fedora
   b. SuSE
   c. CentOS
   d. Ubuntu
   e. Debian
   f. Red Hat Enterprise Linux

Solution

Upgrading Servers to Red Hat Enterprise Linux 8

Choose the correct answers to the following questions:

1. **Which three of the following are currently limitations of Leapp? (Choose three.)**
   a. Does not support bonded network interfaces.
   b. Does not support encrypted volumes.
   c. Only supports x86_64 and IBM Z series.
   d. No support for layered products.
   e. Only supports systems using packages from the rhel-7-server-rpms and rhel-7-server-optional-rpms channels.

2. **Which five of the following are components of Leapp? (Choose five.)**
   a. Subscriber
   b. Topic
   c. Stage
   d. Phase
   e. Consumer
   f. Actor
   g. Workflow

3. **Which three of the following are operating systems supported by Leapp? (Choose three)**
   a. Fedora
   b. SuSE
   c. CentOS
   d. Ubuntu
   e. Debian
   f. Red Hat Enterprise Linux

Lab

Installing or Upgrading to Red Hat Enterprise Linux 8

Performance Checklist

In this lab, you will install a Red Hat Enterprise Linux 8 system.

Outcomes

You should be able to install RHEL 8 from ISO installation media and customize as required.

Before You Begin

The installation media ISO has already been attached to **`serverc`**.

Log in to **`workstation`** as **`student`** using **`student`** as the password.

On **`workstation`**, run the **`lab install-review start`** command.

```
[student@workstation ~]$ lab install-review start
```

Note

If you performed the RHEL 8 installation guided exercise earlier in the chapter, reset the **`serverc`** virtual machine back to default.

1. Locate the icon for the **`serverc`** console, as appropriate for your classroom environment. Power on **`serverc`** and open the console for it. The server is uninstalled, and is preconfigured to boot from a virtual installation boot ISO.
2. From the installation media boot menu, use the arrow keys to select **Install Red Hat Enterprise Linux 8.0**.
3. Select an appropriate language from those available, and then click **Continue**.
4. Configure the system disk using LVM with a **`/ (root)`** volume of 10 GiB, **`/boot`** of 1 GiB, and **`/home`** using the remaining free space.
5. Set the server host name to **`serverc.lab.example.com`**, and enable and configure the only network interface. Add the additional search domain **`example.net`**.

   Use the following information:

Field	Value
IP Address	172.25.250.12
Netmask	255.255.255.0
Gateway	172.25.250.254
DNS server	172.25.250.254
Search Domain	example.net
Hostname	serverc.lab.example.com

6. Set the **Installation Source** to **`http://content.example.com/rhel8.0/x86_64/dvd`**.
7. Select the software required to run a graphical desktop with a basic development environment.
8. Configure the system's purpose as a Red Hat Enterprise Linux Workstation, with Self-support, for development usage.
9. Click **Begin Installation**.
10. While the installation progresses, set the password for **`root`** to **`redhat`**.
11. While the installation progresses, add the **`lab`** administrative user with a password of **`redhat`**.
12. When the installation is complete, click **Reboot**.
13. Ensure the system boots from the local drive.
14. Accept the End User License Agreement (EULA), then click **Continue**.
15. When the system displays the login prompt, log in as **`lab`** with a password of **`redhat`**.

Finish

On **`workstation`**, run **`lab install-review finish`** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab install-review finish
```

This concludes the lab.

Solution

Installing or Upgrading to Red Hat Enterprise Linux 8

Performance Checklist

In this lab, you will install a Red Hat Enterprise Linux 8 system.

Outcomes

You should be able to install RHEL 8 from ISO installation media and customize as required.

Before You Begin

The installation media ISO has already been attached to **`serverc`**.

Log in to **`workstation`** as **`student`** using **`student`** as the password.

On **`workstation`**, run the **`lab install-review start`** command.

```
[student@workstation ~]$ lab install-review start
```

Note

If you performed the RHEL 8 installation guided exercise earlier in the chapter, reset the **`serverc`** virtual machine back to default.

1. Locate the icon for the **`serverc`** console, as appropriate for your classroom environment. Power on **`serverc`** and open the console for it. The server is uninstalled, and is preconfigured to boot from a virtual installation boot ISO.
2. From the installation media boot menu, use the arrow keys to select **Install Red Hat Enterprise Linux 8.0**.
3. Select an appropriate language from those available, and then click **Continue**.
4. Configure the system disk using LVM with a **`/ (root)`** volume of 10 GiB, **`/boot`** of 1 GiB, and **`/home`** using the remaining free space.
   - 4.1. Click **Installation Destination**.
   - 4.2. Select **Custom**, and then click **Done**. This launches the **Manual Partitioning** dialog box.
   - 4.3. Ensure the partition scheme is set to **LVM**.
   - 4.4. Click **Click here to create them automatically** to generate the standard set of mount points.
   - 4.5. Select the **/** mount point, and change the **Desired Capacity** to 10 GiB.

4.6. The **/boot** mount point should not require any changes.

4.7. Click **+** to add a mount point. Enter **/home** for **Mount Point** or select it from the list. Leave **Desired Capacity** blank and the remaining free space should be allocated to it. Click **Add mount point**.

4.8. Click **Done** to display the **Summary of Changes** dialog box. Click **Accept Changes**.

5. Set the server host name to **serverc.lab.example.com**, and enable and configure the only network interface. Add the additional search domain **example.net**.

Use the following information:

Field	Value
IP Address	172.25.250.12
Netmask	255.255.255.0
Gateway	172.25.250.254
DNS server	172.25.250.254
Search Domain	example.net
Hostname	serverc.lab.example.com

5.1. Click **Network & Host Name**.

5.2. In the **Host Name** field enter **serverc.lab.example.com**.

5.3. Click **Apply**.

5.4. Click **Configure**, then select the **IPv4 Settings** tab.

5.5. Select **Manual**.

5.6. In the **Address** field enter **172.25.250.12**.

5.7. In the **Netmask** field enter **255.255.255.0**.

5.8. In the **Gateway** field enter **172.25.250.254**.

5.9. In the **DNS servers:** field enter **172.25.250.254**.

5.10. Add **example.net** in the **Search domains** field.

5.11. Click **Save**.

5.12. Enable the network interface by setting **ON/OFF** to **ON**.

5.13. Click **Done**.

6. Set the **Installation Source** to **http://content.example.com/rhel8.0/x86_64/dvd**.

6.1. Click **Installation Source**.

6.2. In the **http://** field enter **content.example.com/rhel8.0/x86_64/dvd**.

6.3. Click **Done**.

7. Select the software required to run a graphical desktop with a basic development environment.

   7.1. Click **Software Selection**.

   7.2. Select **Workstation** from the Base Environment list.

   7.3. Select **Development Tools** from the Add-ons list.

   7.4. Click **Done**.

8. Configure the system's purpose as a Red Hat Enterprise Linux Workstation, with Self-support, for development usage.

   8.1. Click **System Purpose**.

   8.2. Select a role of **Red Hat Enterprise Linux Workstation**.

   8.3. Select an SLA level of **Self-Support**.

   8.4. Select a usage of **Development/Test**.

   8.5. Click **Done**.

9. Click **Begin Installation**.

10. While the installation progresses, set the password for `root` to `redhat`.

    10.1. Click **Root Password**.

    10.2. Enter `redhat` in the **Root Password** field.

    10.3. Enter `redhat` in the **Confirm** field.

    10.4. The password is weak so you will need to click **Done** twice.

11. While the installation progresses, add the `lab` administrative user with a password of `redhat`.

    11.1. Click **User Creation**.

    11.2. Enter `lab` in the **Full Name** field.

    11.3. Select **Make this user administrator**.

    11.4. Enter `redhat` in the **Password** field.

    11.5. Enter `redhat` in the **Confirm** field.

    11.6. The password is weak so you will need to click **Done** twice.

12. When the installation is complete, click **Reboot**.

13. Ensure the system boots from the local drive.

    13.1. If the system boots from the installation media again, press **Esc**, and then type `local` and press `Enter`. Alternatively, select **Troubleshooting** and **Boot from local drive**.

14. Accept the End User License Agreement (EULA), then click **Continue**.

14.1. Click **License Information**.

14.2. Select **I accept the license agreement**.

14.3. Click **Done**.

14.4. Click **Finish Configuration**.

15. When the system displays the login prompt, log in as **lab** with a password of **redhat**.

Finish

On **workstation**, run **lab install-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab install-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Software repositories for RHEL 8 have been restructured with most packages being hosted in the BaseOS and AppStream repositories.
- The latest version of Yum supports modules, which allow applications to have a different life cycle to the operating system.
- The latest Anaconda installer has a new feature named System Purpose, which automates the selection of system entitlements.
- The new system upgrade tool is the Leapp framework. It is currently under heavy development and has several limitations.

Chapter 3

Provisioning and Configuring Servers

Goal

Identify new package and system management tools and utilities.

Objectives

- Perform package management tasks using the new DNF-based version of **`yum`**.
- Perform local and remote server administration using the RHEL web console utility.
- Build multiple types of system images using the new Image Builder utility.
- Write Ansible Playbooks to automate common operations, using RHEL System Roles included with Red Hat Enterprise Linux.

Sections

- Performing Package Management using Yum (and Guided Exercise)
- Administering Servers with RHEL Web Console (and Guided Exercise)
- Building System Images with Image Builder (and Guided Exercise)
- Automating with RHEL System Roles (and Guided Exercise)

Lab

Provisioning and Configuring Servers

Performing Package Management using Yum

Objectives

After completing this section, students should be able to perform package management tasks using the new DNF-based version of **`yum`**.

Using Yum v4

Red Hat Enterprise Linux 8 includes version 4 of the **`yum`** utility, which uses the DNF technology as its back end. Although the back end for **`yum`** in version 4 has changed, typical options for the command remain the same.

DNF Technology in Yum v4

- DNF replaces YUM as the package management technology for RPM packages in Red Hat Enterprise Linux 8.
- The DNF API enhances integration, and solves stability issues of the Yum v3 API.
- Yum v4 includes the new functionality provided by DNF.
- The recommended **`yum`** command is a link to **`dnf`**.

Enhancements in Yum v4

- Supports modules that enable software AppStreams.
- Now understands weak and boolean dependencies.
- Provides a broader collection of plug-ins and add-on tooling.
- Improves the performance of Yum, which is key for cloud, CI/CD, and container workloads.

Modules, Streams, and Profiles

A module is a collection of packages that are installed together. Modules solve previous issues with dependencies with older or newer packages versions. A module can include one or more streams, which are different versions of the software provided by the module. A module has just one stream active at a time, by default the stream containing the latest version of the software. Modules also include profiles, which are a list of packages which support a use case, for example to deploy a minimal install.

References

For more information, refer to the *Installing software with yum* chapter in the *Red Hat Enterprise Linux 8 Configuring Basic System Settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#installing-software-with-yum_configuring-basic-system-settings

Guided Exercise

Performing Package Management with Yum

In this exercise, you will manage software packages and module streams using Yum.

Outcomes

You should be able to manage installed software versions using Yum.

Before You Begin

Log in as the **student** user with **student** as the password on **workstation**. Use SSH to log in to the **servera** system as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

1. Yum v4 retains previous functionality, and adds new features.

   1.1. Use **yum install postgresql** to install the default PostgreSQL 10 package and dependencies.

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

   1.2. Use **yum remove postgresql** to remove the default PostgreSQL 10 package and dependencies that were added.

```
[root@servera ~]# yum remove postgresql
```

2. Yum v4 now provides *module* and *stream* functions to install and modify groups of packages.

   2.1. Use **yum module list** to list available modules and streams.

```
[root@servera ~]# yum module list
```

   2.2. Use **yum module reset postgresql** to reset the enabled PostgreSQL module stream.

```
[root@servera ~]# yum module reset postgresql
```

   2.3. Use **yum module install postgresql:9.6** to install the PostgreSQL module and specific stream.

```
[root@servera ~]# yum module install postgresql:9.6
```

2.4. Use **`yum module remove postgresql:9.6`** to remove the Postgresql 9.6 module stream.

```
[root@servera ~]# yum module remove postgresql:9.6
```

This concludes the guided exercise.

Administering Servers with the Web Console

Objectives

After completing this section, students should be able to perform local and remote server administration using the RHEL web console utility.

Introducing the Web Console

There have been some major changes in the way Red Hat Enterprise Linux servers can be configured and accessed remotely. All the GUI-dependent **`system-config-*`** tools have been removed. No manual configuration, file editing, or performing multiple commands is required. All these tools or commands have been replaced by the web console (based on the Cockpit project technology). The web console is an interactive server administration interface. The web console interacts directly with the operating system from a real Linux session in a browser and replaces the need for X-forwarding sessions.

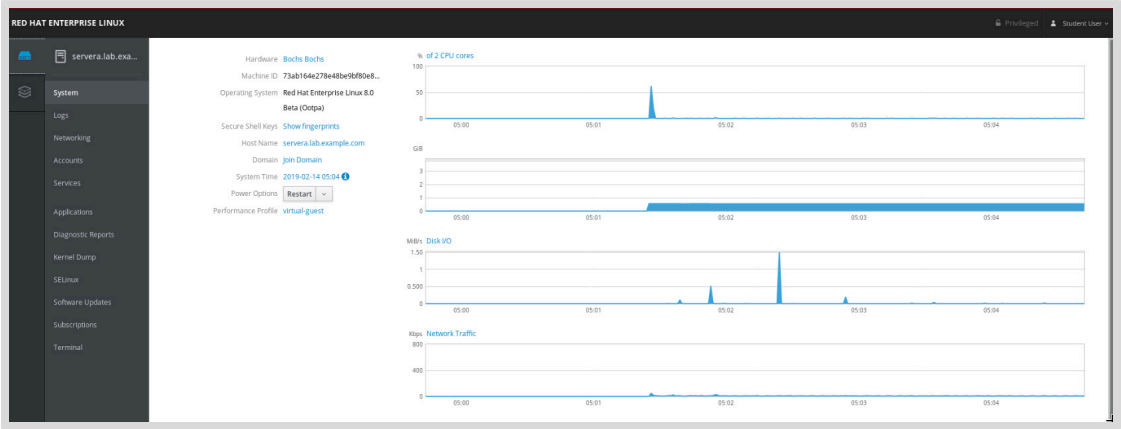


Figure 3.1: Web console interface

The Web Console as a Troubleshooting Tool

- Graphs of system statistics in real-time
- Inspecting and filtering syslog events
- Configuring SELinux and reviewing SELinux access control errors
- Creating diagnostic reports for Red Hat Support
- Enabling and configuring kernel crash dumps
- Running commands from a terminal session

The Web Console as a Management Tool

- Controlling running system services
- Configuring network interfaces and **`firewalld`**
- Administering user accounts
- Monitoring and configuring storage devices
- Managing system subscriptions and software updates
- Managing containers and virtual machines

If a feature you need is not available (for example, managing iptables), the web console can be extended by creating a new plug-in.

Basic system administration using the web console

Basic system operations

- Shutting down or restarting the system
- Accessing hardware information
- Changing performance profiles
- Configuring host name
- Connecting to a **`realmd`** domain

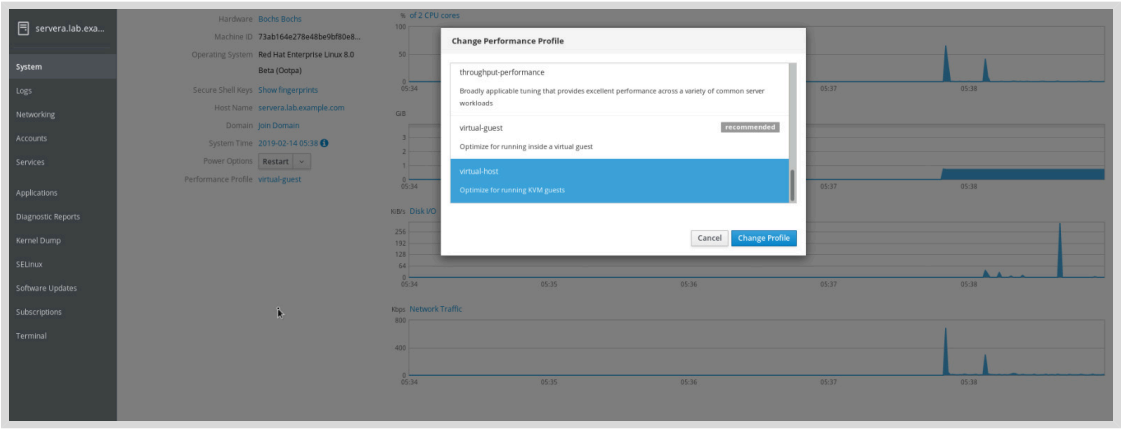


Figure 3.2: Changing system profile

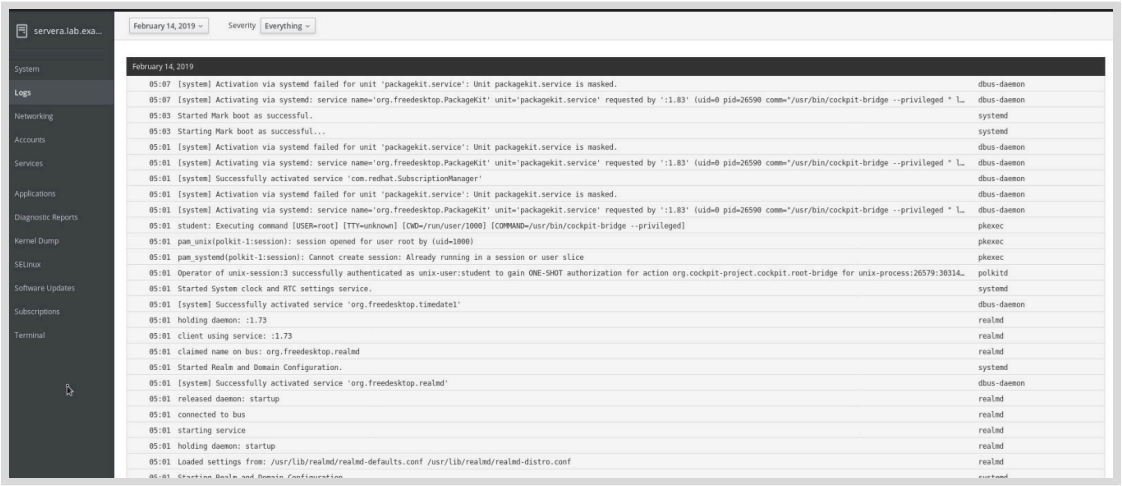


Figure 3.3: Accessing system logs

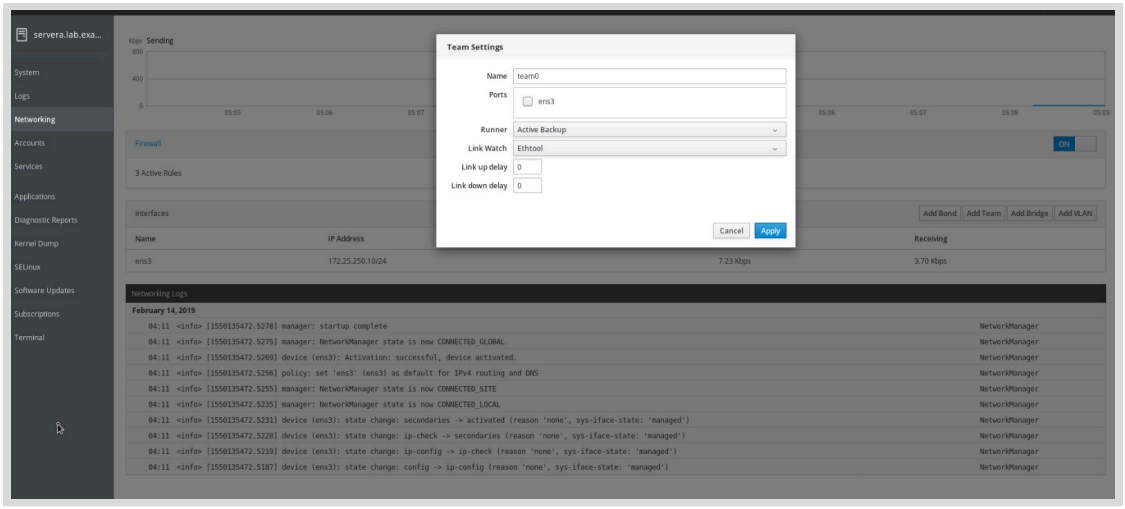


Figure 3.4: Changing network configuration

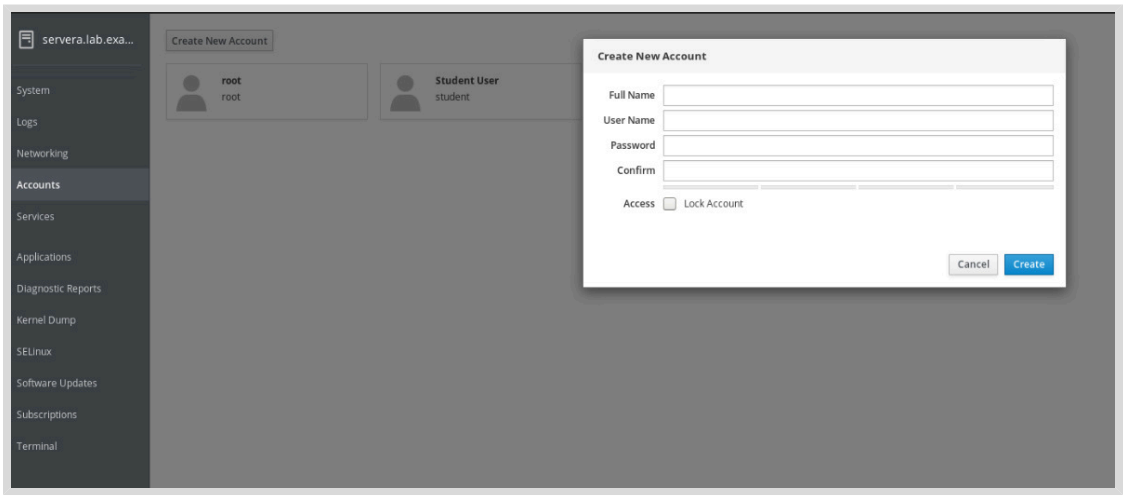


Figure 3.5: Creating new user accounts

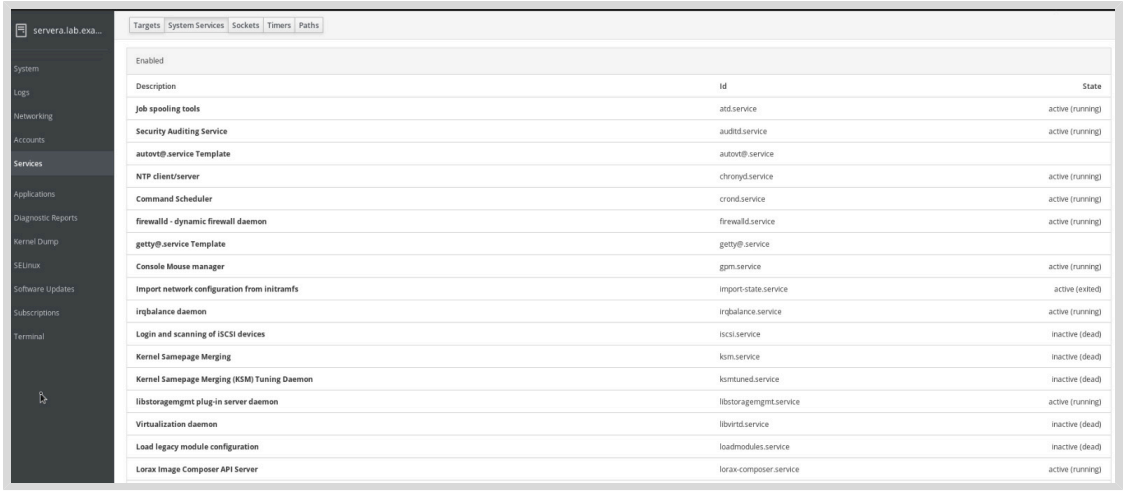


Figure 3.6: Controlling services

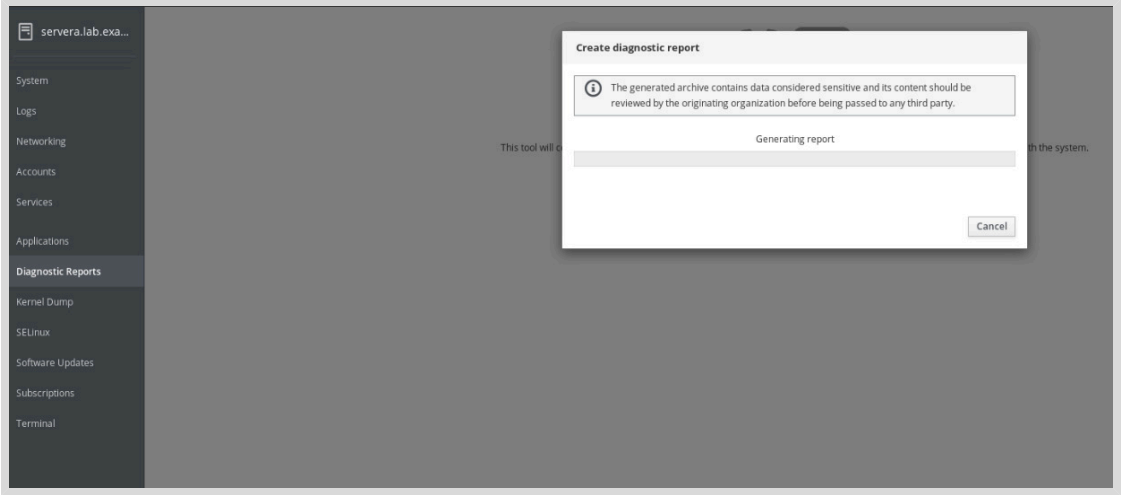


Figure 3.7: Creating diagnostic report

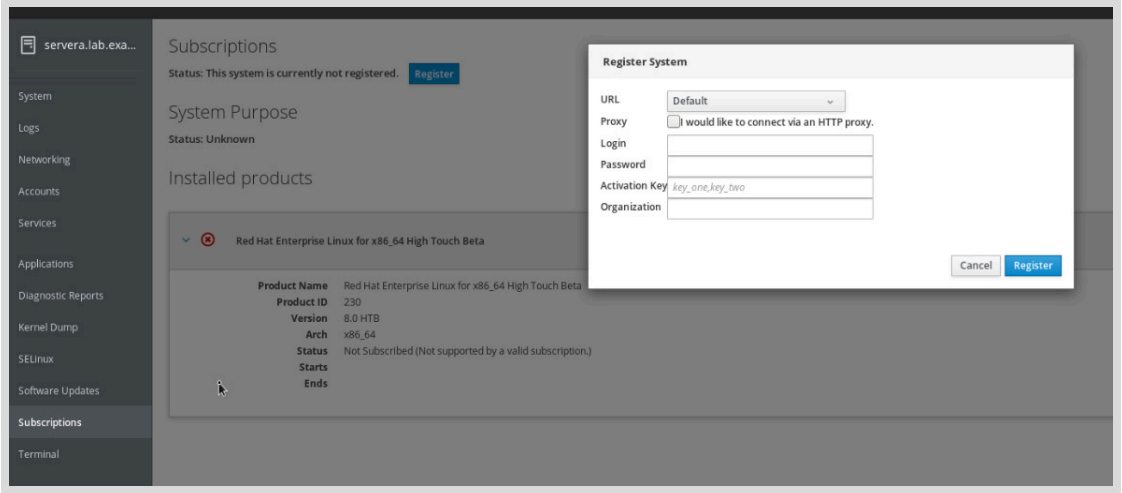


Figure 3.8: Registering a system

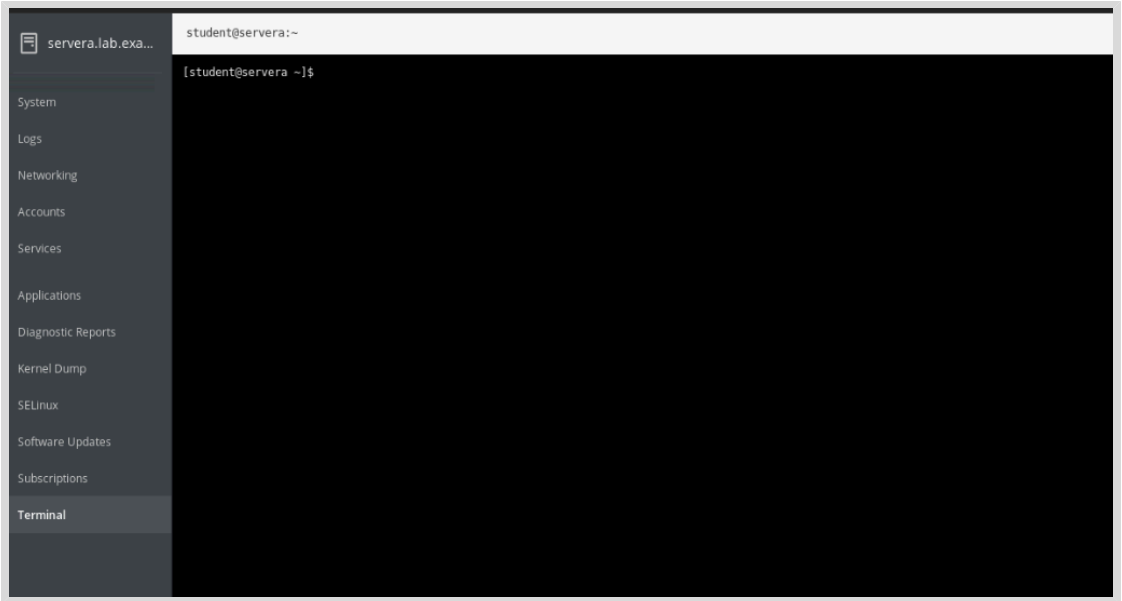


Figure 3.9: Accessing the terminal

References

For more information, refer to the *Managing systems using the web console* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_systems_using_the_web_console

Linux system administration via the management console: Cockpit
https://www.redhat.com/en/blog/linux-system-administration-management-console-cockpit

Creating Plugins for the Cockpit User Interface
https://cockpit-project.org/blog/creating-plugins-for-the-cockpit-user-interface.html

Guided Exercise

Administering Servers with the Web Console

In this exercise, you will manage a remote system using the web console utility.

Outcomes

You should be able to use the web console to monitor basic system features, inspect log files, create diagnostic report, create user account, and access the terminal.

Before You Begin

Log in as the **student** user with **student** as password on **workstation**. Use SSH to log in to the **servera** system as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

Note

In case **workstation** hangs using web console, restart the virtual machine, and log into the web console with a new browser.

1. The web console is already installed on the system, but it is not activated. Use the **systemctl enable --now cockpit.socket** command to enable the web console.

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

2. On **workstation**, open Firefox and log in to the web console interface running on **servera.lab.example.com** system as the **student** user with **student** as password.

   2.1. Open Firefox and go to the **https://servera.lab.example.com:9090** address.

   2.2. Accept the self-signed certificate by adding it as an exception.

   2.3. Log in as **student** user with **student** as password.

   You are now logged in as a normal user, with only minimal privileges.

   2.4. Click **Terminal** in the left navigation bar to access the terminal.

   A terminal session opens with the **student** user already logged in. Type in the **id** command to confirm that you are able to use the terminal remotely from a browser session.

```
[student@servera ~]$ id
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel)
 context=unconfined_u:unconfined_r:unconfined_t:s0
```

2.5. Click **Accounts** in the left navigation bar to manage users.

Notice that the **`student`** user is not permitted to create new accounts.

2.6. Click the **Student User** link.

On the **`student`** user's account details page, notice that the user is only permitted to set a new password, or to add authorized SSH public keys.

2.7. Click the **Student User** drop-down menu in the upper right corner. From the drop-down menu, choose **Log Out**

3. Log back in to the web console as the **`student`** user with **`student`** as password, but this time select the **Reuse my password for privileged tasks** checkbox.

Notice that the **`student`** user is now a **Privileged** user. Some of the previously unavailable administrative tasks are now available and can be executed using the **sudo** mechanism. For example, return to **Accounts** to notice that you are now allowed to create new accounts.

4. Click **`System`** in the left navigation bar to access the system's statistics.

This page shows you some of the basic operating system statistics, such as current load, disk usage, disk I/O, and network traffic.

5. Click **Logs** in the left navigation bar to access system's logs.

This page shows you the **`systemd`** system logs. You can choose the date that you want to access the logs from as well as the type of severity of the log entries you look for.

5.1. Click the **Severity** drop-down menu, and choose **Everything**.

5.2. Review the log entries for your current day of the month, and click on the one you want to view. A log entry detail page opens with additional information about the event. It shows you, for example, the host name, or the SELinux context, or the PID number of the process that the entry corresponds to.

6. Click **Networking** in the left navigation bar.

This page shows you the details of the current network configuration for **`servera`**, as well as real-time network statistics, firewall configuration, and log entries related to networking.

6.1. Click the **Firewall** link.

6.2. Click **Add Services...**.

6.3. In the **Add Services** window, click the checkbox for **Red Hat Satellite 6** service.

6.4. Click **Add Services** to add this service to the firewall configuration.

Notice that the **Red Hat Satellite 6** service was added to the list of allowed services.

6.5. In the same line as the **Red Hat Satellite 6** service, click the trash icon to remove the service from the firewall configuration.

Notice that the service was removed without asking for confirmation.

7. Click **Networking** in the left navigation bar.

Notice that in the **Interfaces** section you can add bonding, teaming, bridging, or a VLAN to that server.

7.1. Click the available network interface name (for example, **ens3**).

A details page opens to show real-time network statistics, and the network interface current configuration.

7.2. Click the **Address 172.25.250.10/24 via 172.25.250.254** link.

A new window opens to change the network interface configuration.

8. Add a second IP address to the interface identified in the previous step.

8.1. In the new **IPv4 Settings** window, click **+** next to **Manual**.

8.2. In the **Address** text box, enter **`172.25.250.99`** as the second IP address.

8.3. In the **Prefix length or Netmask** text box, enter **`24`** as the netmask value.

8.4. Click **Apply** to save the new network configuration.

The new configuration is immediately applied. The new IP address is visible in the **IPv4** line.

9. Click **Accounts** in the left navigation bar.

9.1. Click **Create New Account**.

9.2. In the new **Create New Account** window, fill in the details as follows:

Field	Value
Full Name	Sam Simons
User Name	ssimons
Password	redh@t123
Confirm	redh@t123

9.3. Click **Create**.

10. Click **Services** in the left navigation bar.

10.1. Scroll down the list of **System Services**. Find and click the **Software RAID monitoring and management** link.

To find the required service within Firefox, search for text using **`Ctrl`**+**F**.

The service's details page shows that it is inactive.

10.2. Click **Start** to start the service.

The reason this service fails to start is the missing **`/etc/mdadm.conf`** configuration file.

10.3. In the left pane, click **Services**.

10.4. Scroll down the list of **System Services**. In the **Disabled** section, find and click the **Kernel process accounting** link.

10.5. Click **Enable**.

10.6. Click **Start**.

The service is now enabled and active. This service performs a one-time change. It starts and then exits.

11. Click **Diagnostic Reports** in the left navigation bar.

11.1. Click **Create Report** and wait for the report to be created (it takes up to two minutes to finish).

11.2. When the report is ready, click **Download report**, followed by **Save File**.

11.3. Click **Close**.

12. Click **SELinux** in the left navigation bar.

Notice the SELinux access control errors. Depending on the current environment, there might not be any errors present. If not, proceed with the next step.

Click on any of the reported errors to access detailed information about the event.

In the details section for each event, follow the suggested solution to solve the problem, or click on the trash icon to delete the entry from the list, as well as find the original audit log entry for that event.

13. Add a remote system to the web console dashboard.

13.1. Click **Terminal** in the left navigation bar.

On **`servera.lab.example.com`** install the *cockpit-dashboard* package.

```
[student@servera ~]# sudo yum install cockpit-dashboard
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

13.2. Log out from the web console interface. Log in again as the **`student`** user with **`student`** as password with the privileged user checkbox marked.

In the left navigation bar there is now a new section representing the Dashboard.

13.3. Click **Dashboard** in the left navigation bar.

It shows real-time graphs for various statistics from **`servera.lab.example.com`**.

13.4. Click **+**.

13.5. In the **Add Machine to Dashboard** window, enter the **`serverb.lab.example.com`** host name.

13.6. Click **Add**.

13.7. In the **Unknown Host Key** window click **Connect**.

There are now two hosts in the **Dashboard**. Use the same web console interface running on **`servera.lab.example.com`** to make changes to **`serverb.lab.example.com`**. To switch to a different server, click the name of the desired server from the **Servers** list in the **Dashboard**.

14. Log off from the web console interface.

15. Log off from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

This concludes the guided exercise.

Building System Images with Image Builder

Objectives

After completing this section, students should be able to build multiple types of system images using the new image builder utility.

Image Builder

Image builder (project name Composer) is a new tool available in Red Hat Enterprise Linux 8 as technology preview. It allows administrators to create customized system images of Red Hat Enterprise Linux. Image builder has a graphical user interface in the web console. It can also be accessed from a terminal with the use of the **`composer-cli`** command.

Image Builder Output Formats

- QEMU QCOW2 Image (**`*.qcow2`**)
- Ext4 File System Image (**`*.img`**)
- Raw Partitioned Disk Image (**`*.img`**)
- Live Bootable ISO (**`*.iso`**)
- TAR Archive (**`*.tar`**)
- Amazon Machine Image Disk (**`*.ami`**)
- Azure Disk Image (**`*.vhd`**)
- VMware Virtual Machine Disk (**`*.vmdk`**)

Image Builder Blueprints

Image builder creates system images based on *blueprints*. A blueprint defines the image by listing packages, users, SSH keys and other possible resources that will be part of the system. Blueprints are versioned and can be edited. When a new image is created from a blueprint using the web console, the image is associated with the blueprint.

Image Builder System Requirements

- Dedicated virtual machine with SELinux in permissive mode
- CPU with 2 cores
- 4 GB RAM
- 20 GiB disk space
- Connectivity to Internet and to another server with repository mirrors.

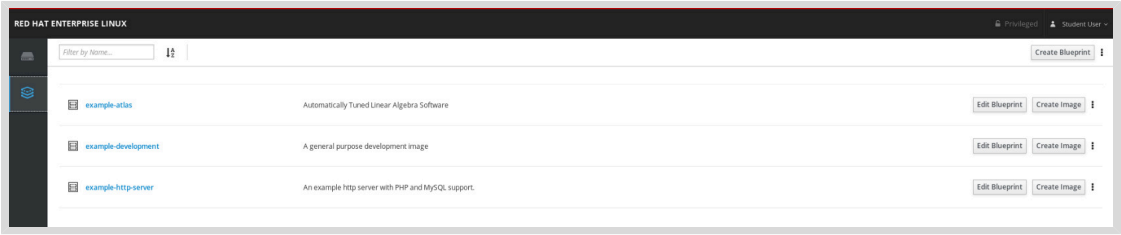


Figure 3.10: Image builder interface

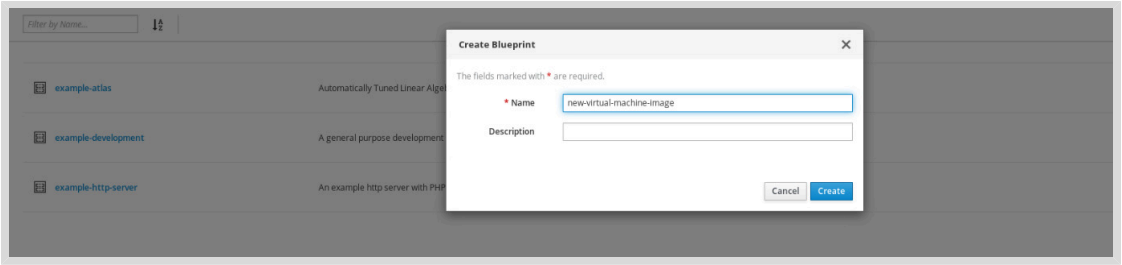


Figure 3.11: Creating a blueprint

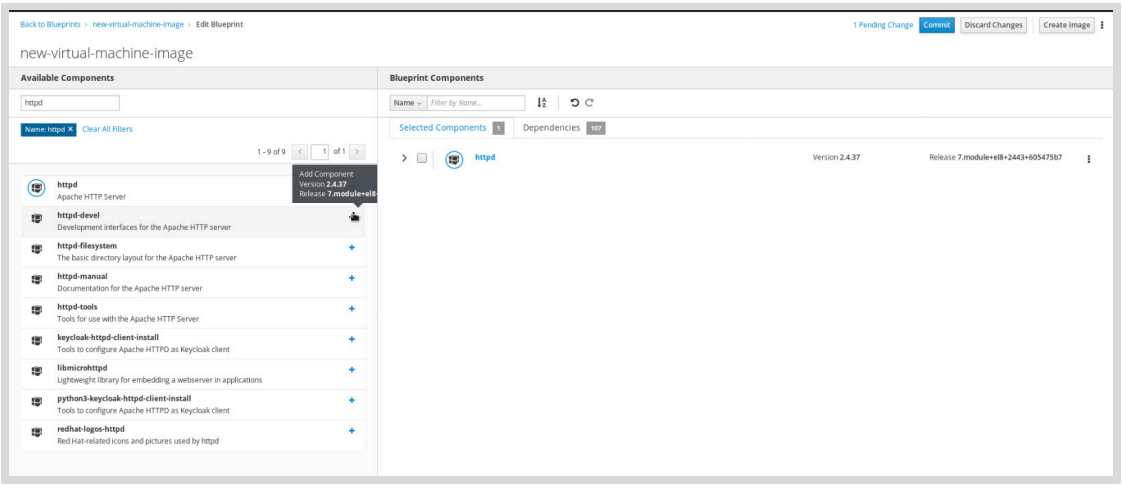


Figure 3.12: Customizing a blueprint

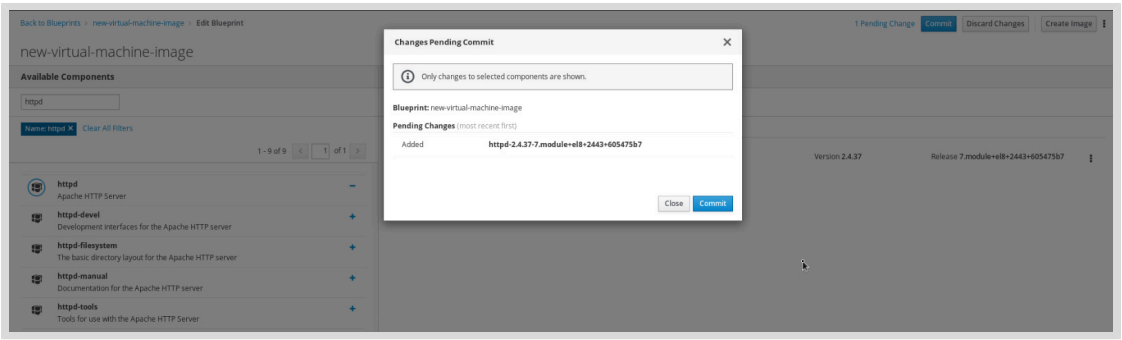


Figure 3.13: Committing changes

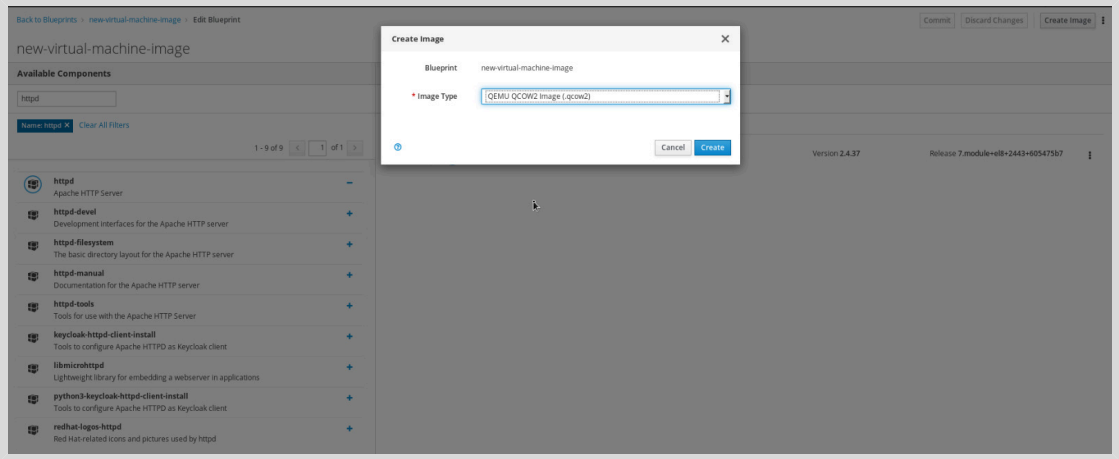


Figure 3.14: Creating new image

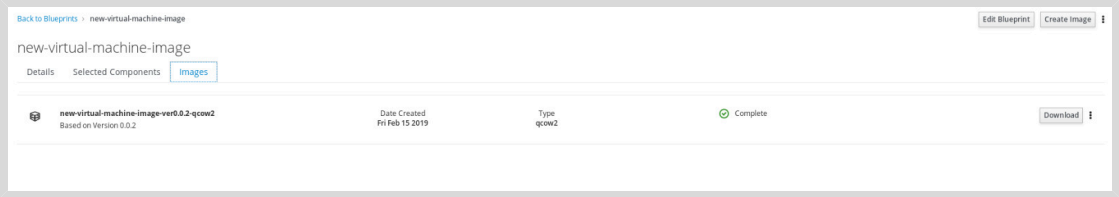


Figure 3.15: New image completed

References

For more information, refer to the *Composing a customized RHEL system image* chapter in the *Installing and deploying RHEL* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/composing_a_customized_rhel_system_image

Lorax (upstream source project for the image builder)
https://weldr.io/lorax/

Guided Exercise

Building System Images with the Image Builder

In this exercise, you will build a system image using the image builder utility.

Outcomes

You should be able to use the image builder to create system images.

1. Log in as the **student** user with **student** as password on **workstation**. From a terminal, use SSH to log in to the **servera** system as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

Note

In case **workstation** hangs using the web console, restart the virtual machine, and log into the web console with a new browser.

2. On **servera** install the image builder and the web console image builder plug-in.

   2.1. Use the **yum install lorax-composer composer-cli cockpit-composer** command.

```
[root@servera ~]# yum install lorax-composer \
> composer-cli cockpit-composer
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

   2.2. Use the **systemctl enable --now** command to enable and start the image builder and web console services.

```
[root@servera ~]# systemctl enable --now lorax-composer.socket cockpit.socket
```

3. On **workstation** use Firefox to open the web console running on **servera.lab.example.com:9090**. Log in as the **student** user with the password **student** and the privileged user checkbox selected.

4. Click **Image Builder** in the left navigation bar. The **Image Builder** will be available if you logged in after the packages were installed, because the image builder adds newly installed interface objects during log in initialization.

5. Create a new bootable ISO image, using one of the existing blueprints.

   5.1. Click the **example-development** blueprint.

5.2. Click the **Images** tab.

From here, you can download completed images, or see the status of previously build images.

5.3. Click the **Create Image** button in the center of the page.

5.4. Click the **Image Type** menu, and from the list choose **Live Bootable ISO (.iso)** type.

5.5. Click **Create**.

The new image appears on the list of available images. The creation process takes around 10 minutes to finish. Requests to create additional images are added to the queue, so it is safe to move on to the next steps.

5.6. When the image creation process is completed, the image can be downloaded to the local system, but this exercise does not need to take time to download the ISO file.

6. Create a new blueprint customized with the *mc* package.

6.1. Use the links at the top of the page to return to the **Blueprints** page.

6.2. Click **Create Blueprint**.

6.3. In the **Create Blueprint** window, enter **`rh-vm`** as the name for the blueprint.

6.4. Click **Create**.

6.5. In the **Filter by Name...** search box, enter the *mc* package name, and press **`Enter`**.

6.6. Scroll down the list of filtered packages, and click the **+** icon next to the *mc* package.

The *mc* package is added to the blueprint.

6.7. Click **Commit** to commit the changes.

6.8. In the **Changes Pending Commit** window, click **Commit**.

7. Use the new **`rh-vm`** blueprint to create a new QCOW2 image.

7.1. Use the top breadcrumb link to navigate back to the **Blueprints** page.

7.2. In the line with the **rh-vm** blueprint, click **Create Image**.

7.3. Click the **Image Type** drop-down menu and select **QEMU QCOW2 Image (.qcow2)**.

7.4. Click **Create**.

The creation process takes around 10 minutes to finish. Requests to create additional images are added to the queue, so it is safe to move on to the next steps. Verify the creation process status by clicking the **`rh-vm`** blueprint, then the **Images** tab.

7.5. When the image creation process is complete, the image can be downloaded to the local system. In this example, you do not need to download the QCOW2 file.

8. Customize the **`rh-vm`** blueprint by adding the **`student`** user and the student's SSH key to that blueprint. To simplify this task, and to avoid errors unrelated to learning the image builder, this course provides a file that contains preconfigured user information and a user SSH key.

8.1. On **servera.lab.example.com**, use the **composer-cli blueprints save *BLUEPRINTNAME*** command, to export the blueprint configuration to a file. The created filename will automatically include **.toml** as its extension.

```
[root@servera ~]# composer-cli blueprints save rh-vm
```

8.2. Use **wget** to download the **custom.user** blueprint customizations from the **classroom** server.

```
[root@servera ~]# wget http://materials.example.com/labs/custom.user
```

8.3. Append the customization from the **custom.user** file to the **rh-vm.toml** blueprint file.

```
[root@servera ~]# cat custom.user >> rh-vm.toml
```

8.4. Review the modified blueprint:

```
[root@servera ~]# cat rh-vm.toml
name = "rh-vm"
description = ""
version = "0.0.2"
modules = []
groups = []

[[packages]]
name = "mc"
version = "4.8.19"

[customizations]

[[customizations.user]]
name = "student"
password = "$6$C3w0glL853ygS8k7$QxoI50 ...output omitted... d5LDdi/dr0Zq1"
key = "ssh-rsa AAAAB3NzaC1yc2EAA ...output omitted... 6QuZf8f7aL LabGradingKey"
groups = ["users", "wheel"]
```

8.5. Push the modified blueprint file back to the image builder server.

```
[root@servera ~]# composer-cli blueprints push rh-vm.toml
```

8.6. Verify that the changes were pushed to the image builder server.

```
[root@servera ~]# composer-cli blueprints show rh-vm
name = "rh-vm"
description = ""
version = "0.0.2"
modules = []
groups = []

[[packages]]
```

```
name = "mc"
version = "4.8.19"

[customizations]

[[customizations.user]]
name = "student"
password = "$6$C3w0glL853 ...output omitted... 7xOdxd5LDdi/dr0Zq1"
key = "ssh-rsa AAAAB3Nza ...output omitted... QuZf8f7aL LabGradingKey"
groups = ["users", "wheel"]
```

8.7. Create a new QCOW2 image that contains the modifications. Use the **composer-cli compose start *BLUEPRINTNAME TYPE*** command.

```
[root@servera ~]# composer-cli compose start rh-vm qcow2
Compose 36b17 ...output omitted... 8299b2 added to the queue
```

9. Use the **composer-cli compose list** command to verify the current status of the build.

```
[root@servera ~]# composer-cli compose list
36b17126-3c1a-4abb-9235-b3748c8299b2 RUNNING rh-vm 0.0.2 qcow2
0ba76f6e-4acf-4909-a750-5676baac586d FINISHED example-development 0.0.1 live-iso
3a899344-3e5d-40b8-bf96-1d91d65f2054 FINISHED rh-vm 0.0.2 qcow2
```

The new image has a new name with a version corresponding to the latest blueprint version. The image name is associated to the blueprint used to create it.

10. After the final build is finished, log off from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

This concludes the guided exercise.

Automating with RHEL System Roles

Objectives

After completing this section, students should be able to perform tasks using Ansible Playbooks to automate common operations, using RHEL System Roles included with Red Hat Enterprise Linux.

System Roles in Red Hat Enterprise Linux 8

Red Hat Enterprise Linux 8 includes Ansible system roles to simplify automation of complex tasks, and configuration of Red Hat Enterprise Linux subsystems. You can run a role with an Ansible Playbook, which is a collection of Ansible plays. Each play can use a role defining it in a **`roles`** section. Plays can also include other sections as a **`post_tasks`** section to define tasks to be executed after the play's normal tasks, or a **`handlers`** section to define handlers. You can run playbooks with the **`ansible-playbook`** command included in Red Hat Ansible Engine. Red Hat Enterprise Linux 8 does not install Red Hat Ansible Engine by default.

Ansible System Roles in RHEL 8

- Includes **`rhel-system-roles.kdump`**, **`rhel-system-roles.network`**, **`rhel-system-roles.postfix`**, **`rhel-system-roles.selinux`**, and **`rhel-system-roles.timesync`**
- Available in the *rhel-system-roles* package
- Additional system roles planned: storage, logging, metrics, hardware management, and host inspection.

Python on Red Hat Enterprise Linux 8 Managed Hosts

Red Hat Enterprise Linux 8 provides Python 3.6 as the default Python implementation. There is no version of Python installed by default. If you wish to install Python on the RHEL 8 managed host, you should install the **`python36`** Yum module.

The installed Ansible version will check if the **`/usr/bin/python`** executable exists, and look for the common system Python interpreter if it is not. Future versions of Ansible 2.x will use the lookup table first, even if the **`/usr/bin/python`** executable is present.

References

Knowledgebase: Red Hat Enterprise Linux (RHEL) System Roles
https://access.redhat.com/articles/3050101

Linux System Roles
https://linux-system-roles.github.io/

Ansible Porting Guides
https://docs.ansible.com/ansible/latest/porting_guides/porting_guides.html

Guided Exercise

Automating with RHEL System Roles

In this exercise, you will use one of the Red Hat Enterprise Linux system roles in conjunction with a normal task to configure time synchronization and the time zone on your servers.

Outcomes

You should be able to:

- Install the Red Hat Enterprise Linux system roles.
- Find and use the RHEL system roles documentation.
- Use the **`rhel-system-roles.timesync`** role in a playbook to configure time synchronization on remote hosts.

Scenario Overview

Your organization maintains two data centers: one in the United States (Chicago) and one in Finland (Helsinki). To aid log analysis of database servers across data centers, ensure that the system clock on each host is synchronized using Network Time Protocol. To aid time-of-day activity analysis across data centers, ensure that each database server has a time zone corresponding to the host's data center location.

Time synchronization has the following requirements:

- Use the NTP server located at **`classroom.example.com`**. Enable the **`iburst`** option to accelerate initial time synchronization.
- Use the *chrony* package for time synchronization.

Before You Begin

From **`workstation`**, run the command **`lab tools-automating start`** to prepare the environment for this exercise. This creates the working directory, **`tools-automating`**, and populates it with an Ansible configuration file and host inventory.

```
[student@workstation ~]$ lab tools-automating start
```

1. Log in to your **`workstation`** host as **`student`**. Verify that *ansible* is available, and change to the **`tools-automating`** working directory.

1.1. Verify that *ansible* is available.

```
[student@workstation ~]$ yum list installed ansible
...output omitted...
```

1.2. Change to the **`tools-automating`** working directory.

```
[student@workstation ~]$ cd ~/tools-automating
[student@workstation tools-automating]$
```

2. Install the RHEL system roles on the control node, **`workstation.lab.example.com`**. Verify the location of the installed roles on the control node.

2.1. Use **`ansible-galaxy`** to verify that no roles are initially available. The empty output indicates that no roles were found.

```
[student@workstation tools-automating]$ ansible-galaxy list
...output omitted...
[student@workstation tools-automating]$
```

2.2. Install the *rhel-system-roles* package.

```
[student@workstation tools-automating]$ sudo yum install rhel-system-roles
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2.3. Use **`ansible-galaxy`** to verify that system roles are available.

```
[student@workstation tools-automating]$ ansible-galaxy list
- linux-system-roles.kdump, (unknown version)
- linux-system-roles.network, (unknown version)
- linux-system-roles.postfix, (unknown version)
- linux-system-roles.selinux, (unknown version)
- linux-system-roles.timesync, (unknown version)
- rhel-system-roles.kdump, (unknown version)
- rhel-system-roles.network, (unknown version)
- rhel-system-roles.postfix, (unknown version)
- rhel-system-roles.selinux, (unknown version)
- rhel-system-roles.timesync, (unknown version)
```

Roles are located in the **`/usr/share/ansible/roles`** directory. A role beginning with **`linux-system-roles`** is a symlink to the matching **`rhel-system-roles`** role.

3. Create a playbook, **`configure_time.yml`**, with one play that targets the **`database_servers`** host group. Include the **`rhel-system-roles.timesync`** role in the **`roles`** section of the play.

The contents of the **`configure_time.yml`** now matches:

```
---
- name: Time Synchronization
  hosts: database_servers

  roles:
    - rhel-system-roles.timesync
```

4. The role documentation contains a description of each role variable, including its default value. Determine the role variables to override to meet the time synchronization requirements.

Place role variable values in a file named **`timesync.yml`**. Because these variable values apply to all inventory hosts, place the file in the **`group_vars/all`** subdirectory.

4.1. Review the *Role Variables* section of the **`README.md`** file for the **`rhel-system-roles.timesync`** role.

```
[student@workstation tools-automating]$ cat \
> /usr/share/doc/rhel-system-roles/timesync/README.md
...output omitted...
Role Variables
--------------

...output omitted...
# List of NTP servers
timesync_ntp_servers:
  - hostname: foo.example.com   # Hostname or address of the server
    minpoll: 4                  # Minimum polling interval (default 6)
    maxpoll: 8                  # Maximum polling interval (default 10)
    iburst: yes                 # Flag enabling fast initial synchronization
                                # (default no)
    pool: no                    # Flag indicating that each resolved address
                                # of the hostname is a separate NTP server
                                # (default no)
...output omitted...
# Name of the package which should be installed and configured for NTP.
# Possible values are "chrony" and "ntp". If not defined, the currently active
# or enabled service will be configured. If no service is active or enabled, a
# package specific to the system and its version will be selected.
timesync_ntp_provider: chrony
```

4.2. Create the **`group_vars/all`** subdirectory.

```
[student@workstation tools-automating]$ mkdir -pv group_vars/all
mkdir: created directory 'group_vars'
mkdir: created directory 'group_vars/all'
```

4.3. Create a new file **`group_vars/all/timesync.yml`** using a text editor. Add variable definitions to satisfy the time synchronization requirements. The file now contains:

```
---
#rhel-system-roles.timesync variables for all hosts

timesync_ntp_provider: chrony

timesync_ntp_servers:
  - hostname: classroom.example.com
    iburst: yes
```

5. Insert a task to set the time zone for each host. Ensure that the task uses the **`timezone`** module and executes after the **`rhel-system-roles.timesync`** role.

Because hosts may not belong to the same time zone, use a variable (**`host_timezone`**) for the time zone name.

5.1. Review the *Examples* section of the **`timezone`** module documentation.

```
[student@workstation tools-automating]$ ansible-doc timezone | grep -A 4
 "EXAMPLES"
EXAMPLES:
- name: set timezone to Asia/Tokyo
  timezone:
    name: Asia/Tokyo
```

5.2. Add a task to the **post_tasks** section of the play in the **configure_time.yml** playbook. Model the task after the documentation example, but use the **host_timezone** variable for the time zone name.

The documentation in **ansible-doc timezone** recommends a restart of the Cron service if the module changes the timezone, to make sure Cron jobs run at the right times. Add a **notify** keyword to the task, with an associated value of **restart crond**. The **post_tasks** section of the play should read:

```
  post_tasks:
    - name: Set timezone
      timezone:
        name: "{{ host_timezone }}"
      notify: restart crond
```

5.3. Add the **restart crond** handler to the **Time Synchronization** play. The complete playbook now contains:

```
---
- name: Time Synchronization
  hosts: database_servers

  roles:
    - rhel-system-roles.timesync

  post_tasks:
    - name: Set timezone
      timezone:
        name: "{{ host_timezone }}"
      notify: restart crond

  handlers:
    - name: restart crond
      service:
        name: crond
        state: restarted
```

6. For each data center, create a file named **timezone.yml** that contains an appropriate value for the **host_timezone** variable. Use the **timedatectl list-timezones** command to find the valid time zone string for each data center.

6.1. Create the **group_vars** subdirectories for the **na_datacenter** and **europe_datacenter** host groups.

```
[student@workstation tools-automating]$ mkdir -pv \
> group_vars/{na,europe}_datacenter
mkdir: created directory 'group_vars/na_datacenter'
mkdir: created directory 'group_vars/europe_datacenter'
```

6.2. Use the **`timedatectl list-timezones`** command to determine the time zone for both the US and European data centers:

```
[student@workstation tools-automating]$ timedatectl list-timezones | grep Chicago
America/Chicago
[student@workstation tools-automating]$ timedatectl list-timezones | grep Helsinki
Europe/Helsinki
```

6.3. Create the **`timezone.yml`** for both data centers:

```
[student@workstation tools-automating]$ echo "host_timezone: America/Chicago" > \
> group_vars/na_datacenter/timezone.yml
[student@workstation tools-automating]$ echo "host_timezone: Europe/Helsinki" > \
> group_vars/europe_datacenter/timezone.yml
```

6.4. The final file structure of the **`~/tools-automating`** folder should look similar to the following **`tree`** command output:

```
[student@workstation tools-automating]$ tree ~/tools-automating
/home/student/tools-automating/
├── ansible.cfg
├── configure_time.yml
├── group_vars
│   ├── all
│   │   └── timesync.yml
│   ├── europe_datacenter
│   │   └── timezone.yml
│   └── na_datacenter
│       └── timezone.yml
├── inventory
└── roles
    └── requirements.yml

5 directories, 7 files
```

7. Run the playbook. Some tasks will result in errors, because they attempt to stop services that are not running in this exercise. The **`timesync`** system role would typically expect to find and stop the deprecated NTP services.

```
[student@workstation tools-automating]$ ansible-playbook configure_time.yml

PLAY [Time Synchronization] ***************************************************

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

```
TASK [rhel-system-roles.timesync : Check if only NTP is needed] **************
ok: [servera.lab.example.com]
ok: [serverb.lab.example.com]

...output omitted...

TASK [rhel-system-roles.timesync : Enable timemaster] ************************
skipping: [servera.lab.example.com]
skipping: [serverb.lab.example.com]

RUNNING HANDLER [rhel-system-roles.timesync : restart chronyd] ***************
changed: [servera.lab.example.com]
changed: [serverb.lab.example.com]

TASK [Set timezone] **********************************************************
changed: [serverb.lab.example.com]
changed: [servera.lab.example.com]

RUNNING HANDLER [restart crond] **********************************************
changed: [serverb.lab.example.com]
changed: [servera.lab.example.com]

PLAY RECAP *******************************************************************
servera.lab.example.com    : ok=17   changed=6    unreachable=0    failed=0
serverb.lab.example.com    : ok=17   changed=6    unreachable=0    failed=0
```

8. Verify the time zone settings of each server. Use the Ansible command module to see the output of the **`timedatectl`** command on all the database servers.

Note

The actual time-zones listed will vary depending on the time of year, and whether daylight savings is active.

```
[student@workstation tools-automating]$ ansible database_servers -m command -a
 timedatectl
serverb.lab.example.com | CHANGED | rc=0 >>
               Local time: Thu 2019-06-13 05:10:57 EEST
           Universal time: Thu 2019-06-13 02:10:57 UTC
                 RTC time: Thu 2019-06-13 02:10:56
                Time zone: Europe/Helsinki (EEST, +0300)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no

servera.lab.example.com | CHANGED | rc=0 >>
               Local time: Wed 2019-06-12 21:10:57 CDT
           Universal time: Thu 2019-06-13 02:10:57 UTC
                 RTC time: Thu 2019-06-13 02:10:56
                Time zone: America/Chicago (CDT, -0500)
```

```
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Each server has a time zone setting based on its geographic location.

Finish

Run the **`lab tools-automating finish`** command to clean up the managed host.

```
[student@workstation ~]$ lab tools-automating finish
```

This concludes the guided exercise.

Lab

Provisioning and Configuring Servers

Performance Checklist

In this lab you will enable a module stream and install packages from it, accessing the server through the web console interface, then run an Ansible Playbook that uses a RHEL System Role to ensure further configuration is correct on the server.

Outcomes

You should be able to:

- Enable and use the web console to provide system access.
- Enable module streams and install packages.
- Use the **`rhel-system-roles.postfix`** role to install and configure Postfix.

Before You Begin

Log in to **`workstation`** as **`student`** using **`student`** as the password. On **`workstation`**, run the **`lab tools-review start`** command to verify that the environment is ready and prepare the systems for the exercise.

```
[student@workstation ~]$ lab tools-review start
```

Note

In case **`workstation`** hangs using the web console, restart the virtual machine, and log into the web console with a new browser.

1. Use the web console to access **`servera`** and Yum to enable the Python 3.6 module stream.
2. Using Red Hat Enterprise Linux Roles on **`workstation`**, create a playbook named **`lab-postfix.yml`** within the working directory of **`~/lab-roles`**. Include the directive **`become: yes`** to enable **`sudo`** usage and configure **`servera`** to be a Postfix MTA, using the following role configuration variables:

Variable	Value
inet_interfaces	**all**
myorigin	**$mydomain**
mydestination	**$myhostname localhost.$mydomain $mydomain**
relay_domains	**$mydestination**
relay_host	**example.com**

Finish

From **workstation**, run **lab tools-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab tools-review finish
```

Note
Python36 is removed from **servera** during the lab finish to prepare for later labs.

This concludes the lab.

Solution

Provisioning and Configuring Servers

Performance Checklist

In this lab you will enable a module stream and install packages from it, accessing the server through the web console interface, then run an Ansible Playbook that uses a RHEL System Role to ensure further configuration is correct on the server.

Outcomes

You should be able to:

- Enable and use the web console to provide system access.
- Enable module streams and install packages.
- Use the **`rhel-system-roles.postfix`** role to install and configure Postfix.

Before You Begin

Log in to **`workstation`** as **`student`** using **`student`** as the password. On **`workstation`**, run the **`lab tools-review start`** command to verify that the environment is ready and prepare the systems for the exercise.

```
[student@workstation ~]$ lab tools-review start
```

> **Note**
> In case **`workstation`** hangs using the web console, restart the virtual machine, and log into the web console with a new browser.

1. Use the web console to access **`servera`** and Yum to enable the Python 3.6 module stream.

   1.1. On **`workstation`**, open Firefox and navigate to `https://servera.lab.example.com:9090`. If a certificate error appears, accept the self-signed certificate.

   Check **Reuse my password for privileged tasks** and sign in using **`student`** as user name and **`student`** as the password.

   Click on **`servera.lab.example.com`** in the left navigation bar and select the **Terminal** menu entry on the left.

   1.2. Use the **`yum module list`** command to list available streams, and choose the one for Python 3.6.

```
[student@servera ~]$ sudo yum module list
```

   1.3. Use the **`yum module install python36`** command to install the Python 3.6 module.

```
[student@servera ~]$ sudo yum module install python36
```

2. Using Red Hat Enterprise Linux Roles on **workstation**, create a playbook named **lab-postfix.yml** within the working directory of **~/lab-roles**. Include the directive **become: yes** to enable **sudo** usage and configure **servera** to be a Postfix MTA, using the following role configuration variables:

Variable	Value
inet_interfaces	**all**
myorigin	**$mydomain**
mydestination	**$myhostname localhost.$mydomain $mydomain**
relay_domains	**$mydestination**
relay_host	**example.com**

2.1. Change to the **/home/student/lab-roles** working directory.

```
[student@workstation ~]$ cd ~/lab-roles
[student@workstation lab-roles]$
```

2.2. Create a playbook, **lab-postfix.yml**, with one play that targets **servera.lab.example.com**. Include the **rhel-system-roles.postfix** role in the **roles** section of the play.

The contents of the **lab-postfix.yml** now matches:

```
---
- hosts: servera.lab.example.com
  become: yes
  vars:
    postfix_conf:
      inet_interfaces: "all"
      myorigin: "$mydomain"
      mydestination: "$myhostname localhost.$mydomain $mydomain"
      relay_domains: "$mydestination"
      relay_host: "example.com"
  roles:
    - rhel-system-roles.postfix
```

2.3. Run the playbook.

```
[student@workstation lab-roles]$ ansible-playbook lab-postfix.yml
PLAY [all] *********************************************************************

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

```
TASK [rhel-system-roles.postfix : Install Postfix] ****************************
changed: [servera.lab.example.com]

...output omitted...

TASK [rhel-system-roles.postfix : Configure Postfix] **************************
changed: [servera.lab.example.com] => (item={'key': 'inet_interface', 'value':
 'all'})
changed: [servera.lab.example.com] => (item={'key': 'myorigin', 'value':
 '$mydomain'})
changed: [servera.lab.example.com] => (item={'key': 'mydestination', 'value':
 '$myhostname localhost.$mydomain $mydomain'})
changed: [servera.lab.example.com] => (item={'key': 'relay_domains', 'value':
 '$mydestination'})
changed: [servera.lab.example.com] => (item={'key': 'relay_host', 'value':
 'example.com'})

RUNNING HANDLER [rhel-system-roles.postfix : check postfix] *******************
changed: [servera.lab.example.com]

RUNNING HANDLER [rhel-system-roles.postfix : restart postfix] *****************
changed: [servera.lab.example.com]

PLAY RECAP ********************************************************************
servera.lab.example.com    : ok=9    changed=8    unreachable=0    failed=0
```

2.4. Use **`telnet servera.lab.example.com 25`** on **`workstation`** to test mail server functionality.

```
[student@workstation ~]$ telnet servera.lab.example.com 25
Connected to servera.lab.example.com.
Escape character is '^]'.
220 servera.lab.example.com ESMTP Postfix
^]

telnet> quit
Connection closed.
```

Finish

From **`workstation`**, run **`lab tools-review finish`** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab tools-review finish
```

Note

Python36 is removed from **`servera`** during the lab finish to prepare for later labs.

This concludes the lab.

Summary

In this chapter, you learned:

- Red Hat Enterprise Linux 8 includes version 4 of the Yum utility, which uses the DNF technology as its back end.
- A Yum module is a collection of packages which are installed together. Modules solve issues with dependencies with older or newer package versions.
- The web console is an interactive server administration interface.
- The image builder is a new tool allowing administrators to create customized system images of Red Hat Enterprise Linux.
- The image builder creates system images based on blueprints.
- Red Hat Enterprise Linux 8 includes Ansible system roles to simplify automation of complex tasks, and configuration of Red Hat Enterprise Linux subsystems.

Chapter 4

Adapting to Core System Changes

Goal

Manage core system components that have had significant changes, including the graphical desktop infrastructure, user authentication configuration, the NTP service, and Python version selection.

Objectives

- Explain the difference between Wayland and the legacy X Window System, and configure a server or user session to use either Wayland or X.
- Manage user authentication settings in PAM, NSS, and dconf with Authselect, and explain the differences between Authselect and Authconfig.
- Maintain NTP time synchronization using Chrony, and configure the time zone with `timedatectl`.
- Identify, install, and control the specific versions of Python available to a server.

Sections

- Displaying the Desktop with Wayland and X (and Guided Exercise)
- Managing User Authentication with Authselect (and Guided Exercise)
- Configuring NTP with Chrony (and Guided Exercise)
- Managing Python Versions in RHEL 8 (and Guided Exercise)

Lab

Adapting to Core System Changes

Displaying the Desktop with Wayland and X

Objectives

After completing this section, students should be able to:

- Explain the difference between Wayland and the legacy X Window System
- Configure a server or user session to use either Wayland or X

Wayland Windowing System

Wayland is a client/server windowing system designed to replace X11. The performance improvements of Wayland over X11 are derived from the simplification of the architecture.

Wayland Components

- *Wayland Compositor* - combines the display server and compositor roles.
- *Wayland Client* - the graphical application communicating with the Wayland Compositor. Wayland clients are responsible for performing their own rendering.
- *Linux Kernel* - responsible for communicating between the Wayland Compositor, the event device driver handling input and output devices (evdev), and the kernel mode-setting API (KMS) for resolution, depth and refresh.

Switching to Wayland

Wayland is not an extension of the X11 window system, but a replacement for it.

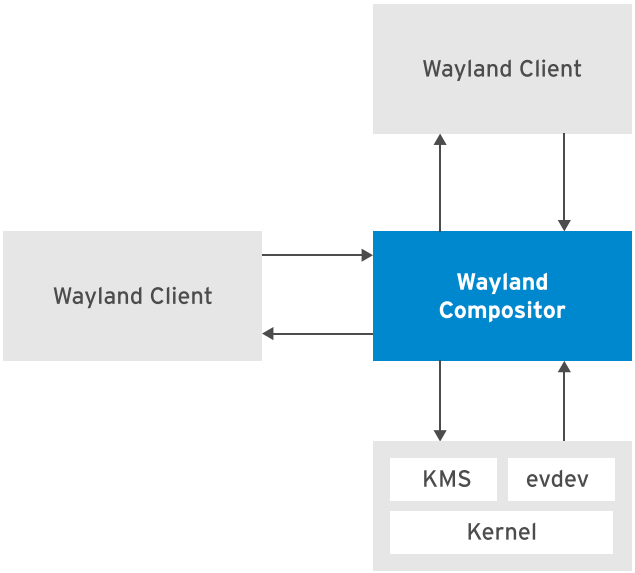


Figure 4.1: The Wayland architecture

Reasons to Switch to Wayland

- The X display server has become a bottleneck between clients, the compositor, and the kernel.

- X has legacy code such as the rendering API that is rarely used but must remain to conform to the X11 protocol.
- Most graphical desktops have a choice of compositor, which adds latency because the X display server must send the content to the compositor process for rendering, then return the rendered content to the X server to be displayed.

X11 Application Compatibility

X11 applications that have not been migrated to Wayland can still run in a Wayland compositor. XWayland is an Xorg server that runs as a Wayland client. When an X11 application is launched, the Wayland Compositor will call XWayland to handle the application's requests. XWayland is provided by the xorg-x11-server-Xwayland package.

Remote Rendering

Wayland is not network transparent, and does not support remote rendering. Instead, a remote rendering server, such as RDP or VNC desktop protocols, can be run on top of Wayland. It is also expected that upstream Wayland compositors will be created that include a remoting protocol.

Verify Wayland is Being Used

Use the following steps to determine whether your desktop is using Wayland.

```
[user@demo ~]$ loginctl
SESSION  UID USER  SEAT  TTY
      2 1000 user  seat0 tty2
     c1   42 gdm   seat0 tty1

2 sessions listed.

[user@demo ~]$ loginctl show-session 2 -p Type
Type=wayland
```

Gnome 3 uses Wayland by default. However, not all applications in Red Hat Enterprise Linux 8 have been migrated. Unmigrated applications are managed with an **XWayland** driver to provide their X server requirements.

If Wayland is incompatible with your graphics hardware or application, it is a simple process to revert to using Xorg. Uncomment the following line in **/etc/gdm/custom.conf** and then reboot your machine.

```
#WaylandEnable=false
```

This forces GDM to use Xorg and also sets the default session to Xorg.

References

Wayland Architecture
https://wayland.freedesktop.org/architecture.html

Wayland FAQ
https://wayland.freedesktop.org/faq.html

How to debug Wayland problems
https://fedoraproject.org/wiki/How_to_debug_Wayland_problems

Guided Exercise

Displaying the Desktop with Wayland and X

In this exercise, you will examine **workstation** to determine which applications use Wayland natively, and practice switching between Wayland and Xorg.

Outcomes

You should be able to determine which applications use Wayland, and which use X11 with the XWayland compatibility layer. You should also be able to switch between Wayland and Xorg, which is helpful when troubleshooting issues with graphical applications.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab core-wayland start** command.

```
[student@workstation ~]$ lab core-wayland start
```

1. Launch the Cheese photo application.

   1.1. Press **Alt+F2** to open the **Enter a Command** dialog box.

   1.2. Type **cheese** to launch the application.

2. Launch Looking Glass, the Gnome integrated debugger and inspector.

   2.1. Press **Alt+F2** to open the **Enter a Command** dialog box.

   2.2. Type **lg** to launch the inspector.

3. Use Looking Glass to determine which applications are using Wayland natively.

   3.1. Click **Windows** at the upper right.

   3.2. Click the text **gnome-shell** to display the object inspection dialog box.

   The first line in the object inspection dialog box lists the object type, which includes the text **MetaWindowXwayland**. This indicates that Gnome Shell is using the XWayland display server and so is still configured to use X11.

   Close the dialog box.

   3.3. Click the text **student@workstation:~**, which represents the open Gnome terminal.

   The first line in the object inspection dialog box includes the text **MetaWindowWayland**, confirming that Gnome terminal is a native Wayland application.

   Close the dialog box.

3.4. Click the text **`Take a photo`**, which represents the Cheese application.

The first line in the object inspection dialog box includes the text **`MetaWindowWayland`**, confirming that Cheese is a native Wayland application.

Close the dialog box.

Press **`Esc`** to exit from Looking Glass.

4. From the terminal, verify the session type using the **`loginctl`** command.

4.1. Run the **`loginctl`** command to find your session number.

```
[student@workstation ~]$ loginctl
SESSION  UID USER    SEAT  TTY
      2 1000 student seat0 tty2
     c1   42 gdm     seat0 tty1

2 sessions listed.
```

4.2. Run the **`loginctl show-session`** command to find the session type.

```
[student@workstation ~]$ loginctl show-session 2 -p Type
Type=wayland
```

5. From the terminal, become **`root`** and then edit **`/etc/gdm/custom.conf`**. Uncomment the line containing **`WaylandEnable`**, save your changes, and then reboot **`workstation`**.

```
# Uncomment the line below to force the login screen to use Xorg
WaylandEnable=False
```

6. Log in as **`student`** and verify the session type using the **`loginctl`** command.

6.1. Open a terminal and run the **`loginctl`** command to find your session number.

```
[student@workstation ~]$ loginctl
SESSION  UID USER    SEAT  TTY
      2 1000 student seat0 tty2
     c1   42 gdm     seat0 tty1

2 sessions listed.
```

6.2. Run the **`loginctl show-session`** command to find the session type.

```
[student@workstation ~]$ loginctl show-session 2 -p Type
Type=x11
```

7. From the terminal, become **`root`** and edit **`/etc/gdm/custom.conf`**. Comment the line containing **`WaylandEnable`**, save your changes, and then reboot **`workstation`**.

```
# Uncomment the line below to force the login screen to use Xorg
#WaylandEnable=False
```

Finish

On **workstation**, run **lab core-wayland finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab core-wayland finish
```

This concludes the guided exercise.

Managing User Authentication with Authselect

Objectives

After completing this section, students should be able to manage user authentication settings in PAM, NSS, and dconf using Authselect, and explain the differences between Authselect and Authconfig.

Introducing Authselect

Red Hat Enterprise Linux 8 ships with Authselect, which simplifies the configuration of user authentication, and replaces Authconfig. Authselect uses a different and safer approach, based on profiles that make configuration changes simpler for system administrators. Authselect is used to configure the usual authentication parameters such as passwords, certificates, smart cards, and fingerprints.

Features of Authselect

- Adjusts PAM, NSS, and GNOME dconf settings.
- Ships with three ready-to-use profiles: **`sssd`**, **`winbind`**, and **`nis`**.
- **`pam_pwquality`** is enabled by default to enforce password quality restrictions on local users.

Comparing Authselect and Authconfig

- Authselect uses tested profiles, instead of directly modifying the system authentication configuration files.
- Authselect only modifies files in **`/etc/nsswitch.conf`**, **`/etc/pam.d/*`**, and **`/etc/dconf/db/distro.d/*`**.

How to Use Authselect

- Use the **`authselect list`** command to list the default and custom profiles.
- The default profiles are stored in **`/usr/share/authselect/default`**.
- Use the **`authselect create-profile`** command to create new custom profiles.
- Custom profiles are stored in the **`/etc/authselect/custom/`** directory.

When to use Authselect

- Use **`authselect`** in local and semi-centralized identity management environments, such as Winbind or NIS.
- Continue using **`ipa-client`** or **`realmd`** when joined to a Red Hat Enterprise Linux Identity Management, or Active Directory, domain. These tools correctly configure host authentication parameters on their own.

Note

The *authselect-compat* package provides a migration tool for **`/usr/sbin/authconfig`** that will translate some **`authconfig`** calls into **`authselect`** calls. It provides only minimum backward compatibility and you should use **`authselect`** instead.

References

`authselect`(8), **`authselect-migration`**(7), and **`authselect-profiles`**(5) man pages.

For more information, refer to the *Configuring authentication on a Red Hat Enterprise Linux host* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_authentication_on_a_red_hat_enterprise_linux_host

Guided Exercise

Managing User Authentication with Authselect

In this exercise, you will view and configure local user authentication settings using Authselect.

Outcomes

You should be able to configure identity and authorization parameters with Authselect, which replaces Authconfig.

1. Explore the Authselect application.

   1.1. Log in to **servera** then become the **root** user.

```
[student@workstation ~]$ ssh servera
[student@servera ~]$ sudo -i
[root@servera ~]#
```

   1.2. Authselect is installed by default in RHEL 8. Review the available options on **servera**

```
[root@servera ~]# authselect --help
Usage:
authselect COMMAND COMMAND-ARGS

Available commands:
- select           Select profile
- apply-changes    Regenerate configuration for currently selected command
- list             List available profiles
- show             Show profile information
- requirements     Print profile requirements
- current          Get identificator of currently selected profile
- check            Check if the current configuration is valid
- test             Print changes that would be otherwise written
- enable-feature   Enable feature in currently selected profile
- disable-feature  Disable feature in currently selected profile
- create-profile   Create new authselect profile

Common options:
  --debug          Print error messages
  --trace          Print trace messages
  --warn           Print warning messages

Help options:
  -?, --help       Show this for a command
  --usage          Show brief usage message for a command
```

1.3. List the available Authselect profiles.

```
[root@servera ~]# authselect list
- nis      Enable NIS for system authentication
- sssd     Enable SSSD for system authentication (also for local users only)
- winbind  Enable winbind for system authentication
```

1.4. List the currently selected Authselect profile.

```
[root@servera ~]# authselect current
Profile ID: sssd
Enabled features: None
```

1.5. Determine if the current configuration is valid.

```
[root@servera ~]# authselect check
Current configuration is valid.
```

1.6. Print the Authselect profile requirements.

```
[root@servera ~]# authselect requirements sssd
Make sure that SSSD service is configured and enabled. See SSSD documentation for
 more information.
```

1.7. Show the Authselect profile details and optional features.

```
[root@servera ~]# authselect show sssd
...output omitted...
AVAILABLE OPTIONAL FEATURES
---------------------------

with-faillock::
    Enable account locking in case of too many consecutive
    authentication failures.

with-mkhomedir::
    Enable automatic creation of home directories for users on their
    first login.

with-ecryptfs::
    Enable automatic per-user ecryptfs.

with-smartcard::
    Enable authentication with smartcards through SSSD. Please note that
    smartcard support must be also explicitly enabled within
    SSSD's configuration.

with-smartcard-lock-on-removal::
    Lock screen when a smartcard is removed.
    Note: "with-smartcard" must be set as well.

with-smartcard-required::
```

```
    Smartcard authentication is required. No other means of authentication
    (including password) will be enabled.
    Note: "with-smartcard" must be set as well.

with-fingerprint::
    Enable authentication with fingerprint reader through *pam_fprintd*.

with-silent-lastlog::
    Do not produce pam_lastlog message during login.

with-sudo::
    Allow sudo to use SSSD as a source for sudo rules in addition of /etc/sudoers.

with-pamaccess::
    Check access.conf during account authorization.

without-nullok::
    Do not add nullok parameter to pam_unix.
...output omitted...
```

2. Select and review the Authselect profiles.

   2.1. Select an Authselect profile on **servera** .

```
[root@servera ~]# authselect select sssd
Profile "sssd" was selected.
The following nsswitch maps are overwritten by the profile:
- passwd
- group
- netgroup
- automount
- services

Make sure that SSSD service is configured and enabled. See SSSD documentation for
 more information.
```

   2.2. Review the contents of **/etc/nsswitch.conf** on **servera**.

```
[root@servera ~]# grep sss /etc/nsswitch.conf
...output omitted...
passwd:     sss files systemd
group:      sss files systemd
netgroup:   sss files
automount:  sss files
services:   sss files
...output omitted...
shadow:     files sss
```

   2.3. Select another Authselect profile on **servera** to compare.

```
[root@servera ~]# authselect select nis
Profile "nis" was selected.
The following nsswitch maps are overwritten by the profile:
```

```
- aliases
- automount
- ethers
- group
- hosts
- initgroups
- netgroup
- networks
- passwd
- protocols
- publickey
- rpc
- services
- shadow

Make sure that NIS service is configured and enabled. See NIS documentation for
 more information.
```

2.4. Review the contents of **/etc/nsswitch.conf** on **servera** to compare both profiles.

```
[root@servera ~]# grep nis /etc/nsswitch.conf
...output omitted...
aliases:    files nis
automount:  files nis
ethers:     files nis
group:      files nis systemd
hosts:      files nis dns myhostname
initgroups: files nis
netgroup:   files nis
networks:   files nis
passwd:     files nis systemd
protocols:  files nis
publickey:  files nis
rpc:        files nis
services:   files nis
shadow:     files nis
...output omitted...
```

Notice the differences between both profiles in the **/etc/nsswitch.conf** configuration file.

3. Create a custom Authselect profile.

3.1. Create a custom Authselect profile based on the **sssd** profile on **servera**. Create symbolic links for meta files.

```
[root@servera ~]# authselect create-profile my-custom-profile \
> -b sssd --symlink-meta
New profile was created at /etc/authselect/custom/my-custom-profile
```

3.2. Review the contents of the custom Authselect profile on **/etc/authselect/custom/my-custom-profile/**.

```
[root@servera ~]# cd /etc/authselect/custom/my-custom-profile/
[root@servera my-custom-profile]# ls -lah
   total 32K
drwxr-xr-x. 2 root root  199 Feb  5 18:15 .
drwxr-xr-x. 3 root root   31 Feb  5 18:15 ..
-rw-r--r--. 1 root root  425 Feb  5 18:15 dconf-db
-rw-r--r--. 1 root root  224 Feb  5 18:15 dconf-locks
-rw-r--r--. 1 root root 2.1K Feb  5 18:15 fingerprint-auth
-rw-r--r--. 1 root root  393 Feb  5 18:15 nsswitch.conf
-rw-r--r--. 1 root root 2.9K Feb  5 18:15 password-auth
-rw-r--r--. 1 root root  588 Feb  5 18:15 postlogin
lrwxrwxrwx. 1 root root   41 Feb  5 18:15 README -> /usr/share/authselect/default/
sssd/README
lrwxrwxrwx. 1 root root   47 Feb  5 18:15 REQUIREMENTS -> /usr/share/authselect/
default/sssd/REQUIREMENTS
-rw-r--r--. 1 root root 1.9K Feb  5 18:15 smartcard-auth
-rw-r--r--. 1 root root 3.4K Feb  5 18:15 system-auth
```

Notice the PAM, dconf, and **nsswitch.conf** templates, and the symbolic links for the **README** and **REQUIREMENTS** meta files.

3.3. Review the contents of the **nsswitch.conf** template in the custom Authselect profile.

```
[root@servera my-custom-profile]# cat nsswitch.conf
passwd:     sss files systemd   {exclude if "with-custom-passwd"}
group:      sss files systemd   {exclude if "with-custom-group"}
netgroup:   sss files           {exclude if "with-custom-netgroup"}
automount:  sss files           {exclude if "with-custom-automount"}
services:   sss files           {exclude if "with-custom-services"}
sudoers:    files sss           {include if "with-sudo"}
```

3.4. Modify the **services** and **sudoers** parameters in the custom **nsswitch.conf** template, as shown below:

```
[root@servera my-custom-profile]# vim nsswitch.conf
...output omitted...
services:   files sss           {exclude if "with-custom-services"}
sudoers:    sss files           {exclude if "with-custom-sudoers"}
```

In this exercise, this modification has no other purpose than allowing you to review those changes in the resulting **/etc/nsswitch.conf** file in a later step.

3.5. List the available Authselect profiles.

```
[root@servera my-custom-profile]# authselect list
 -nis                     Enable NIS for system authentication
- sssd                    Enable SSSD for system authentication (also for local
 users only)
- winbind                 Enable winbind for system authentication
- custom/my-custom-profile  Enable SSSD for system authentication (also for local
 users only)
```

3.6. Select the custom Authselect profile.

```
[root@servera my-custom-profile]# authselect select custom/my-custom-profile
Profile "custom/my-custom-profile" was selected.
The following nsswitch maps are overwritten by the profile:
- passwd
- group
- netgroup
- automount
- services
- sudoers

Make sure that SSSD service is configured and enabled. See SSSD documentation for
 more information.
```

3.7. Review the modified options in the **/etc/nsswitch.conf** configuration file.

```
[root@servera my-custom-profile]# grep sss /etc/nsswitch.conf
...output omitted...
passwd:     sss files systemd
group:      sss files systemd
netgroup:   sss files
automount:  sss files
services:   files sss
sudoers:    sss files
...output omitted...
```

Notice the changes in the **passwd**, **group**, **netgroup**, **automount** and the custom parameters **services** and **sudoers**.

4. Clean up the Authselect configuration.

4.1. List the current Authselect profile on **servera**.

```
[root@servera my-custom-profile]# authselect current
Profile ID: custom/my-custom-profile
Enabled features: None
```

4.2. Select the default **sssd** Authselect profile on **servera**.

```
[root@servera my-custom-profile]# authselect select sssd
Profile "sssd" was selected.
The following nsswitch maps are overwritten by the profile:
- passwd
- group
- netgroup
- automount
- services

Make sure that SSSD service is configured and enabled. See SSSD documentation for
 more information.
```

4.3. Delete the custom Authselect profile folder in **/etc/authselect/custom/**.

```
[root@servera my-custom-profile]# cd
[root@servera ~]# rm -rf /etc/authselect/custom/my-custom-profile
```

4.4. List the available Authselect profiles.

```
[root@servera ~]# authselect list
 -nis                       Enable NIS for system authentication
- sssd                      Enable SSSD for system authentication (also for local
 users only)
- winbind                   Enable winbind for system authentication
```

4.5. Log off from **servera**.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

This concludes the guided exercise.

Configuring NTP with Chrony

Objectives

After completing this section, students should be able to maintain NTP time synchronization using Chrony, and configure the time zone with **`timedatectl`**.

Chrony Replaces ntpd

The *Network Time Protocol (NTP)* is the standard for machines to provide and synchronize time between authorized systems. A machine may get accurate time information from public NTP services on the Internet such as the NTP Pool Project. A high-quality hardware clock to serve accurate time to local clients is another option.

In Red Hat Enterprise Linux 8, Chrony is the only available implementation of an NTP server. The ntpd implementation, available alongside Chrony on Red Hat Enterprise Linux 7, is no longer available.

Using Chrony as the Default NTP Implementation

- Chrony replaces ntpd, which is no longer available.
- Chrony uses a different configuration file, with a different format.
- The **`/usr/share/doc/chrony/ntp2chrony.py`** script converts your **`/etc/ntp.conf`** file to **`/etc/chrony.conf`**.
- **`timedatectl`** displays an overview of the current time related system settings.
- Use **`timedatectl set-timezone`** to define the system time zone.

Introducing Chrony

The **`chronyd`** service keeps the local hardware clock (RTC) on track by synchronizing it with the configured NTP servers. If no network connectivity is available, it synchronizes to the calculated RTC clock drift, which is recorded in the **`driftfile`** variable specified in the **`/etc/chrony.conf`** configuration file.

By default, **`chronyd`** uses servers from the NTP Pool project for the time synchronization and does not need additional configuration. It may be useful to change the NTP servers when the machine in question is on an isolated network.

The **`server`** lines in the **`/etc/chrony.conf`** configuration file give the IP addresses or DNS names of the NTP servers. Following the server IP address or name, you can list a series of options for the server. Red Hat recommends using the **`iburst`** option because, after the service starts, Chrony takes four measurements in a short period for a more accurate initial clock synchronization.

```
[user@demo ~]$ cat /etc/chrony.conf
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
# pool 2.rhel.pool.ntp.org iburst
server 192.168.122.14 iburst
server tick.example.com iburst
server tock.example.com iburst
```

```
# Record the rate at which the system clock gains/losses time.
driftfile /var/lib/chrony/drift
...output omitted...
```

The **pool** directive is similar to the **server** directive, but instead of indicating a single NTP server, it indicates a DNS name that resolves to multiple addresses that may change over time.

After updating the **/etc/chrony.conf** configuration file, restart the service.

```
[root@demo ~]# systemctl restart chronyd
```

The **chronyc** command acts as a client to the **chronyd** service. After setting up NTP synchronization, verify the NTP server used to synchronize the system clock with the **chronyc sources** command or, for a more verbose output, **chronyc sources -v**:

```
[user@demo ~]$ chronyc sources -v
210 Number of sources = 3

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured offset,
||                                \     |          |  zzzz = estimated error.
||                                 |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample
^* tick.example.com              3   7   377    87    +70us[  +90us] +/-   63ms
^+ tock.example.com              1   6    37    23  +1384us[ +409us] +/-   58ms
^+ 192.168.122.14                3   6    37    23  -1514us[-2490us] +/-   84ms
```

The asterisk character (*) in the **S** (Source state) field indicates that the **tick.example.com** server has been used as a time source and is the NTP server the machine is currently synchronized to.

Converting the NTPd Configuration to Chrony

The Chrony configuration file, **/etc/chrony.conf**, is similar to the ntpd configuration file, **/etc/ntp.conf**. Some configuration directives are specific to Chrony because the two programs have different features.

Notably, the following ntpd features are not available in Chrony.

- Chrony does not implement the broadcast and multicast client features. Therefore, the **broadcastclient** and **multicastclient** directives do not exist in **chrony.conf**. These NTPd features enable reception of NTP messages sent by broadcast or multicast NTP servers.
- The **autokey** directive is not available in **chrony.conf**. In **ntp.conf**, this directive activates public key algorithms for packet authentication between NTP clients and servers.
- Chrony does not support ephemeral associations when using symmetric peer directives. For example, the **peer** directive allows two servers at the same level to synchronize their clocks. When only one server defines the **peer** directive, the other server accepts the connection and establishes an ephemeral association. Chrony does not allow this implicit configuration; both servers must include the **peer** directive.

To help you convert your **ntp.conf** file to **chrony.conf**, the *chrony* package provides the **ntp2chrony.py** script in **/usr/share/doc/chrony/**. Use the **--ntp-conf** option to specify the **ntp.conf** file to convert. The **--chrony-conf** option specifies the output **chrony.conf** file. The **--chrony-keys** option specifies the path to the output file containing the key pairs for servers requiring authentication. Review the converted configuration file before restarting the **chronyd** service.

```
[root@demo ~]# python3 /usr/share/doc/chrony/ntp2chrony.py \
> --ntp-conf ./ntp.conf \
> --chrony-conf ./chrony.conf \
> --chrony-keys ./chrony.key
[root@demo ~]# cat ./chrony.conf
...output omitted...
[root@demo ~]# cp ./chrony.conf ./chrony.key /etc
[root@demo ~]# systemctl restart chronyd
```

Setting the Time Zone

The **timedatectl** command displays an overview of the current time-related system settings, including the current time, time zone, and NTP synchronization settings of the system.

```
[user@demo ~]$ timedatectl
               Local time: Fri 2019-02-15 03:54:03 EST
           Universal time: Fri 2019-02-15 08:54:03 UTC
                 RTC time: Fri 2019-02-15 08:54:02
                Time zone: America/New_York (EST, -0500)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

The system provides a database with the known time zones that you can list with the **timedatectl list-timezones** command.

```
[user@demo ~]$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
...output omitted...
```

The **tzselect** command is useful for identifying the correct name for a time zone. It interactively prompts the user with questions about the system's location and outputs the name of the correct time zone. It does not make any changes to the time zone setting of the system.

As **root**, you can adjust the system setting for the current time zone with the **timedatectl set-timezone** command.

```
[root@demo ~]# timedatectl set-timezone America/Phoenix
[root@demo ~]# timedatectl
               Local time: Fri 2019-02-15 01:58:19 MST
           Universal time: Fri 2019-02-15 08:58:19 UTC
                 RTC time: Fri 2019-02-15 08:58:18
```

```
                Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

References

`chrony.conf`(5), **`chronyc`**(1), **`timedatectl`**(1), and **`tzselect`**(8) man pages.

For more information on Chrony, refer to the *Using the Chrony suite to configure NTP* chapter in the *Configuring basic system settings* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#using-chrony-to-configure-ntp

Guided Exercise

Configuring NTP with Chrony

In this exercise, you will adjust the time zone on a server and ensure that its system clock is synchronized with an NTP time source, using Chrony.

Outcomes

Students should be able to configure the **servera** system to use the time zone appropriate for Haiti and configure **chronyd** on **servera** to use the NTP server running on **classroom.example.com** as the time source.

Before You Begin

Log in to **workstation** as **student** using **student** as the password. On **workstation**, run the **lab core-chrony start** command to verify that the environment is ready and to prepare the systems for the exercise.

```
[student@workstation ~]$ lab core-chrony start
```

1. Your **servera** machine has been relocated to Haiti. Set the appropriate time zone and verify your work.

   1.1. Log in to **servera** then become the **root** user.

```
[student@workstation ~]$ ssh servera
[student@servera ~]$ sudo -i
[root@servera ~]#
```

   1.2. Identify the correct time zone for Haiti.

```
[root@servera ~]# tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
 1) Africa
 2) Americas
...output omitted...
#? 2
Please select a country whose clocks agree with yours.
...output omitted...
10) Canada            28) Haiti             46) Suriname
11) Caribbean NL      29) Honduras          47) Trinidad & Tobago
...output omitted...
#? 28

The following information has been given:

  Haiti
```

```
Therefore TZ='America/Port-au-Prince' will be used.
Selected time is now:  Fri Feb 15 01:24:44 EST 2019.
Universal Time is now: Fri Feb 15 06:24:44 UTC 2019.
Is the above information OK?
1) Yes
2) No
#? 1

You can make this change permanent for yourself by appending the line
  TZ='America/Port-au-Prince'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
America/Port-au-Prince
```

1.3. Change the time zone to America/Port-au-Prince.

```
[root@servera ~]# timedatectl set-timezone America/Port-au-Prince
```

1.4. Retrieve the time zone set on **servera** to verify your work.

```
[root@servera ~]# timedatectl
               Local time: Fri 2019-02-15 01:45:31 EST
           Universal time: Fri 2019-02-15 06:45:31 UTC
                 RTC time: Fri 2019-02-15 06:47:25
                Time zone: America/Port-au-Prince (EST, -0500)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: no
```

2. Enable NTP synchronization on the **servera** system and use **classroom.example.com** as the time source.

2.1. Configure **chronyd** to synchronize the time on **servera** with **classroom.example.com**. Edit **/etc/chrony.conf** to include the following configuration file excerpt:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
# pool 2.rhel.pool.ntp.org iburst
server classroom.example.com iburst
...output omitted...
```

2.2. Start and enable the **chronyd** service, turning on the NTP synchronization on **servera**.

```
[root@servera ~]# timedatectl set-ntp true
```

3. Confirm that the clock on **servera** is synchronized with **classroom.example.com**.

3.1. Verify the clock synchronization status.

```
[root@servera ~]# timedatectl
...output omitted...
System clock synchronized: yes
...output omitted...
```

3.2. Confirm that **classroom.example.com** is the time source for the clock synchronization on **servera**.

```
[root@servera ~]# chronyc sources -v
210 Number of sources = 1

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||                                                /   xxxx = adjusted offset,
||         Log2(Polling interval) -.             |    yyyy = measured offset,
||                                  \            |    zzzz = estimated error.
||                                   |           |
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* classroom.example.com         8   6    37    51   -25ns[-703us] +/-  128us
```

3.3. Log off from **servera**.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

Finish

On **workstation**, run **lab core-chrony finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab core-chrony finish
```

This concludes the guided exercise.

Managing Python Versions in Red Hat Enterprise Linux 8

Objectives

After completing this section, students should be able to identify, install, and control the specific versions of Python available to a server.

Python 2 to Python 3 Transition

The RHEL 8 default minimum installation does not install the user-level Python binaries, only the version for operating system tooling. The Python 2.7 and Python 3.6 application binaries are available from the Application Stream (AppStream) repository for Red Hat Enterprise Linux 8.

Installing Python on RHEL 8

- RHEL 8 includes Yum modules for both Python 2.7 and Python 3.6.
- Python 3.6 is the default. Python 2.7 is a short life release to assist users to migrate to RHEL 8 while their applications are being ported to python3.
- Future releases of Python 3.x will be available from the AppStream repository as Yum modules named **`python3x`**.

Working with Python in Red Hat Enterprise Linux 8

- By default, Red Hat Enterprise Linux 8 does not provide a **`/usr/bin/python`** command. Use the **`alternatives`** command to link to the desired release for command line simplicity.
- Do not use **`/usr/bin/python`** in scripts. Always write scripts using **`/usr/bin/python3`** or **`/usr/bin/python2`**, because their behavior will always be version dependent.
- System tools written in Python use separate python3 installation called **`/usr/libexec/platform-python`**. Do not use this version in your scripts as it only includes python modules required for operating system tooling.

Setting up Alternative Python Versions

The Python 2.7 module installs the **`/usr/bin/python2`** binary. Python 3.6 installs the **`/usr/bin/python3`** binary. Use the **`alternatives`** command to link **`/usr/bin/python`** to either **`/usr/bin/python2`**, or **`/usr/bin/python3`**, recommended only for interactive use.

References

For more information, refer to the *Using Python in Red Hat Enterprise Linux 8* chapter in the *Red Hat Enterprise Linux 8 Configuring Basic System Settings Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#using-python3_configuring-basic-system-settings

Guided Exercise

Managing Python Versions in RHEL 8

In this exercise, you will ensure that both Python 3 and Python 2 are installed on a server, and use the **alternatives** command to control whether an unversioned Python executable is available and which version of Python it uses.

Outcomes

You should be able to install Python 2.7 and Python 3.6, and configure the **python** command to link to one of their binaries.

1. On **servera**, verify that the **python** command does not exist by default on Red Hat Enterprise Linux 8.

   1.1. Log in to **servera** then become the **root** user.

```
[student@workstation ~]$ ssh servera
[student@servera ~]$ sudo -i
[root@servera ~]#
```

   1.2. Verify that the **python** command does not exist by default.

```
[root@servera ~]# which python
/usr/bin/which: no python in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/
root/bin)
```

2. Install Python 2.7 and verify that it is available.

   2.1. Use Yum to install Python 2.7 using the **python27** module.

```
[root@servera ~]# yum module install python27
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

   2.2. Verify that Python 2.7 is available.

```
[root@servera ~]# python2 -V
Python 2.7.15
```

3. Install Python 3.6 and verify that it is available.

   3.1. Use Yum to install Python 3.6 using the **python36** module.

```
[root@servera ~]# yum module install python36
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

3.2. Verify that Python 3.6 is available.

```
[root@servera ~]# python3 -V
Python 3.6.8
```

4. Use the **alternatives** command to enable **/usr/bin/python3**. When done, log off from **servera**.

4.1. Use the **alternatives** command to point **/usr/bin/python** to **/usr/bin/python3**.

```
[root@servera ~]# alternatives --config python
There are 3 programs which provide 'python'.

  Selection    Command
-----------------------------------------------
*+ 1           /usr/libexec/no-python
   2           /usr/bin/python3
   3           /usr/bin/python2

Enter to keep the current selection[+], or type selection number: 3
```

4.2. Verify that the **python** command uses Python 3.6.

```
[root@servera ~]# python -V
Python 3.6.8
```

4.3. Log off from **servera**.

```
[root@servera ~]# exit
[student@servera ~]$ exit
[student@workstation ~]$
```

This concludes the guided exercise.

Lab

Adapting to Core System Changes

Performance Checklist

In this lab, you will make changes to user authentication, configure NTP time synchronization, and adjust the Python configuration on a server.

Outcomes

You should be able to configure identity and authorization parameters with the new Authselect tool, including to configure the **serverb** system to use the time zone appropriate for Suriname, configure **chronyd** on **serverb** to use the NTP server running on **classroom.example.com** as a time source, and ensure that both Python 3 and Python 2 are installed on a server.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab core-review start** command.

```
[student@workstation ~]$ lab core-review start
```

1. On **serverb**, create a new **lab-profile** Authselect profile based on the existing **sssd** profile.
2. Your **serverb** machine has been relocated to Suriname in South America. Set the appropriate time zone.
3. Enable NTP synchronization on the **serverb** system and use **classroom.example.com** as the time source.
4. Confirm that the clock on **serverb** is synchronized with **classroom.example.com**.
5. On **serverb**, install Python 2.7 and Python 3.6, and configure the **python** command to point to the Python 3.6 binaries. Point **/usr/bin/python** to **/usr/bin/python3** command.
6. Run the **lab core-review grade** command on **workstation** to grade your work.

Finish

On **workstation**, run **lab core-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab core-review finish
```

This concludes the lab.

Solution

Adapting to Core System Changes

Performance Checklist

In this lab, you will make changes to user authentication, configure NTP time synchronization, and adjust the Python configuration on a server.

Outcomes

You should be able to configure identity and authorization parameters with the new Authselect tool, including to configure the **serverb** system to use the time zone appropriate for Suriname, configure **chronyd** on **serverb** to use the NTP server running on **classroom.example.com** as a time source, and ensure that both Python 3 and Python 2 are installed on a server.

Before You Begin

Log in to **workstation** as **student** using **student** as the password.

On **workstation**, run the **lab core-review start** command.

```
[student@workstation ~]$ lab core-review start
```

1. On **serverb**, create a new **lab-profile** Authselect profile based on the existing **sssd** profile.

   1.1. Log in to **serverb** then become the **root** user.

```
[student@workstation ~]$ ssh root@serverb
[student@serverb ~]$ sudo -i
[root@serverb ~]#
```

   1.2. Create a new **lab-profile** Authselect profile based on the existing **sssd** profile.

```
[root@serverb ~]# authselect create-profile lab-profile \
> -b sssd --symlink-meta
New profile was created at /etc/authselect/custom/lab-profile
```

2. Your **serverb** machine has been relocated to Suriname in South America. Set the appropriate time zone.

   2.1. Identify the correct time zone for Suriname/South America.

```
[root@serverb ~]# tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
 1) Africa
 2) Americas
```

```
...output omitted...
#? 2
Please select a country whose clocks agree with yours.
...output omitted...
 9) Brazil           27) Guyana            45) St Vincent
10) Canada           28) Haiti             46) Suriname
11) Caribbean NL     29) Honduras          47) Trinidad & Tobago
...output omitted...
#? 46

The following information has been given:

  Suriname

Therefore TZ='America/Paramaribo' will be used.
Selected time is now: Tue Feb 19 18:51:03 -03 2019.
Universal Time is now: Tue Feb 19 21:51:03 UTC 2019.
Is the above information OK?
1) Yes
2) No
#? 1

You can make this change permanent for yourself by appending the line
 TZ='America/Paramaribo'; export TZ
to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
America/Paramaribo
```

2.2. Change the time zone to America/Paramaribo.

```
[root@serverb ~]# timedatectl set-timezone America/Paramaribo
```

3. Enable NTP synchronization on the **serverb** system and use **classroom.example.com** as the time source.

3.1. Configure **chronyd** to synchronize the time on **serverb** with **classroom.example.com**. Edit **/etc/chrony.conf** to include the following configuration file excerpt:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
# pool 2.rhel.pool.ntp.org iburst
server classroom.example.com iburst
...output omitted...
```

3.2. Restart and enable the **chronyd** service.

```
[root@serverb ~]# systemctl restart chronyd
[root@serverb ~]# systemctl enable chronyd
```

3.3. Enable NTP synchronization on **serverb**.

```
[root@serverb ~]# timedatectl set-ntp true
```

4. Confirm that the clock on **serverb** is synchronized with **classroom.example.com**.

   4.1. Verify the clock synchronization status.

```
[root@serverb ~]# timedatectl
...output omitted...
System clock synchronized: yes
...output omitted...
```

   4.2. Confirm that **classroom.example.com** is the time source for clock synchronization on **serverb**.

```
[root@serverb ~]# chronyc sources -v
210 Number of sources = 1

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||                                                /   xxxx = adjusted offset,
||         Log2(Polling interval) -.             |    yyyy = measured offset,
||                                  \            |    zzzz = estimated error.
||                                   |           |
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* classroom.example.com         8   6    37    51   -25ns[-703us] +/-  128us
```

5. On **serverb**, install Python 2.7 and Python 3.6, and configure the **python** command to point to the Python 3.6 binaries. Point **/usr/bin/python** to **/usr/bin/python3** command.

   5.1. Use Yum to install Python 2.7 using the **python27** module.

```
[root@serverb ~]# yum module install python27
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

   5.2. Verify that Python 2.7 is available.

```
[root@serverb ~]# python2 -V
Python 2.7.15
```

   5.3. Use Yum to install Python 3.6 using the **python36** module.

```
[root@serverb ~]# yum module install python36
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

5.4. Verify that Python 3.6 is available.

```
[root@serverb ~]# python3 -V
Python 3.6.8
```

5.5. Use the **alternatives** command to point **/usr/bin/python** to **/usr/bin/python3**.

```
[root@serverb ~]# alternatives --config python
There are 3 programs which provide 'python'.

  Selection    Command
-----------------------------------------------
*+ 1           /usr/libexec/no-python
   2           /usr/bin/python2
   3           /usr/bin/python3

Enter to keep the current selection[+], or type selection number: 3
```

5.6. Verify that the **python** command uses Python 3.6.

```
[root@serverb ~]# python -V
Python 3.6.8
```

5.7. Log off from **serverb**.

```
[root@serverb ~]# exit
[student@serverb ~]$ exit
[student@workstation ~]$
```

6. Run the **lab core-review grade** command on **workstation** to grade your work.

Finish

On **workstation**, run **lab core-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab core-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- Wayland is the new windowing system designed to replace X11 in Red Hat Enterprise Linux 8.
- RHEL 8 introduces the new Authselect utility, which simplifies the configuration of user authentication, replacing the Authconfig utility.
- On RHEL 8, Chrony is the implementation of the Network Time Protocol (NTP).
- A RHEL 8 default installation does not include Python. Both Python 2.7 and Python 3.6 are available on the AppStream repository.

Chapter 5

Implementing Storage Using New Features

Goal

Describe the major enhancements in local and remote file system and volume management components.

Objectives

- Describe the modern file system designs that include volume and pool management features.
- Describe new storage driver enhancements that include runtime compression and deduplication.
- Configure NFS service with new tools and an awareness of the enhancements and changes to NFS in RHEL 8.

Sections

- Managing Layered Storage with Stratis (and Guided Exercise)
- Compressing and Deduplicating Storage with VDO (and Guided Exercise)
- Administering NFS Enhancements (and Guided Exercise)

Lab

Implementing Storage Using New Features

Managing Layered Storage with Stratis

Objectives

After completing this section, students should be able to describe modern file system designs that include volume and pool management features.

Stratis Storage Manager

Red Hat Enterprise Linux 8 includes the Stratis storage manager, which supports management of collections of block devices to create flexible file systems. The combined file system and volume management functionality first learned in ZFS or Btrfs is now available in Stratis. Both the Btrfs and the ZFS file systems are unsupported and no longer available in Red Hat Enterprise Linux 8. Stratis also provides advanced storage features like thin provisioning, snapshotting, and monitoring.

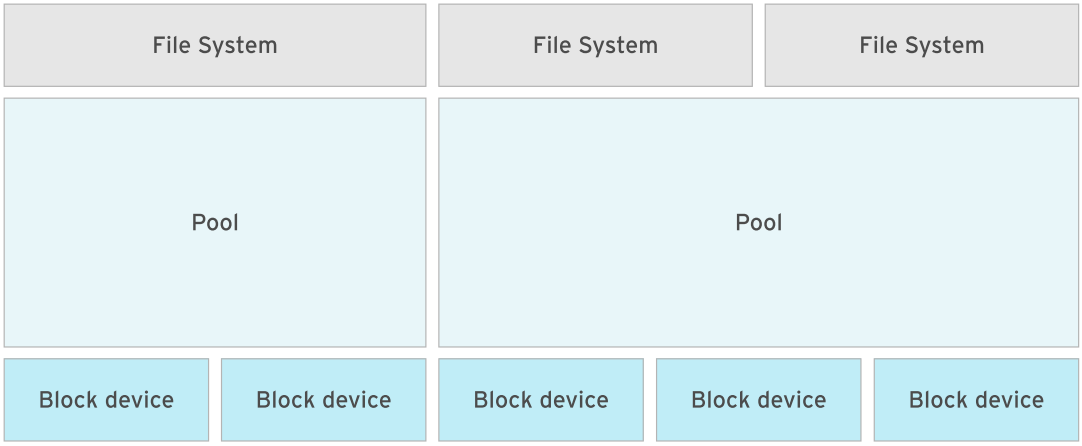


Figure 5.1: Stratis storage manager elements

Implementing a Volume-managing File System with Stratis

- Stratis is a volume managing file system (VMF).
- Volume managing file systems integrate the file system in the volume itself, in contrast with LVM where the volume requires a file system on top of it.
- Stratis' design assumes SSD as the default storage type, or at least as a cache tier, so the focus of the design is on improving flexibility and reliability.
- **`BtrFS`** and **`ZFS`** are no longer supported or available in Red Hat Enterprise Linux 8.
- Stratis provides advanced features like thin provisioning, snapshotting, and monitoring.

Stratis File System Persistent Mounting

- Persistent Stratis file system mounts in **`/etc/fstab`** may require using the **`x-systemd.requires=stratisd.service`** mount option.
- Use the **`blkid`** command to locate the file system UUID foruse in **`/etc/fstab`**.

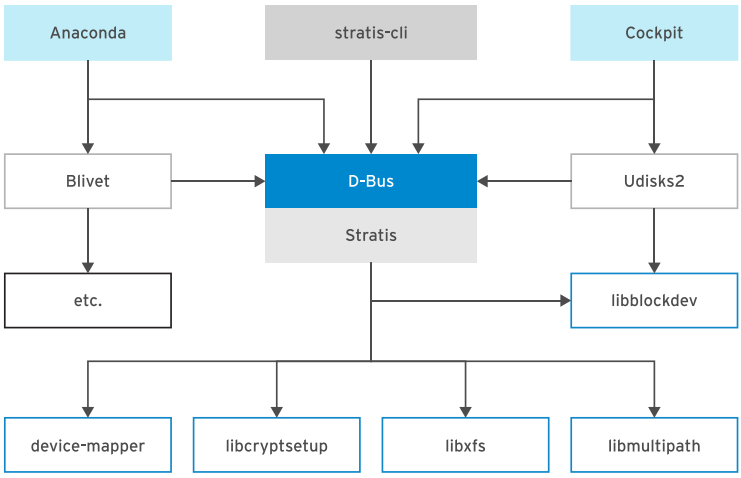


Figure 5.2: Stratis in the Linux Storage Management Stack

Describing Stratis Layers

- Stratis manages pools which are a collection of block devices. Stratis file systems are created within those pools. Those file systems incorporate both the volume and the file system, by default XFS.
- Internally, Stratis uses the Backstore subsystem to manage the block devices, and the Thinpool subsystem to manage the thin-provisioned pools.
- The Backstore has a data tier which maintains the on-disk metadata on block devices, and detects and corrects data corruption.
- The cache tier uses high-performance block devices to act as a cache on top of the data tier.
- The Thinpool subsystem manages the thin-provisioned volumes associated with the Stratis file systems. This subsystem uses the **`dm-thin`** device mapper driver to replace LVM on the virtual volume sizing and management. These volumes are created with a large virtual size, and are formatted with XFS. If the Stratis file system's virtual size is approached, the file system is automatically enlarged.

Note

As Stratis volumes are thin-provisioned, tools like **`df`** will report incorrect sizes.

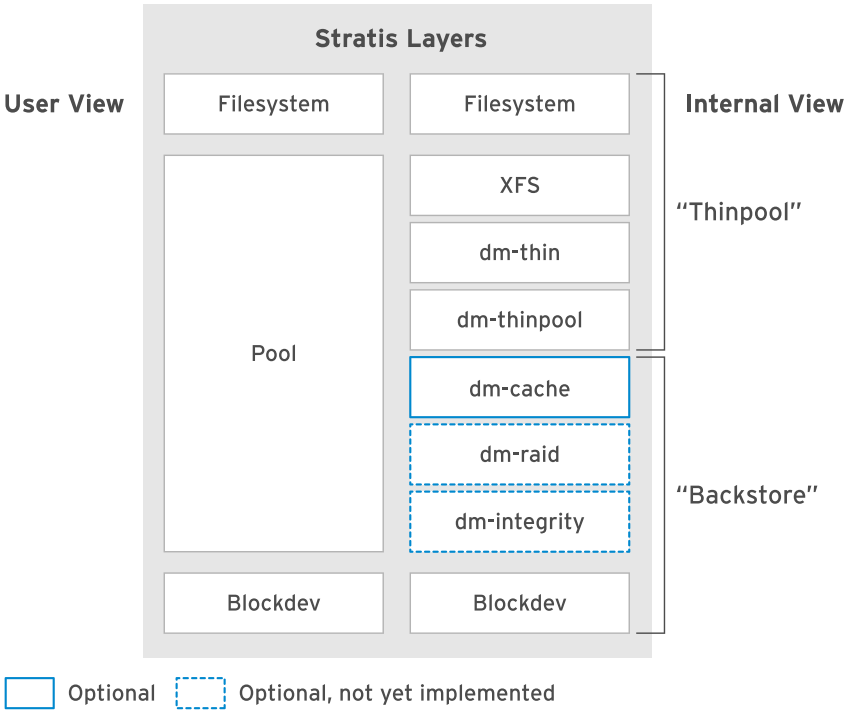


Figure 5.3: Stratis layers

Managing Volume Managed File Systems with the Stratis Storage Manager

- Create pools of one or several block devices with the **`stratis pool create`** command.
- Add additional block devices to a pool with the **`stratis pool add-data`** command.
- Create dynamic and flexible file systems on top of pools with the **`stratis filesystem create`** command.
- Stratis supports file system snapshotting with the **`stratis fs snapshot`** command. Snapshots are independent of the source file systems.
- You can use the D-Bus API to communicate with the **`stratisd`** daemon.

Stratis simplifies the storage stack

Stratis simplifies many aspects of local storage provisioning and configuration. For example, installing the OS to a Stratis pool using Anaconda. After selecting the disks to use for the pool, the filesystem sizing workflow is omitted. Anaconda could use the Stratis API directly, instead of needing work in Blivet to build an API on top of command line tools. Other tools and products like Cockpit, RHV, and Atomic would be simpler and less error-prone using Stratis for storage and snapshots, because they do not worry about per-filesystem sizing but only that the pool has enough "backing store". The API allows better tool-to-tool integration than using any CLI programmatically.

References

For more information, refer to the *Managing layered local storage with Stratis* chapter in the *Red Hat Enterprise Linux 8 Configuring and Managing File Systems Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/managing_file_systems

Stratis Storage
https://stratis-storage.github.io/

What Stratis learned from ZFS, Btrfs, and Linux Volume Manager
https://opensource.com/article/18/4/stratis-lessons-learned

Guided Exercise

Managing Layered Storage with Stratis

In this exercise, you will use the Stratis Storage Manager to create pools, volumes and file systems that work in cooperation.

Outcomes

You should be able to use the Stratis storage manager to combine disks into pools and create volumes and file systems, and show how file systems and volumes share pools cooperatively.

1. Install the Stratis storage manager on **servera**.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. Install the **stratisd** and the **stratis-cli** packages on **servera**.

```
[root@servera ~]# yum install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

   1.3. Enable the **stratisd** service on **servera** to start at boot time.

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

2. Use the Stratis storage manager to create a new pool with the **/dev/vdb** device.

   2.1. Create the **stratispool** pool with the **/dev/vdb** device.

```
[root@servera ~]# stratis pool create stratispool /dev/vdb
```

   2.2. Verify that the **stratispool** pool is now available.

```
[root@servera ~]# stratis pool list
Name        Total Physical Size  Total Physical Used
stratispool              1 GiB               52 MiB
```

3. Add **/dev/vdc** as an additional device to the **stratispool** pool.

   3.1. Add **/dev/vdc** as an additional device to the **stratispool** pool.

```
[root@servera ~]# stratis pool add-data stratispool /dev/vdc
```

3.2. Verify that **/dev/vdc** is part of the **stratispool** pool.

```
[root@servera ~]# stratis blockdev
Pool Name   Device Node    Physical Size   State  Tier
stratispool /dev/vdb              1 GiB  In-use  Data
stratispool /dev/vdc              1 GiB  In-use  Data
```

4. Create the **filesystem1** file system in the **stratispool** pool.

4.1. Create the **filesystem1** file system in the **stratispool** pool.

```
[root@servera ~]# stratis fs create stratispool filesystem1
```

4.2. Verify that the **filesystem1** file system is now available.

```
[root@servera ~]# stratis fs list stratispool
Pool Name       Name         Used     Created            Device
stratispool     filesystem1  546 MiB  Jan 23 2019 08:44  /stratis/stratispool/
filesystem1
```

5. Mount the **filesystem1** file system, and create a new file on it.

5.1. Create a new mount point, named **/stratisvol**.

```
[root@servera ~]# mkdir /stratisvol
```

5.2. Mount the **filesystem1** file system into the **/stratisvol** directory.

```
[root@servera ~]# mount /stratis/stratispool/filesystem1 /stratisvol
```

5.3. Verify that the **filesystem1** file system is now accessible.

```
[root@servera ~]# mount
... output omitted ...
/dev/mapper/stratis-1-b5ceb83a7c57488a9c02b501d1419ce4-thin-
fs-333a2d9922c04ee6abebcc71bdb047b5 on /stratisvol type xfs
 (rw,relatime,seclabel,attr2,inode64,sunit=2048,swidth=2048,noquota)
```

5.4. Create a new empty file named **file.txt**.

```
[root@servera ~]# touch /stratisvol/file.txt
```

6. Create a snapshot of the **filesystem1** file system. When done, remove the file you previously created.

6.1. Create a snapshot, named **filesystem1-snapshot**, of the **filesystem1** file system.

```
[root@servera ~]# stratis fs snapshot stratispool filesystem1 filesystem1-snapshot
```

6.2. Remove the **file.txt** file.

```
[root@servera ~]# rm -f /stratisvol/file.txt
```

7. Inspect the **filesystem1-snapshot** snapshot you just created, and verify that the file is available.

7.1. Create a mount point named **/stratisvolsnap**, for the snapshot.

```
[root@servera ~]# mkdir /stratisvolsnap
```

7.2. Mount the **filesystem1-snapshot** snapshot in the **/stratisvolsnap** directory.

```
[root@servera ~]# mount /stratis/stratispool/filesystem1-snapshot /stratisvolsnap/
```

7.3. Verify that the **file.txt** file is available.

```
[root@servera ~]# ls /stratisvolsnap
file.txt
```

8. Destroy the **filesystem1-snapshot** snapshot.

8.1. Unmount the **filesystem1-snapshot** snapshot.

```
[root@servera ~]# umount /stratis/stratispool/filesystem1-snapshot
```

8.2. Destroy the **filesystem1-snapshot** snapshot.

```
[root@servera ~]# stratis filesystem destroy stratispool filesystem1-snapshot
```

9. Destroy the **filesystem1** file system.

9.1. Unmount the **filesystem1** file system.

```
[root@servera ~]# umount /stratis/stratispool/filesystem1
```

9.2. Destroy the **filesystem1** file system.

```
[root@servera ~]# stratis filesystem destroy stratispool filesystem1
```

10. Destroy the **stratispool** pool. When done, log out from **servera**.

10.1. Destroy the **stratispool** pool.

```
[root@servera ~]# stratis pool destroy stratispool
```

10.2. Verify that the **stratispool** pool is not available.

```
[root@servera ~]# stratis pool list
Name    Total Physical Size  Total Physical Used
```

10.3. Log out from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

This concludes the guided exercise.

Compressing and Deduplicating Storage with VDO

Objectives

After completing this section, students should be able to describe new storage driver enhancements that include run time compression and deduplication.

Virtual Data Optimizer

RHEL 8 includes the virtual data optimizer (VDO) driver, which optimizes the data footprint on block devices. VDO includes two utilities, **`vdo`** to manage and configure VDO volumes, and **`vdostats`** to display the usage and block I/O for a VDO volume.

Reducing Data Footprint with VDO

- Virtual data optimizer (VDO) is a Linux device mapper driver to reduces disk space usage on block devices, and minimize replication bandwidth.
- VDO includes two kernel modules, the **`kvdo`** module to control data compression, and the **`uds`** module for deduplication.
- VDO supports both inline data deduplication and compression.
- Persistent VDO mounts in /etc/fstab require using the **`x-systemd.requires=vdo.service`** mount option.

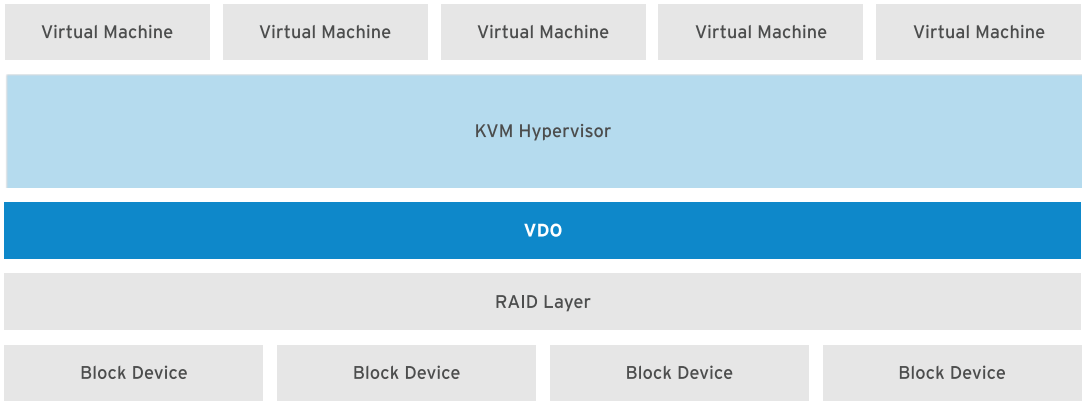


Figure 5.4: VDO-based virtual machines

Virtual Data Optimization Phases

- VDO reduces the data footprint on storage in three phases: zero-block elimination, deduplication of redundant blocks, and data compression.
- VDO removes blocks which only include zeros, and keeps their metadata.
- The universal deduplication service (UDS) kernel module reviews the available VDO metadata to detect duplicated blocks. If a duplicated block is found, this block points to the metadata of the already available block.
- When done with zero-block elimination and deduplication, the **`kvdo`** kernel module compresses blocks using LZ4 compression and groups them on 4 KB blocks.

Structure of a VDO volume

- A VDO volume contains two parts: UDS and VDO
- The UDS part includes the name and location of each block to support the block deduplication
- The VDO part includes the VDO volume data and metadata

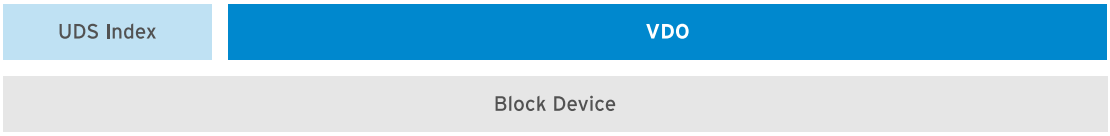


Figure 5.5: Structure of a VDO volume

References

For more information, refer to the *Deploying VDO* chapter in the *Red Hat Enterprise Linux 8 Deduplicating and Compressing Storage Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/deduplicating_and_compressing_storage/deploying-vdo_deduplicating-and-compressing-storage

Introducing Virtual Data Optimizer

https://rhelblog.redhat.com/2018/04/11/introducing-virtual-data-optimizer-to-reduce-cloud-and-on-premise-storage-costs/

Guided Exercise

Compressing and Deduplicating Storage with VDO

In this exercise, you will perform file activities that demonstrate the integrated compression and deduplication features.

Outcomes

You should be able to see the effects of deduplication on a Virtual Data Optimizer (VDO) volume.

1. On **servera**, create a VDO volume named **vdo1**, with the **/dev/vdd** device. Set its logical size to **20 GB**.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. Create the **vdo1** volume, using **/dev/vdd** as the device. Use **20 GB** as its logical size.

```
[root@servera ~]# vdo create --name=vdo1 --device=/dev/vdd --vdoLogicalSize=20G
Creating VDO vdo1
Starting VDO vdo1
Starting compression on VDO vdo1
VDO instance 0 volume is ready at /dev/mapper/vdo1
```

   1.3. Verify that the **vdo1** volume is available.

```
[root@servera ~]# vdo list
vdo1
```

2. Verify that both compression and deduplication features are enabled on the **vdo1** volume.

   2.1. Verify that deduplication is enabled on the **vdo1** volume.

```
[root@servera ~]# vdo status -n vdo1 | grep Deduplication
  Deduplication: enabled
```

   2.2. Verify that compression is enabled on the **vdo1** volume.

```
[root@servera ~]# vdo status -n vdo1 | grep Compression
  Compression: enabled
```

3. Create an XFS file system on the **vdo1** volume, and mount it on **/mnt/vdo1**.

3.1. Create an XFS file system on the **vdo1** volume. Wait until the system registers the device node.

```
[root@servera ~]# mkfs.xfs -K /dev/mapper/vdo1
...output omitted...
[root@servera ~]# udevadm settle
```

3.2. Create the **/mnt/vdo1** directory.

```
[root@servera ~]# mkdir -m 1777 /mnt/vdo1
```

3.3. Mount the **vdo1** volume on **/mnt/vdo1**.

```
[root@servera ~]# mount /dev/mapper/vdo1 /mnt/vdo1/
```

4. Create several copies of a file on the **vdo1** volume to verify how deduplication takes place.

4.1. Verify the status of the **vdo1** volume. When created, a VDO volume reserves around 3 or 4 GB for itself.

```
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1         20.0G      4.0G     16.0G  20%           98%
```

4.2. Download the `http://content.example.com/rhel8.0/x86_64/dvd/images/install.img` file, and copy it to the **/mnt/vdo1** directory. It may take up to a minute to download it.

```
[root@servera ~]# wget \
> http://content.example.com/rhel8.0/x86_64/dvd/images/install.img
[root@servera ~]# cp install.img /mnt/vdo1
```

4.3. Verify the status of the **vdo1** volume to see the changes in **Use%**, and **Space saving%**.

```
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1         20.0G      4.4G     15.6G  22%            3%
```

4.4. Create another copy of the **install.img** file in the **/mnt/vdo1** directory.

```
[root@servera ~]# cp install.img /mnt/vdo1/install.img.2
```

4.5. Verify the status of the **vdo1** volume, and the size increase of **Space saving%**.

```
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1         20.0G      4.5G     15.5G  22%           50%
```

4.6. Create an additional copy of the **install.img** file in the **/mnt/vdo1** directory.

```
[root@servera ~]# cp install.img /mnt/vdo1/install.img.3
```

4.7. Verify the status of the **vdo1** volume, and the size increase of **Space saving%**.

```
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1         20.0G      4.5G     15.5G  22%           67%
```

5. Remove the **vdo1** volume. When done, log out from **servera**.

5.1. Unmount the **vdo1** volume.

```
[root@servera ~]# umount /mnt/vdo1
```

5.2. Remove the **vdo1** volume.

```
[root@servera ~]# vdo remove --name=vdo1
Removing VDO vdo1
Stopping VDO vdo1
```

5.3. Log out from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

This concludes the guided exercise.

Administering NFS Enhancements

Objectives

After completing this section, students should be able to configure NFS service with new tools and an awareness of enhancements and changes to NFS in RHEL 8.

Using NFS on Red Hat Enterprise Linux 8

Red Hat Enterprise Linux 8 supports the use of version 4.2 of the *network file system (NFS)*.

Red Hat Enterprise Linux 8 NFS Enhancements

- The default NFS version in Red Hat Enterprise Linux 8 is 4.2. NFSv4 and NFSv3 major versions are supported, but NFSv2 is no longer supported.
- The NFS configuration file is now **`/etc/nfs.conf`**. The **`/etc/sysconfig/nfs`** configuration file is deprecated.
- The introduction of the new **`nfsconf`** command to manage NFS configuration files.
- RHEL 8 removes the **`nfsnobody`** user, and changes the UID and GID of the **`nobody`** user and group to **`65534`**.
- NFSv4 no longer requires the **`rpcbind`** service to operate, eliminating the need for UDP connections.

NFS 4.2 Features

- Server-side copy enables the NFS client to instruct the NFS server to copy data directly from one NFS share to another. This helps to optimize resources because the copied data is not transmitted over the network.
- Sparse files enable files to have multiple holes, which are data blocks consisting only of zeroes. These are transferred as zeroes improving the overall speed.
- Space reservation allows users to reserve free space on NFS servers, which prevents the NFS share from running out of space.
- Labeled NFS enforces data access rights and enables SELinux on NFS file systems.

The `nfsconf` command

RHEL 8 introduces the **`nfsconf`** command to manage the NFS client and server configuration files.

- The **`nfsconf`** command handles NFSv4 and NFSv3 configuration.
- The default **`nfsconf`** configuration file is **`/etc/nfs.conf`**.
- The **`nfsconf --file *filename*`** command loads the specified configuration file.
- The **`nfsconf`** command can get, set, or test NFS configuration files.

References

`nfsconf`(8), and **`nfs.conf`**(5) man pages.

For more information, refer to the *Mounting NFS shares* chapter in the *Red Hat Enterprise Linux 8 Managing file Systems Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/managing_file_systems

Network File System (NFS) Version 4 Minor Version 2 Protocol
https://tools.ietf.org/html/rfc7862

Guided Exercise

Administering NFS Enhancements

In this exercise, you will configure NFS using the **nfsconf** utility and investigate changes to the NFS service in Red Hat Enterprise Linux 8.

Outcomes

You should be able to install and configure an NFSv4-only Server, to provide shared storage, and use autofs on client side to consume it.

1. Install all necessary packages to export NFS shares on **servera**.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. Install the **nfs-utils** package on **servera**, if not already installed.

```
[root@servera ~]# yum install nfs-utils
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

   1.3. Enable the **nfs-server** service on **servera** to start at boot time.

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

2. Using the **nfsconf** command to configure **/etc/nfs.conf**, enable NFS Server to work in just version 4.X, and also ensure that TCP mode is enabled and UDP mode is disabled.

   2.1. Review the available tags and sections in the **/etc/nfs.conf** file.

```
[root@servera ~]# cat /etc/nfs.conf
#
# This is a general configuration for the
# NFS daemons and tools
...output omitted...
[nfsd]
# debug=0
# threads=8
# host=
# port=0
# grace-time=90
# lease-time=90
# tcp=y
```

```
# vers2=n
# vers3=y
# vers4=y
# vers4.0=y
# vers4.1=y
# vers4.2=y
# rdma=n
#
...output omitted...
```

2.2. Disable the tags **udp**, **vers2**, **vers3**, configure them with **nfsconf** tool.

```
[root@servera ~]# nfsconf --set nfsd udp n
[root@servera ~]# nfsconf --set nfsd vers2 n
[root@servera ~]# nfsconf --set nfsd vers3 n
```

2.3. Enable the tags **vers4**, **vers4.0**, **vers4.1**, **vers4.2** , configure them with **nfsconf** tool.

```
[root@servera ~]# nfsconf --set nfsd tcp y
[root@servera ~]# nfsconf --set nfsd vers4 y
[root@servera ~]# nfsconf --set nfsd vers4.0 y
[root@servera ~]# nfsconf --set nfsd vers4.1 y
[root@servera ~]# nfsconf --set nfsd vers4.2 y
```

2.4. Disable listening for the **RPCBIND**, **MOUNT**, and **NSM** protocol calls, which are not necessary in the NFSv4-only case. Disable related services:

```
[root@servera ~]# systemctl mask --now rpc-statd.service \
> rpcbind.service rpcbind.socket
Created symlink /etc/systemd/system/rpc-statd.service → /dev/null.
Created symlink /etc/systemd/system/rpcbind.service → /dev/null.
Created symlink /etc/systemd/system/rpcbind.socket → /dev/null.
```

2.5. Restart the **nfs-server** service to apply the changes.

```
[root@servera ~]# systemctl restart nfs-server
```

3. Configure **firewalld** on **servera** to allow incoming connections for the NFSv4 Server.

3.1. Inspect which ports are listening, on **servera** as **root** with the **ss** command.

```
[root@servera ~]$ ss --listening --tcp --udp
Netid  State   Recv-Q  Send-Q  Local Address:Port  Peer Address:Port
udp    UNCONN  0       0           127.0.0.1:323       0.0.0.0:*
udp    UNCONN  0       0               [::1]:323          [::]:*
tcp    LISTEN  0       128           0.0.0.0:ssh       0.0.0.0:*
tcp    LISTEN  0       64            0.0.0.0:nfs       0.0.0.0:*
tcp    LISTEN  0       128              [::]:ssh          [::]:*
tcp    LISTEN  0       64               [::]:nfs          [::]:*
tcp    LISTEN  0       128                 *:websm           *:*
```

3.2. Configure **firewalld** on **servera** to allow NFSv4 service.

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

3.3. Make the **firewalld** changes persistent with **firewall-cmd**.

```
[root@servera ~]# firewall-cmd --runtime-to-permanent
success
```

3.4. List the services enabled on **firewalld**.

```
[root@servera ~]# firewall-cmd --list-services
cockpit dhcpv6-client nfs ssh
```

4. Configure **servera** to export content with NFSv4 Server.

4.1. Create a folder on **servera** to export content.

```
[root@servera ~]# mkdir /exports
```

4.2. Change the **/exports** folder ownership to the user **nobody** and group **nobody**.

```
[root@servera ~]# chown nobody:nobody /exports
```

4.3. Configure and restore **SELinux** contexts on the **/exports** folder.

```
[root@servera ~]# semanage fcontext -a -t public_content_rw_t "/exports(/.*)?"
[root@servera ~]# restorecon -vvFR /exports/
```

4.4. Export the **/exports** folder by creating a new configuration file in **/etc/exports.d/**

```
[root@servera ~]# echo "/exports *(rw,security_label)" > \
> /etc/exports.d/example.exports
```

4.5. Reload the NFSv4 exports:

```
[root@servera ~]# exportfs -r
```

4.6. Log out from **servera**.

```
[root@servera ~]# exit
Connection to servera closed.
```

5. Configure **serverb** as client system of **servera** NFSv4 exports, and use **autofs**.

5.1. Log in to **serverb** as the **root** user.

```
[student@workstation ~]$ ssh root@serverb
```

5.2. Install the **autofs** package on **serverb**.

```
[root@serverb ~]# yum install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

5.3. Enable the **autofs** service on **serverb** to start at boot time.

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

5.4. Create a mount point folder on **serverb**.

```
[root@serverb ~]# mkdir -p /mnt/exports
```

5.5. Configure a new master map file, name it as **/etc/auto.master.d/example.autofs** on **serverb**

```
[root@serverb ~]# echo "/mnt/exports  /etc/auto.exports" > \
> /etc/auto.master.d/example.autofs
```

5.6. Configure **/etc/auto.exports** with a **map-file** on **serverb**

```
[root@serverb ~]# cat << EOF > /etc/auto.exports
> example  -fstype=nfs4  servera.lab.example.com:/exports/example
> EOF
```

5.7. Restart the **autofs** service to apply the changes.

```
[root@serverb ~]# systemctl restart autofs
```

6. Test the **autofs** and NFSv4 Server, create some small text content file on **servera** and review it on **serverb**. When done, log out from **serverb**.

6.1. Switch to **servera** to create a subdirectory **/exports/example** and then create a text file inside the folder.

```
[root@servera ~]# mkdir -p /exports/example
[root@servera ~]# echo "Test from $(hostname)" > /exports/example/TEST
```

6.2. On **serverb**, use **autofs** to explore the on-demand mount point exported on **servera**. Verify the **SELinux** context.

```
[root@serverb ~]# cd /mnt/exports/example
[root@serverb example]# ls -lZ
tal 4
-rw-r--r--. 1 root root system_u:object_r:public_content_rw_t:s0 34 May 19 17:24
 TEST
[root@serverb example]# cat TEST
Test from servera.lab.example.com
```

6.3. Explore the **mount** options for the NFSv4 auto mounted share.

```
[root@serverb example]# mount | grep nfs4
servera.lab.example.com:/exports/example on /mnt/exports/example type nfs4
(rw,relatime,seclabel,vers=4.2,rsize=524288,wsize=524288,namlen=255,hard,
proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.11,local_lock=none,
addr=172.25.250.10)
```

6.4. Log out from **serverb**.

```
[root@serverb example]# exit
[student@workstation ~]$
```

This concludes the guided exercise.

Lab

Implementing Storage Using New Features

Performance Checklist

In this lab, you will use the Stratis storage manager to create a file system, and configure a VDO volume.

Outcomes

You should be able to create a file system with the Stratis storage manager, and a VDO volume.

1. On **servera**, configure a pool with the Stratis storage manager with the **/dev/vdb** and the **/dev/vdc** devices.
2. Create a file system in the previous pool, and create a snapshot.
3. Remove the file system, its snapshot, and the pool.
4. On **servera**, create a VDO volume using **/dev/vdd** with a size of 30 GB. When done, create an XFS file system on it, and mount it.
5. Remove the VDO volume. When done, log out from **servera**.

This concludes the lab.

Solution

Implementing Storage Using New Features

Performance Checklist

In this lab, you will use the Stratis storage manager to create a file system, and configure a VDO volume.

Outcomes

You should be able to create a file system with the Stratis storage manager, and a VDO volume.

1. On **servera**, configure a pool with the Stratis storage manager with the **/dev/vdb** and the **/dev/vdc** devices.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. If not previously done, install the **stratisd** and the **stratis-cli** packages on **servera**.

```
[root@servera ~]# yum install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

   1.3. If not previously done, enable the **stratisd** service on **servera** to start at boot time.

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

   1.4. Verify that the **stratisd** service is available.

```
[root@servera ~]# systemctl status stratisd
```

   1.5. Create the **labpool** pool with the **/dev/vdb** and the **/dev/vdc** devices.

```
[root@servera ~]# stratis pool create labpool /dev/vdb /dev/vdc
```

   1.6. Verify that the **labpool** pool is now available.

```
[root@servera ~]# stratis pool list
Name        Total Physical Size  Total Physical Used
labpool              2 GiB             56 MiB
```

2. Create a file system in the previous pool, and create a snapshot.

2.1. Create the **labfs** file system in the **labpool** pool.

```
[root@servera ~]# stratis fs create labpool labfs
```

2.2. Verify that the **labfs** file system is now available.

```
[root@servera ~]# stratis fs list labpool
Pool Name       Name         Used     Created            Device
labpool     labfs  546 MiB  Jan 23 2019 08:44  /stratis/labpool/labfs
```

2.3. Create a new mount point named **/stratislab**.

```
[root@servera ~]# mkdir /stratislab
```

2.4. Mount the **labfs** file system into the **/stratislab** directory.

```
[root@servera ~]# mount /stratis/labpool/labfs /stratislab
```

2.5. Verify that the **labfs** file system is now accessible.

```
[root@servera ~]# mount
... output omitted ...
/dev/mapper/stratis-1-b5ceb83a7c57488a9c02b501d1419ce4-thin-
fs-333a2d9922c04ee6abebcc71bdb047b5 on /stratislab type xfs
 (rw,relatime,seclabel,attr2,inode64,sunit=2048,swidth=2048,noquota)
```

2.6. Create a snapshot of the **labfs** file system. Name the snapshot **labfs-snapshot**.

```
[root@servera ~]# stratis fs snapshot labpool labfs labfs-snapshot
```

3. Remove the file system, its snapshot, and the pool.

3.1. Destroy the **labfs-snapshot** snapshot.

```
[root@servera ~]# stratis filesystem destroy labpool labfs-snapshot
```

3.2. Unmount the **labfs** file system.

```
[root@servera ~]# umount /stratis/labpool/labfs
```

3.3. Destroy the **labfs** file system.

```
[root@servera ~]# stratis filesystem destroy labpool labfs
```

3.4. Destroy the **labpool** pool.

```
[root@servera ~]# stratis pool destroy labpool
```

3.5. Verify that the **labpool** pool is not available.

```
[root@servera ~]# stratis pool list
Name    Total Physical Size  Total Physical Used
```

4. On **servera**, create a VDO volume using **/dev/vdd** with a size of 30 GB. When done, create an XFS file system on it, and mount it.

4.1. Create the **vdolab** volume with the **/dev/vdd** device. Use **30 GB** as its logical size.

```
[root@servera ~]# vdo create --name=vdolab --device=/dev/vdd --vdoLogicalSize=30G
Creating VDO vdolab
Starting VDO vdolab
Starting compression on VDO vdolab
VDO instance 0 volume is ready at /dev/mapper/vdolab
```

4.2. Verify that the **vdolab** volume is available.

```
[root@servera ~]# vdo list
vdolab
```

4.3. Create an XFS file system on the **vdolab** volume. Wait until the system registers the device node.

```
[root@servera ~]# mkfs.xfs -K /dev/mapper/vdolab
...output omitted...
[root@servera ~]# udevadm settle
```

4.4. Create the **/mnt/vdolab** directory.

```
[root@servera ~]# mkdir -m 1777 /mnt/vdolab
```

4.5. Mount the **vdolab** volume on **/mnt/vdolab**.

```
[root@servera ~]# mount /dev/mapper/vdolab /mnt/vdolab/
```

5. Remove the VDO volume. When done, log out from **servera**.

5.1. Unmount the **vdolab** volume.

```
[root@servera ~]# umount /mnt/vdolab
```

5.2. Remove the **vdolab** volume.

```
[root@servera ~]# vdo remove --name=vdolab
Removing VDO vdolab
Stopping VDO vdolab
```

5.3. Log out from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

This concludes the lab.

Summary

In this chapter, you learned:

- Red Hat Enterprise Linux 8 includes the Stratis storage manager, which supports the management of collections of block devices to create flexible file systems.
- Stratis also provides advanced storage features, like thin provisioning, snapshotting, and monitoring.
- The virtual data optimizer (VDO) is a kernel module that uses zero-block elimination, deduplication of redundant blocks, and data compression to reduce disk space usage and minimize replication bandwidth.
- Red Hat Enterprise Linux 8 supports NFS 4.2, which includes the new **`/etc/nfs.conf`** configuration file, removes the **`nfsnobody`** user, and changes the UID and GID of the **`nobody`** user and group to **`65534`**.
- The **`nfsconf`** tool supports the management of the NFS 4.2 server and client configuration files.

Chapter 6

Managing Containers with the New Runtime

Goal	Explain the new container runtime engine and tools which replace the **`docker`** container engine.
Objectives	• Describe the new container engine and utilities and observe the planned similarity in syntax and function, and the increase in performance and features.
Sections	• Deploying Containers with the New Container Runtime (and Guided Exercise)
Lab	Managing Containers with the New Runtime

Deploying Containers with the New Container Runtime

Objectives

After completing this section, students should be able to describe the new container engine and utilities and observe the planned similarity in syntax and function, and the increase in performance and features.

The Podman Container Engine

Red Hat Enterprise Linux 8 includes the *container-tools* package module, which provides a new container engine named Podman which replaces Docker and Moby. The *container-tools* package module also contains other tools such as Buildah to build container images, Skopeo to manage container images on registries, and runc. In contrast to Docker, which depends on daemons to build and run containers, this new toolset and container engine allow building and running containers without daemons.

The New Container Runtime Toolset

- RHEL 8 replaces Docker with a new container runtime which supports most of the Docker functionality.
- RHEL 8's container runtime toolset supports Open Container Initiative (OCI) standards, which for example enables reusing third-party containers images.
- The container runtime provides a daemon-less container engine. This architecture does not require an active **`root`**-privileged daemon to run containers. Users run containers without **`root`** privileges.
- This architecture uses a fork/exec model, which enhances integration with the kernel's **`audit`** security feature. This replaces the Docker client/server model, which uses what **`audit`** refers to as the *unset audit UID*.
- The *container-tools* package module provides the new container runtime toolset and engine.

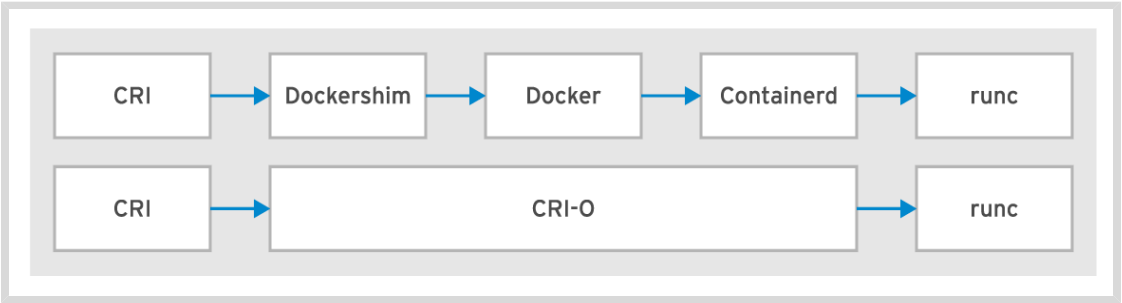


Figure 6.1: From Docker to RHEL 8's container runtime

Describing the new Container Runtime Toolset

- The podman container engine is daemon-less and supports the execution of containers.
- The **`podman`** syntax is similar to the **`docker`** command, and also supports Dockerfile use.
- Buildah builds containers images, either from scratch or from 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.

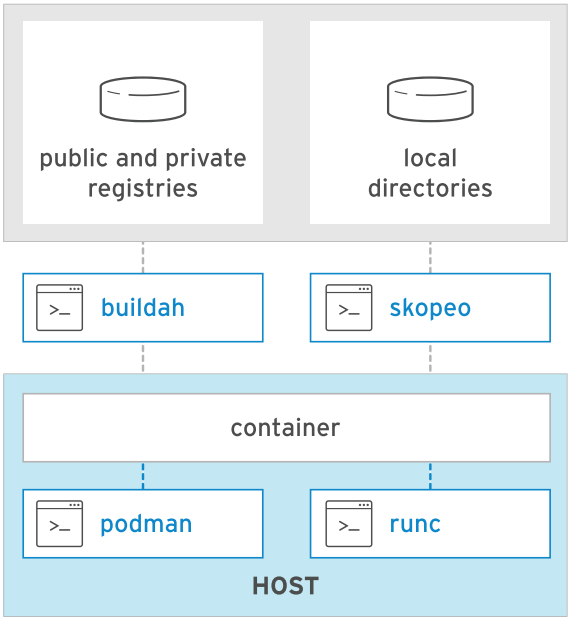


Figure 6.2: New Container Runtime

Pacemaker Resource Bundles

Describing Pacemaker Resource Bundles

- RHEL 8 includes Pacemaker container bundles with podman as a technology preview.
- A Pacemaker bundle supports the execution of the same container across all hosts belonging to a specific node type, for example an OpenStack controller node.
- A bundle also maps the required storage inside the container directories, and customizes specific attributes in the container.
- Red Hat OpenStack Platform currently supports Pacemaker bundles.

References

For more information, refer to the *Working with containers* and *Building container images with Buildah* chapters in the *Red Hat Enterprise Linux 8 Building, Running, and Managing Containers Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/building_running_and_managing_containers

Containers without daemons: Podman and Buildah
https://developers.redhat.com/blog/2018/11/20/buildah-podman-containers-without-daemons/

Knowledgebase: Pacemaker 2.0 upgrade in Red Hat Enterprise Linux 8
https://access.redhat.com/articles/3681151/

Guided Exercise

Deploying Containers with the New Container Runtime

In this exercise, you will build an image and deploy it as a containerized application using the new container utilities.

Outcomes

You should be able to create a container image and deploy a container using the new container tools.

1. Install the container software packages.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. Install the **container-tools** module.

```
[root@servera ~]# yum module install container-tools
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2. Create a container from scratch and configure it to run **bash**. Label this container **rhel-base**.

   2.1. Create a new container from scratch.

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

   2.2. Mount the root filesystem of the **working-container** container.

```
[root@servera ~]# buildah mount working-container
/var/lib/containers/storage/overlay/240a...b6ce/merged
```

   2.3. Install the *bash* and the *coreutils* packages on the **working-container** container.

```
[root@servera ~]# yum install \
> --installroot /var/lib/containers/storage/overlay/240a...b6ce/merged \
> bash coreutils \
> --releasever 8 \
> --setopt install_weak_deps=false
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2.4. Clean up the **yum** cache on the **working-container** container.

```
[root@servera ~]# yum clean all \
> --installroot /var/lib/containers/storage/overlay/240a...b6ce/merged \
> --releasever 8
...output omitted...
12 files removed
```

2.5. Configure **bash** as the first command to run.

```
[root@servera ~]# buildah config --cmd /bin/bash working-container
```

2.6. Label the **working-container** container **rhel-base**.

```
[root@servera ~]# buildah config --label name=rhel-base working-container
```

3. Create a container image named **rhel-base** based on the **working-container** container.

3.1. Unmount the **working-container** container.

```
[root@servera ~]# buildah unmount working-container
e6d1...2850
```

3.2. Create the **rhel-base** container image based on the **working-container** container.

```
[root@servera ~]# buildah commit working-container rhel-base
...output omitted...
Writing manifest to image destination
Storing signatures
b6d8...0a6a
```

3.3. Verify that the **rhel-base** container image is available.

```
[root@servera ~]# buildah images
IMAGE NAME           IMAGE TAG  IMAGE ID       CREATED AT          SIZE
localhost/rhel-base  latest     b6d88a1866eb   Feb 4, 2019 07:20   291 MB
```

4. Create a new container with the **rhel-base** container image.

4.1. Review the specifications of the **rhel-base** container image.

```
[root@servera ~]# podman inspect localhost/rhel-base
...output omitted...
        "Config": {
            "Cmd": [
                "/bin/bash"
            ],
            "Labels": {
                "name": "rhel-base"
            }
        },
...output omitted...
```

4.2. Create a container using the **rhel-base** container image. Use the **--rm** option to delete the container when you exit it. When done, exit the container.

```
[root@servera ~]# podman run --rm -it localhost/rhel-base /bin/bash
bash-4.4# exit
exit
[root@servera ~]#
```

5. Remove the **working-container** container and the **rhel-base** container image. When done, log off from **servera**.

5.1. Remove the **working-container** container.

```
[root@servera ~]# buildah rm working-container
e724...797f
```

5.2. Remove the **rhel-base** container image.

```
[root@servera ~]# buildah rmi rhel-base
...output omitted...
```

5.3. Log off from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

This concludes the guided exercise.

Lab

Managing Containers with the New Runtime

Performance Checklist

In this lab, you will use the new container runtime to build and run a custom container.

Outcomes

You should be able to build and deploy a custom container using the new container runtime.

1. Install the new container runtime software packages on **servera**.
2. Build a new container from scratch on **servera** and label it **custom-container**.
3. Prepare the new container for installing software on **servera**.
4. Customize the new container to run an **apache** web server, with some text content and listening on TCP port 80, on **servera**.
5. Create a new container image based on your custom container and name it **custom-image**, on **servera**.
6. Deploy and test a container based on the new custom container image, on **servera**.
7. Try accessing the web server running inside the container from **serverb**. When done, log off from **serverb**.
8. Clean up the exercise data on **servera**. When done, log off from **servera**.

This concludes the lab.

Solution

Managing Containers with the New Runtime

Performance Checklist

In this lab, you will use the new container runtime to build and run a custom container.

Outcomes

You should be able to build and deploy a custom container using the new container runtime.

1. Install the new container runtime software packages on **servera**.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. Install the **container-tools** module, if not already installed.

```
[root@servera ~]# yum module install container-tools
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2. Build a new container from scratch on **servera** and label it **custom-container**.

   2.1. Use **buildah** to create a new container from scratch.

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

   2.2. List the available containers.

```
[root@servera ~]# buildah containers
CONTAINER ID  BUILDER  IMAGE ID     IMAGE NAME     CONTAINER NAME
bc4c2a71c4c9     *                 scratch        working-container
```

   2.3. Label the newly created container **custom-container**.

```
[root@servera ~]# buildah config --label name=custom-container working-container
```

   2.4. Inspect the container.

```
[root@servera ~]# buildah inspect working-container
...output omitted...
    "Container": "working-container",
```

```
    "ContainerID":
 "bc4c2a71c4c9fbb11b7b88127a67173b82d0e4f95e7935c4a36b8d3010b8de16",
    "MountPoint": "",
    "ProcessLabel": "",
    "MountLabel": "",
    "ImageAnnotations": null,
    "ImageCreatedBy": "",
    "OCIv1": {
        "created": "2019-02-07T05:24:20.845648888Z",
        "architecture": "amd64",
        "os": "linux",
        "config": {
            "Labels": {
                "name": "custom-container"
...output omitted...
```

3. Prepare the new container for installing software on **servera**.

3.1. Mount the root file system of **working-container**.

Note
Verify the correct mount point for your case.

```
[root@servera ~]# buildah mount working-container
/var/lib/containers/storage/overlay/14e1...fbed/merged
```

3.2. Initialize the RPM database within the container.

```
[root@servera ~]# rpm --root \
> /var/lib/containers/storage/overlay/14e1...fbed/merged \
> --initdb
```

3.3. Download the *redhat-release-server* RPM on **servera**.

```
[root@servera ~]# yum install yum-utils
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

```
[root@servera ~]# yumdownloader --destdir=/tmp redhat-release-server
...output omitted...
redhat-release-8.0-0.44.el8.x86_64.rpm            3.4 MB/s |  45 kB   00:00
```

3.4. Install the *redhat-release-server* RPM on **working-container**.

```
[root@servera ~]# rpm -ivh \
> --root /var/lib/containers/storage/overlay/14e1...fbed/merged \
> /tmp/redhat-release*.rpm
...output omitted...
Updating / installing...
   1:redhat-release-8.0-0.44.el8      ################################# [100%]
```

3.5. Copy the local repository file to the **working-container** mount point.

```
[root@servera ~]# cp \
> /etc/yum.repos.d/rhel_dvd.repo \
> /var/lib/containers/storage/overlay/14e1...fbed/merged/etc/yum.repos.d/
```

4. Customize the new container to run an **apache** web server, with some text content and listening on TCP port 80, on **servera**.

4.1. Install the *httpd* package in the container.

```
[root@servera ~]# yum install \
> --installroot /var/lib/containers/storage/overlay/14e1...fbed/merged \
> httpd
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

4.2. Create some text content and expose it as an **index.html** on **working-container** mount directory.

```
[root@servera ~]# echo "custom-container Working OK" > \
> /var/lib/containers/storage/overlay/14e1...fbed/merged/var/www/html/index.html
```

4.3. Configure **/usr/sbin/httpd** as the first command to run in **working-container**.

```
[root@servera ~]# buildah config --cmd "/usr/sbin/httpd -DFOREGROUND" \
> working-container
```

4.4. Configure the internal listening port for httpd daemon in **working-container**.

```
[root@servera ~]# buildah config --port 80/tcp working-container
```

4.5. Clean up the Yum cache on the **working-container** container.

```
[root@servera ~]# yum clean all \
> --installroot /var/lib/containers/storage/overlay/14e1...fbed/merged \
...output omitted...
```

4.6. Unmount the root file system of **working-container**.

```
[root@servera ~]# buildah unmount working-container
14e1392a5a7d06da689e26c1438b9092008f9c012affbae943ee25720ecbfbed
```

5. Create a new container image based on your custom container and name it **custom-image**, on **servera**.

   5.1. Create the container image **custom-image** based on the **working-container** container.

```
[root@servera ~]# buildah commit working-container custom-image
Getting image source signatures
Copying blob
 sha256:02f83791177ef46b84dfbf49d491971a9a83f793af08a507c1a744f842366010
 144.82 MiB / 144.82 MiB [==========================================================] 9s
Copying config
 sha256:09901c90da2a4ea033e8083316983605e09dd34860729f1047f2d78ecea06558
 332 B / 332 B [============================================================] 0s
Writing manifest to image destination
Storing signatures
09901c90da2a4ea033e8083316983605e09dd34860729f1047f2d78ecea06558
```

   5.2. List the available images on **servera**.

```
[root@servera ~]# buildah images
IMAGE NAME              IMAGE TAG  IMAGE ID      CREATED AT         SIZE
localhost/custom-image  latest     09901c90da2a  Feb 6, 2019 22:22  483 MB
```

6. Deploy and test a container based on the new custom container image, on **servera**.

   6.1. Run a container based on the **localhost/custom-image** image.

```
[root@servera ~]# podman run -d -p 80:80 --name lab-container
 localhost/custom-image
1aedf5420e73b71105b02bc7fdbea406b3313ec29bde7fa3f661c1ccf4fc1a7a
```

   6.2. Test the container **httpd** daemon.

```
[root@servera ~]# curl localhost
custom-container Working OK
```

   6.3. Use firewalld to open the HTTP service port on **servera**.

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

7. Try accessing the web server running inside the container from **serverb**. When done, log off from **serverb**.

   7.1. Open a new terminal and log in to **serverb** as the **student** user.

```
[student@workstation ~]$ ssh student@serverb
```

7.2. Test the container **httpd** daemon inside **servera**.

```
[student@serverb ~]$ curl http://servera
custom-container Working OK
```

7.3. Log off from **serverb**.

```
[student@serverb ~]$ exit
[student@workstation ~]$
```

8. Clean up the exercise data on **servera**. When done, log off from **servera**.

8.1. Switch to the **servera** terminal, and stop the running container.

```
[root@servera ~]# podman stop lab-container
1aedf5420e73b71105b02bc7fdbea406b3313ec29bde7fa3f661c1ccf4fc1a7a
```

8.2. Delete the container on **servera**.

```
[root@servera ~]# podman rm lab-container
1aedf5420e73b71105b02bc7fdbea406b3313ec29bde7fa3f661c1ccf4fc1a7a
```

8.3. Delete the container image on **servera**.

```
[root@servera ~]# podman rmi localhost/custom-image
a8fe25c677f2878d73ace848d7fd2f2bf06f1470c94e33fee9a444eecd8e49d9
```

8.4. Delete the **working-container** container on **servera**.

```
[root@servera ~]# buildah delete working-container
bc4c2a71c4c9fbb11b7b88127a67173b82d0e4f95e7935c4a36b8d3010b8de16
```

8.5. Log off from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

This concludes the lab.

Summary

In this chapter, you learned:

- Red Hat Enterprise Linux 8 includes the *container-tools* package module, which provides a new container engine named Podman, which replaces **`docker`** and **`moby`**.
- The *container-tools* package module also contains other tools such as **`buildah`** to build container images, and **`skopeo`** to manage container images on registries.
- Both **`podman`** and **`buildah`** support the use of Dockerfiles.
- **`Skopeo`** supports public and private registries, the Atomic registry, and local directories, which can use or not the OCI-layout.
- RHEL 8 includes as a technology preview the usage of Pacemaker container bundles with **`podman`**.

Chapter 7

Implementing Enhanced Networking Features

Goal

Describe the major enhancements in network packet processing and network device management.

Objectives

- Explain the new nftables firewall back-end design, advantages and configuration.
- Explain how the NetworkManager service has become an integral and mandatory component in modern network management, able to configure complex, layered network interfaces and components.

Sections

- Managing Server Firewalls in RHEL 8 (and Guided Exercise)
- Configuring Server Networking with NetworkManager (and Guided Exercise)

Lab

Implementing Enhanced Networking Features

Managing Server Firewalls in RHEL 8

Objectives

After completing this section, students should be able to explain the new nftables firewall back-end design, advantages, and configuration.

Introducing Nftables

Firewalld, the firewall management tool in Red Hat Enterprise Linux, uses nftables as its new default firewall back end. In Red Hat Enterprise Linux 8, nftables replaces iptables, which is now deprecated.

The **`nft`** command replaces the iptables commands **`iptables`**, **`ip6tables`**, **`arptables`**, and **`ebtables`**, as a unified, consistent, and simpler command. Additionally, nftables is more efficient and can perform multiple actions in a single rule.

Introducing Nftables as the New Back End for Firewalld

- Firewalld uses nftables as its back end.
- The **`nft`** command replaces the **`iptables`**, **`ip6tables`**, **`arptables`**, and **`ebtables`** commands.
- Firewalld is the recommended way to manage the firewall, over the low-level **`nft`** command.
- The iptables commands are links to the **`xtables-nft-multi`** command, which accepts iptables syntax but creates nftables rules instead.

Red Hat recommends using **`firewalld`** when managing your firewall. Even though **`firewalld`** now uses the nftables back end, it behaves the same as in the previous releases. Its syntax is the same, even for the direct and rich rules. In addition to the **`firewall-cmd`** and **`firewall-config`** commands, you can also manage **`firewalld`** with the web console.

Comparing Nftables with Iptables

- Issues with iptables regarding performance, code maintenance, ease of use, and scalability are solved by nftables.
- Although you should use **`firewalld`** to inspect your firewall rules, you can view all firewall information with a single underlying tool: **`nft`**.
- You only need a single rule for both IPv4 and IPv6 instead of duplicating rules with **`iptables`** and **`ip6tables`**.
- You can add multiple actions per rule. For example, you can log and deny in the same rule.

Create nftables tables, Chains, and Rules using the nft Command Line Tool

Red Hat recommends using Firewalld for managing the firewall. As an alternative, for complex use cases, you can also work directly with the **`nft`** command-line tool to create nftables tables, chains, and rules.

Nftables Objects

- In nftables, *tables* are the top-level objects logically organizing your firewall configuration.

- Inside tables, *chains* group your firewall *rules*.
- Rules can have multiple conditions and multiple actions.
- Use the **`nft`** command to create and review all those nftables objects.
- For persistence, define your objects in **`/etc/sysconfig/nftables.conf`** and enable the systemd **`nftables`** service.

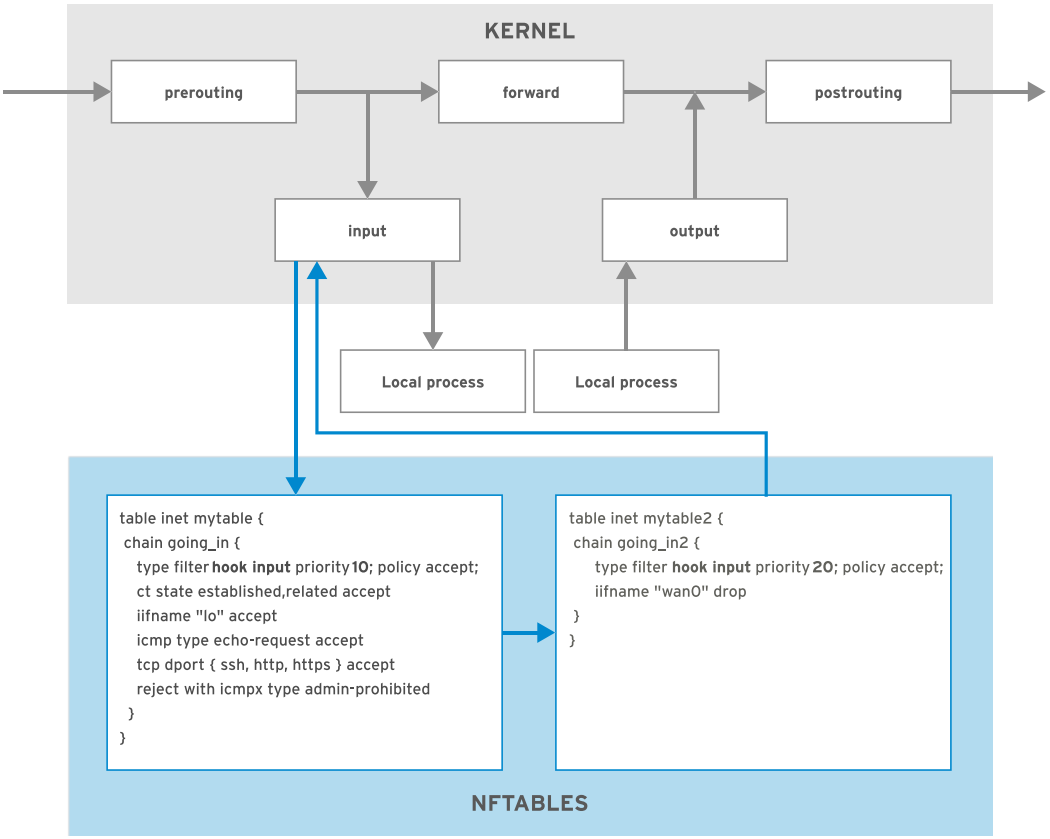


Figure 7.1: Attaching nftables chains to the input packet processing stage in the kernel

Introducing the Table Object

In nftables, tables are top-level objects that you use to group your firewall rules. To list the tables, use the **`nft list tables`** command.

```
[root@demo ~]# nft list tables
table ip filter
table ip6 filter
...output omitted...
table inet firewalld
table ip firewalld
table ip6 firewalld
```

When the **`firewalld`** service starts, it creates tables from the previous output. Notice that **`firewalld`** creates its own tables, named **`firewalld`**, to group its rules. It also creates the other tables, such as **`filter`**, so you can add your own tables with the **`nft`** command.

Each table has an address family, such as **`ip`**, **`ip6`**, or **`inet`**, indicating the type of packets it is processing. **`inet`** is a special address family that combines IPv4 and IPv6. This is useful for rules

that apply to both families. This way, you do not have to write the same rule for IPv4 and again for IPV6.

Create new tables with the **`nft add table family name`** command and delete tables with **`nft delete table family name`**.

```
[root@demo ~]# nft add table inet mytable
```

With nftables, administrators can create as many tables as needed to organize the firewall configuration. On the other hand, iptables has predefined tables that cannot be changed.

Introducing nftables Chains

Inside tables, chains group the firewall rules. Use the **`nft list table family name`** to list the contents of a table.

```
[root@demo ~]# nft list table inet filter
table inet filter {
  chain input {
    type filter hook input priority 0; policy drop;
  }

  chain forward {
    type filter hook forward priority 0; policy accept;
  }

  chain output {
    type filter hook output priority 0; policy accept;
  }
}
```

The previous output shows three chains inside the **`filter`** table. Chains have special attributes:

name
: When you create a chain, you give it a name of your choice. The names in the previous output, **`input`**, **`forward`**, and **`output`**, have been chosen to help you transition from iptables. They have no other meaning.

type
: The type indicates the purpose of the chain. This can be **`filter`** to filter the traffic, **`nat`** to perform network address translation, or **`route`** to mark packets.

hook
: The hook attaches the chain to a packet processing stage in the kernel. The hook value can be **`prerouting`**, **`input`**, **`output`**, **`forward`**, or **`postrouting`**.

priority
: When two or more chains have the same hook value, the priority indicates which chain to process first. The chain with the lowest priority value is processed first.

  The priority is evaluated globally, across all the tables. In the following example, the system evaluates the rules from the **`myinput1`** chain, then from the **`Input_ch`** chain, and finally from the **`myinput2`** chain.

```
[root@demo ~]# nft list table inet mytable1
table inet mytable1 {
  chain myinput1 {
    type filter hook input priority -10; policy accept;
  }
  chain myinput2 {
    type filter hook input priority 100; policy accept;
  }
}
[root@demo ~]# nft list table inet mytable2
table inet mytable2 {
  chain Input_ch {
    type filter hook input priority 50; policy accept;
  }
}
```

policy
The policy establishes the default behavior of the chain when no rule matches. The policy is optional and defaults to **accept**.

You create a new chain with the **nft add chain *family table_name chain_name* { *attributes* }** command and you delete chains with **nft delete chain *family table_name chain_name***.

```
[root@demo ~]# nft add chain inet mytable go_in \
> { type filter hook input priority 10 \; policy drop \;}
```

Remember to protect the **;** character from Bash with ****.

Writing nftables Rules

The chains group your firewall rules. Each rule has two parts: the matches part which gives the conditions the packet must meet, and the statements part to indicate the actions to perform when the conditions match. Use the **nft add rule *family table_name chain_name match... statement...*** command to add a rule to a chain.

```
[root@demo ~]# nft add rule inet mytable go_in ct state established,related accept
[root@demo ~]# nft add rule inet mytable go_in iifname lo accept
[root@demo ~]# nft add rule inet mytable go_in icmp type echo-request accept
[root@demo ~]# nft add rule inet mytable go_in tcp dport {ssh, http, https} accept
[root@demo ~]# nft list table inet mytable
table inet mytable {
  chain go_in {
    type filter hook input priority 10; policy accept;
    ct state established,related accept
    iifname "lo" accept
    icmp type echo-request accept
    tcp dport { ssh, http, https } accept
  }
}
```

Consult the **nft**(8) man page to get the full rule syntax. You can use the **iptables-translate** command to translate the iptables syntax to nft.

```
[root@demo ~]# iptables-translate -A INPUT -p tcp --dport 22 \
> -m conntrack --ctstate NEW -j ACCEPT
nft add rule ip filter INPUT tcp dport 22 ct state new counter accept
```

The **iptables-translate** command shows the corresponding **nft** command but does not run it.

When a packet matches the conditions of a rule, and the statement is **accept**, nftables does not evaluate further the following rules in the chain. However, if there is another chain with the same hook, nftables carries on evaluating the packet against the rules in that second chain. This may result in the packet being dropped if the second chain does not accept it. The following example illustrates that situation.

```
[root@demo ~]# nft list table inet mytable
table inet mytable {
  chain go_in {
    type filter hook input priority 10; policy accept;
    tcp dport 8181 accept
}
[root@demo ~]# nft list table inet mytable2
table inet mytable2 {
  chain go_in2 {
    type filter hook input priority 20; policy drop;
  }
}
```

In this example, a TCP packet with a destination port of 8181 first goes through the **go_in** chain (priority 10) where a rule accepts it. Then, the packet goes through the **go_in2** chain (priority 20) where nftables drops it because of its default policy. Ultimately the packet is dropped even though there is an explicit rule that accepts it in the first chain.

The **nft add rule** command adds the new rule at the end on the chain. The **nft insert rule** command takes the same parameters but inserts the rule at the beginning of the chain. By adding the **handle** option, you can also insert a rule at a specific position. A handle is a unique object identifier. When you create an object, such as a rule, the system automatically gives it a handle.

First, retrieve the handle of the existing rules by adding the **--handle** (**-a**) option to the **nft list table** command.

```
[root@demo ~]# nft list table inet mytable --handle
table inet mytable { # handle 21
  chain go_in { # handle 1
    type filter hook input priority 10; policy accept;
    ct state established,related accept # handle 2
    iifname "lo" accept # handle 3
    icmp type echo-request accept # handle 4
    tcp dport { ssh, http, https } accept # handle 6
  }
}
```

To insert a rule after the rule with handle 4, use the **nft add rule** command as follows:

```
[root@demo ~]# nft add rule inet mytable go_in handle 4 tcp dport ftp log reject
[root@demo ~]# nft list table inet mytable --handle
table inet mytable { # handle 21
  chain go_in { # handle 1
    type filter hook input priority 10; policy accept;
    ct state established,related accept # handle 2
    iifname "lo" accept # handle 3
    icmp type echo-request accept # handle 4
    tcp dport ftp log reject # handle 8
    tcp dport { ssh, http, https } accept # handle 6
  }
}
```

You also use the handle to remove a rule with the **`nft delete rule`** command.

```
[root@demo ~]# nft delete rule inet mytable go_in handle 8
[root@demo ~]# nft list table inet mytable --handle
table inet mytable { # handle 21
  chain go_in { # handle 1
    type filter hook input priority 10; policy accept;
    ct state established,related accept # handle 2
    iifname "lo" accept # handle 3
    icmp type echo-request accept # handle 4
    tcp dport { ssh, http, https } accept # handle 6
  }
}
```

Describing Nftables and Iptables Persistent Configuration

Red Hat recommends using Firewalld for managing your system firewall. As an alternative, and with complex rules, you can disable **`firewalld`** and directly use nftables through the **`nft`** command. Also, for compatibility with previous releases, the iptables commands and the Systemd service are still available but deprecated.

Replacing Firewalld by Nftables or Iptables

- Red Hat recommends using **`firewalld`** for managing your system firewall.
- For complex configurations, you can disable **`firewalld`** and directly use nftables.
- Use the **`iptables-restore-translate`** command to migrate your iptables rules to nftables.
- As an alternative, use the iptables compatibility layer to reuse your legacy iptables rules.

Replacing Firewalld by Nftables Rules

Firewalld rich and direct rules allow for complex firewall configurations. Sometimes, however, you may prefer to disable **`firewalld`** and directly use nftables from the command line.

In preparation for that configuration, stop, disable, and mask **`firewalld`**.

```
[root@demo ~]# systemctl stop firewalld
[root@demo ~]# systemctl disable firewalld
Removed /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@demo ~]# systemctl mask firewalld
Created symlink /etc/systemd/system/firewalld.service -> /dev/null.
[root@demo ~]# nft flush ruleset
```

When the **firewalld** service stops, it does not clear its configuration. The **nft flush ruleset** command removes all the nftables chains and tables.

Enable and start the **nftables** service.

```
[root@demo ~]# systemctl enable --now nftables
Created symlink /etc/systemd/system/multi-user.target.wants/nftables.service -> /
usr/lib/systemd/system/nftables.service.
```

When the service starts, it uses the **nft -f /etc/sysconfig/nftables.conf** command to load the rules from the **/etc/sysconfig/nftables.conf** file. By default, the file only contains comments that you can use as a starting point to develop your rules. You can also generate the file from your current **nftables** configuration by using the **nft list ruleset** command.

```
[root@demo ~]# nft list ruleset > /etc/sysconfig/nftables.conf
```

As an alternative, you can also use a script. For that, make sure to start your file with **#!/usr/sbin/nft -f** for the **nft -f** command to identify it as a script and not as a list of rules in the pseudo JSON format. The **nft -f** command accepts both formats.

```
[root@demo ~]# cat /etc/sysconfig/nftables.conf
#!/usr/sbin/nft -f

flush ruleset

add table inet mytable

add chain inet mytable go_in { type filter hook input priority 10;policy drop;}

add rule inet mytable go_in ct state established,related accept
add rule inet mytable go_in iifname lo accept
add rule inet mytable go_in icmp type echo-request accept
add rule inet mytable go_in tcp dport {ssh, http, https} accept
```

Translating Iptables Rules to Nftables

From a backup of your iptables rules, use the **iptables-restore-translate** command. The command generates nftables commands on its output.

```
[root@demo6 ~]# iptables-save > save.txt
[root@demo6 ~]# cat save.txt
# Generated by iptables-save v1.4.7 on Mon Feb 18 16:25:28 2019
*filter
:INPUT ACCEPT [0:0]
```

```
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [85:14053]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
# Completed on Mon Feb 18 16:25:28 2019
[root@demo6 ~]# scp save.txt root@demo:/root

[root@demo ~]# iptables-restore-translate -f save.txt
# Translated by iptables-restore-translate v1.8.2 on Tue Feb 19 02:57:51 2019
add table ip filter
add chain ip filter INPUT { type filter hook input priority 0; policy accept; }
add chain ip filter FORWARD { type filter hook forward priority 0; policy
 accept; }
add chain ip filter OUTPUT { type filter hook output priority 0; policy accept; }
add rule ip filter INPUT ct state related,established  counter accept
add rule ip filter INPUT ip protocol icmp counter accept
add rule ip filter INPUT iifname "lo" counter accept
add rule ip filter INPUT ct state new  tcp dport 22 counter accept
add rule ip filter INPUT counter reject with icmp type host-prohibited
add rule ip filter FORWARD counter reject with icmp type host-prohibited
# Completed on Tue Feb 19 02:57:51 2019
```

Review the rules and redirect the **`iptables-restore-translate -f`** command output to **`/etc/sysconfig/nftables.conf`** for the **`nftables`** service. Do not forget to add the **`#!/usr/sbin/nft -f`** line at the top of the file.

Replacing Nftables by Iptables

nftables provides a compatibility layer for iptables. The **`iptables`** and **`ip6tables`** commands are still available, but they are links to the nftables **`xtables-nft-multi`** command. That command accepts the iptables syntax but creates nftables objects instead.

In addition, the *iptables-services* package provides the Systemd **`iptables`** service and the associated **`/etc/sysconfig/iptables`** and **`/etc/sysconfig/iptables-config`** configuration files. For this reason, you can reuse your configuration from previous Red Hat Enterprise Linux releases.

The following example shows how to switch your configuration to the now deprecated iptables infrastructure.

```
[root@demo ~]# yum install iptables-services
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
[root@demo ~]# systemctl disable --now firewalld nftables
[root@demo ~]# systemctl mask firewalld nftables
Created symlink /etc/systemd/system/firewalld.service -> /dev/null.
Created symlink /etc/systemd/system/nftables.service -> /dev/null.
[root@demo ~]# systemctl enable --now iptables
```

```
Created symlink /etc/systemd/system/basic.target.wants/iptables.service -> /usr/
lib/systemd/system/iptables.service.
[root@demo ~]# iptables -nL
Chain INPUT (policy ACCEPT)
target     prot opt source       destination
ACCEPT     all  --  0.0.0.0/0    0.0.0.0/0    state RELATED,ESTABLISHED
ACCEPT     icmp --  0.0.0.0/0    0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0    0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0    0.0.0.0/0    state NEW tcp dpt:22
REJECT     all  --  0.0.0.0/0    0.0.0.0/0    reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT)
target     prot opt source       destination
REJECT     all  --  0.0.0.0/0    0.0.0.0/0    reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT)
target     prot opt source       destination
[root@demo ~]# nft list ruleset
table ip filter {
  chain INPUT {
    type filter hook input priority 0; policy accept;
    ct state related,established counter packets 104 bytes 6984 accept
    meta l4proto icmp counter packets 0 bytes 0 accept
    iifname "lo" counter packets 0 bytes 0 accept
    meta l4proto tcp ct state new tcp dport 22 counter packets 0 bytes 0 accept
    counter packets 18 bytes 5634 reject with icmp type host-prohibited
  }

  chain FORWARD {
    type filter hook forward priority 0; policy accept;
    counter packets 0 bytes 0 reject with icmp type host-prohibited
  }

  chain OUTPUT {
    type filter hook output priority 0; policy accept;
  }
}
```

Notice in the previous output that the **`iptables -nL`** command reformats nftables rules. If you insert a rule with **`nft`**, the **`iptables -nL`** command reflects that new rule. In the same way, if you insert a rule with the **`iptables`** command, **`nft`** also reports that rule.

References

The **`nft`**(8) man page.

For more information on firewall management, refer to the *Using and configuring firewalls* chapter in the *Configuring and Managing Networking* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/#assembly_using-firewalls_Configuring-Networking-with-GNOME-GUI

For more information on nftables, refer to the *Firewalld: The Future is nftables* article at https://developers.redhat.com/blog/2018/08/10/firewalld-the-future-is-nftables/

Guided Exercise

Managing Server Firewalls in RHEL 8

In this exercise, you will configure and test firewall rules that use the nftables back end.

Outcomes

In this exercise, you will configure and test firewall rules that use the nftables back end.

1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

2. Nftables basic usage and configuration.

   2.1. List all tables currently active.

```
[root@servera ~]# nft list tables
table ip filter
table ip6 filter
...output omitted...
```

   2.2. List all currently active chains.

```
[root@servera ~]# nft list chains
table ip filter {
  chain INPUT {
    type filter hook input priority 0; policy accept;
  }
  chain FORWARD {
    type filter hook forward priority 0; policy accept;
  }
  chain OUTPUT {
    type filter hook output priority 0; policy accept;
  }
}
table ip6 filter {
  chain INPUT {
    type filter hook input priority 0; policy accept;
  }
  chain FORWARD {
    type filter hook forward priority 0; policy accept;
  }
  chain OUTPUT {
    type filter hook output priority 0; policy accept;
  }
}
...output omitted...
```

2.3. Add a firewall rule allowing inbound HTTP.

```
[root@servera ~]# nft insert rule ip filter INPUT tcp dport http accept
```

2.4. List all chains for the table **`ip filter`** in order to locate the handle for the rule just added.

```
[root@servera ~]# nft list table ip filter -n -a
table ip filter { # handle 1
  chain INPUT { # handle 1
    type filter hook input priority 0; policy accept;
    tcp dport http accept # handle 4
  }
...output omitted...
```

2.5. Remove the rule currently allowing HTTP access using the handle.

```
[root@servera ~]# nft delete rule filter INPUT handle 4
```

2.6. Create a rule enabling access for multiple ports at the same time.

```
[root@servera ~]# nft insert rule ip filter INPUT \
> tcp dport { ssh, http, https, 8181 } accept
```

2.7. Set the **`INPUT`** chain default policy to **`drop`** all traffic not specifically accepted.

```
[root@servera ~]# nft add chain ip filter INPUT \
> { type filter hook input priority 0\; policy drop\; }
```

3. Remove rules added during this exercise.

3.1. Set the **`INPUT`** chain default policy to **`accept`** all traffic by default.

```
[root@servera ~]# nft add chain ip filter INPUT \
> { type filter hook input priority 0\; policy accept\; }
```

3.2. Find the handle and remove the rule currently allowing access for SSH, HTTP, HTTPS, and 8181.

```
[root@servera ~]# nft list table ip filter -n -a
table ip filter { # handle 1
  chain INPUT { # handle 1
    type filter hook input priority 0; policy accept;
    tcp dport { ssh, http, https, 8181 } accept # handle 6
  }
...output omitted...
[root@servera ~]# nft delete rule filter INPUT handle 6
```

4. Log off from **`servera`**.

```
[root@servera ~]# exit
Connection to servera closed.
```

This concludes the guided exercise.

Configuring Server Networking with NetworkManager

Objectives

After completing this section, students should be able to explain how the NetworkManager service has become an integral and mandatory component in modern network management, and to configure complex, layered network interfaces and components.

Introducing NetworkManager

NetworkManager is the preferred network configuration tool in Red Hat Enterprise Linux 8. It can handle complex configurations, and layered products such as OpenStack, OpenShift, and Red Hat Virtualization now rely on it.

Configuring the Network with NetworkManager

- Red Hat recommends using NetworkManager for network configuration.
- NetworkManager can now handle complex configurations, such as Open vSwitch or SR-IOV.
- Layered products, such as OpenStack, are using NetworkManager.
- Multiple front ends are available: the web console, **`nmcli`**, **`nmtui`**, and the Network RHEL System Role.
- The Systemd **`network`** service and the network scripts are no longer available.
- **`ifup`** and **`ifdown`** are links to NetworkManager scripts.

Configuring System Networks with NetworkManager

NetworkManager provides a uniform way to configure the system network across different Red Hat Enterprise Linux releases. Because the Network Red Hat Enterprise Linux System Role relies on NetworkManager, you can often use a single Playbook for multiple RHEL releases.

In RHEL 8, the system does not install the legacy Systemd **`network`** service. However, NetworkManager still relies on the configuration files in the **`/etc/sysconfig/network-scripts/`** directory. You can still edit those files, and use the **`ifup`** command for NetworkManager to reread the interface configuration.

```
[user@demo ~]# vim /etc/sysconfig/network-scripts/ifcfg-ens3
[user@demo ~]# ifup ens3
```

The **`ifup`** and **`ifdown`** commands are now links to scripts that call the **`nmcli`** command to reload the configuration from the **`ifcfg-*`** file and start or stop the connection.

References

NetworkManager(8), **`nmcli`**(1), and **`nmtui`**(1) man pages

For more information, refer to the *Configuring and managing networking* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

Guided Exercise

Configuring Server Networking with NetworkManager

In this exercise, you will manage network interfaces and devices using NetworkManager.

Outcomes

You should be able to:

- Review the network configurations from a system with **nmcli**.
- Configure Ethernet network interfaces with **nmcli**.
- Configure bonded network interfaces with **nmcli**.

Before You Begin

Log in to **workstation** as **student** using **student** as the password. On **workstation**, run the **lab net-manager start** command to verify that the environment is ready and prepare the system for the exercise.

```
[student@workstation ~]$ lab net-manager start
```

1. Log in to **serverb** as the **root** user.

```
[student@workstation ~]$ ssh root@serverb
```

2. Explore the **nmcli** tool.

   2.1. The **nmcli** tool is included in the NetworkManager package and is installed by default in Red Hat Enterprise Linux 8.

```
[root@serverb ~]# nmcli --help
...output omitted...
OBJECT
  g[eneral]       NetworkManager's general status and operations
  n[etworking]    overall networking control
  r[adio]         NetworkManager radio switches
  c[onnection]    NetworkManager's connections
  d[evice]        devices managed by NetworkManager
  a[gent]         NetworkManager secret agent or polkit agent
  m[onitor]       monitor NetworkManager changes
```

   2.2. Explore the general network status on **serverb**.

```
[root@serverb ~]# nmcli general status
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

2.3. Verify network connectivity on **serverb**.

```
[root@serverb ~]# nmcli networking connectivity check
full
```

2.4. Explore the wireless options with **nmcli radio**.

```
[root@serverb ~]# nmcli radio help
Usage: nmcli radio { COMMAND | help }

COMMAND := { all | wifi | wwan }

  all | wifi | wwan [ on | off ]
```

2.5. Show network device status on **serverb**.

```
[root@serverb ~]# nmcli device status
ens3    ethernet  connected     Wired connection 1
ens4    ethernet  disconnected  --
ens5    ethernet  disconnected  --
ens6    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

2.6. Show network connection status on **serverb**.

```
[root@serverb ~]# nmcli connection
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  d27327cd-9e6e-45dd-b71f-4c0b99e62742  ethernet  ens3
```

2.7. Rename a network connection on **serverb**.

```
[root@serverb ~]# nmcli connection modify "Wired connection 1" \
> connection.id ens3
[root@serverb ~]# nmcli connection
NAME  UUID                                  TYPE      DEVICE
ens3  d27327cd-9e6e-45dd-b71f-4c0b99e62742  ethernet  ens3
```

3. Configure a network interface with the **nmcli** tool on **serverb**.

Parameter	Value
Interface	**ens4**
IPv4 Address	172.25.250.20/24

3.1. Create an Ethernet network connection on **serverb** using the **ens4** interface and name it **ens4**.

```
[root@serverb ~]# nmcli connection add type ethernet ifname ens4 \
> connection.id ens4
Connection 'ens4' (99671499-dacd-4f87-8c09-25d9e0b7301e) successfully added.
```

3.2. Assign an IPv4 address to the **ens4** network connection on **serverb**.

```
[root@serverb ~]# nmcli connection modify ens4 ipv4.addresses 172.25.250.20/24
```

3.3. Enable manual IPv4 address assignation on interface **ens4** on **serverb**.

```
[root@serverb ~]# nmcli connection modify ens4 ipv4.method manual
```

3.4. Activate the **ens4** network connection on **serverb**.

```
[root@serverb ~]# nmcli connection up ens4
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/124)
```

3.5. Verify the system interface status on **serverb**.

```
[root@serverb ~]# ip address
...output omitted...
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 default qlen 1000
    link/ether 2c:c2:60:23:34:b3 brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.20/24 brd 172.25.250.255 scope global noprefixroute ens4
...output omitted...
```

3.6. Show network device status on **serverb**.

```
[root@serverb ~]# nmcli device status
DEVICE  TYPE      STATE         CONNECTION
ens3    ethernet  connected     ens3
ens4    ethernet  connected     ens4
ens5    ethernet  disconnected  --
ens6    ethernet  disconnected  --
lo      loopback  unmanaged     --
```

4. Configure a bonded interface with **nmcli** tool on **serverb**.

Parameter	Value
Interface Name	**bond0**
Interface Members	**ens5** and **ens6**
Mode	Active Backup
IPv4 Address	172.25.250.21/24

4.1. Create an **Active-Backup** bonded interface, name it **bond0** with **ens5** and **ens6** as members on **serverb**.

```
[root@serverb ~]# nmcli connection add type bond \
> ifname bond0 connection.id bond0 \
> ipv4.method disabled ipv6.method ignore \
> mode active-backup
Connection 'bond0' (bf741496-bd5f-4573-a453-16dae9743bdb) successfully added.
```

4.2. Assign the network interfaces **ens5** and **ens6** as members of the bonded interface **bond0** on **serverb**.

```
[root@serverb ~]# nmcli connection add type ethernet \
> ifname ens5 connection.id ens5 \
> ipv4.method disabled ipv6.method ignore \
> master bond0
Connection 'ens5' (17b957ca-a691-433d-b632-5048944e3646) successfully added.
[root@serverb ~]# nmcli connection add type ethernet \
> ifname ens6 connection.id ens6 \
> ipv4.method disabled ipv6.method ignore \
> master bond0
Connection 'ens6' (c06fc02d-3cc0-4b45-babd-b4e1895f631f) successfully added.
```

4.3. Enable manual IPv4 address assignment, and set an IPv4 address to the bonded interface **bond0** on **serverb**.

```
[root@serverb ~]# nmcli connection modify bond0 \
> ipv4.method manual ipv4.addresses 172.25.250.21/24
```

4.4. Activate the member interfaces **ens5**, **ens6** an the bonded interface **bond0** network connections on **serverb**.

```
[root@serverb ~]# nmcli connection up ens5
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/128)
[root@serverb ~]# nmcli connection up ens6
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/129)
[root@serverb ~]# nmcli connection up bond0
Connection successfully activated (master waiting for slaves) (D-Bus active
 path: /org/freedesktop/NetworkManager/ActiveConnection/130)
```

4.5. Show the network devices status on **serverb**.

```
[root@serverb ~]# nmcli device status
DEVICE  TYPE      STATE      CONNECTION
ens3    ethernet  connected  ens3
bond0   bond      connected  bond0
ens4    ethernet  connected  ens4
ens5    ethernet  connected  ens5
ens6    ethernet  connected  ens6
lo      loopback  unmanaged  --
```

4.6. Verify the system interfaces status on **serverb**.

```
[root@serverb ~]# ip address
...output omitted...
3: ens5: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bond0 state UP group default qlen 1000
    link/ether 2c:c2:60:33:e6:79 brd ff:ff:ff:ff:ff:ff
4: ens6: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bond0 state UP group default qlen 1000
    link/ether 2c:c2:60:33:e6:79 brd ff:ff:ff:ff:ff:ff
8: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default qlen 1000
    link/ether 2c:c2:60:33:e6:79 brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.21/24 brd 172.25.250.255 scope global noprefixroute bond0
...output omitted...
```

5. Verify the network configurations on **serverb**.

5.1. Explore the new bonded interface **bond0** properties on **serverb**.

```
[root@serverb ~]# nmcli device show bond0
GENERAL.DEVICE:                         bond0
GENERAL.TYPE:                           bond
GENERAL.HWADDR:                         2C:C2:60:33:E6:79
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     bond0
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/
ActiveConnection/459
IP4.ADDRESS[1]:                         172.25.250.21/24
...output omitted...
```

5.2. Explore the new Ethernet interface **ens4** properties on **serverb**.

```
[root@serverb ~]# nmcli device show ens4
GENERAL.DEVICE:                         ens4
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         2C:C2:60:23:34:B3
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     ens4
```

```
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/
ActiveConnection/124
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         172.25.250.20/24
...output omitted...
```

5.3. Log off from **serverb**.

```
[root@serverb ~]# exit
Connection to serverb closed.
```

Finish

From **workstation**, run **lab net-manager finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab net-manager finish
```

This concludes the guided exercise.

Lab

Implementing Enhanced Networking Features

In this lab, you will configure network interfaces and devices using the web console.

Outcomes

You should be able to:

- Review the network configurations from a system using the web console and **nmcli**.
- Configure Ethernet network interfaces using the web console.
- Configure bonded network interfaces using the web console.

Before You Begin

Log in to **workstation** as **student** using **student** as the password. On **workstation**, run the **lab net-review start** command to verify that the environment is ready and prepare the system for the lab.

```
[student@workstation ~]$ lab net-review start
```

1. The web console is already installed on the system, but it is not activated. Enable and start the web console service on **serverb**.
2. On **workstation** open Firefox and log in to the web console running on **serverb** system as the **student** user with **student** as password.
3. Configure and enable an Ethernet interface on **serverb**, using the web console.

Parameter	Value
Interface	**ens4**
Address	172.25.250.20/24

4. Configure a Bonded interface, on **serverb**, using the web console.

Parameter	Value
Interface Name	**bond0**
Interface Members	**ens5** and **ens6**
Mode	Active Backup
Address	172.25.250.21/24

Important

When creating Bonded/Teaming Interfaces through web console, you can potentially get those interfaces in an unmanaged state. That is caused by **dhclient** exhausted its timeout if there are not DHCP Server available, and those interfaces are activated by default with DHCP as Addressing method. You can get the interface back to a managed state with the **nmcli** tool, and try again to configure the IP settings. For example, for a bonded interface named **bond0**:

```
nmcli device set bond0 managed yes
```

5. Review the changes with the **nmcli** tool, and system tools like **ip**, on **serverb**.

Finish

From **workstation**, run **lab net-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab net-review finish
```

This concludes the guided exercise.

Solution

Implementing Enhanced Networking Features

In this lab, you will configure network interfaces and devices using the web console.

Outcomes

You should be able to:

- Review the network configurations from a system using the web console and **nmcli**.
- Configure Ethernet network interfaces using the web console.
- Configure bonded network interfaces using the web console.

Before You Begin

Log in to **workstation** as **student** using **student** as the password. On **workstation**, run the **lab net-review start** command to verify that the environment is ready and prepare the system for the lab.

```
[student@workstation ~]$ lab net-review start
```

1. The web console is already installed on the system, but it is not activated. Enable and start the web console service on **serverb**.

   1.1. Log in to **serverb** as the **root** user.

```
[student@workstation ~]$ ssh root@serverb
```

   1.2. Use the **systemctl enable cockpit.socket** command to enable the web console service.

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

   1.3. Use the **systemctl start cockpit** command to start the web console service.

```
[root@serverb ~]# systemctl start cockpit
```

   1.4. Log off from **serverb**.

```
[root@serverb ~]# exit
Connection to serverb closed.
```

2. On **workstation** open Firefox and log in to the web console running on **serverb** system as the **student** user with **student** as password.

   2.1. Open Firefox and go to the **https://serverb.lab.example.com:9090** address.

   2.2. Accept the self-signed certificate by adding it as an exception.

   2.3. Log in as **student** user with **student** as password, mark the **Reuse my password for privileged tasks** check box.

   You are now logged in as a **student** user, with **Privileged** access.

3. Configure and enable an Ethernet interface on **serverb**, using the web console.

Parameter	Value
Interface	**ens4**
Address	172.25.250.20/24

   3.1. Click **Networking** in the left menu bar.

   3.2. Click on the **ens4** interface, to edit IPv4 details.

   3.3. Click on **Automatic (DHCP)** link.

   3.4. In the new **IPv4 Settings** window, for the Addresses choose **Manual**.

   3.5. In the **Address** text box, type in **172.25.250.20** as IP Address.

   3.6. In the **Prefix length or Netmask** text box, type in the **255.255.255.0** netmask value.

   3.7. Click the **Apply** button to save the new network configuration.

   Notice that the new configuration is immediately applied, and the new IP Address is visible in the **IPv4** line.

   3.8. Enable the network interface by toggling **ON/OFF** to **ON**.

4. Configure a Bonded interface, on **serverb**, using the web console.

Parameter	Value
Interface Name	**bond0**
Interface Members	**ens5** and **ens6**
Mode	Active Backup
Address	172.25.250.21/24

Important

When creating Bonded/Teaming Interfaces through web console, you can potentially get those interfaces in an unmanaged state. That is caused by **`dhclient`** exhausted its timeout if there are not DHCP Server available, and those interfaces are activated by default with DHCP as Addressing method. You can get the interface back to a managed state with the **`nmcli`** tool, and try again to configure the IP settings. For example, for a bonded interface named **`bond0`**:

```
nmcli device set bond0 managed yes
```

4.1. Click **Networking** in the left navigation bar.

4.2. Click the **Add Bond** button.

4.3. In the new **Bond Settings** window, use **`bond0`** as Name, mark the **`ens5`** and **`ens6`** interfaces as Members, and ensure the selected Mode is **`Active-Backup`**. Leave the other settings options as default.

4.4. Click the **Apply** button to save the new bonded interface.

4.5. Click on the **`bond0`** interface, to edit IPv4 details.

4.6. Click on **Automatic (DHCP)** link.

4.7. In the new **IPv4 Settings** window, for the Addresses choose **Manual**.

4.8. In the **Address** text box, type in **`172.25.250.21`** as IP Address.

4.9. In the **Prefix length or Netmask** text box, type in the **`255.255.255.0`** netmask value.

4.10. Click the **Apply** button to save the new network configuration.

Notice that the new configuration is immediately applied, and the new IP Address is visible in the **IPv4** line.

5. Review the changes with the **`nmcli`** tool, and system tools like **`ip`**, on **`serverb`**.

5.1. In the web console, click on **Terminal** in the left navigation bar to access the terminal.

A terminal session opens with the **`student`** user already logged in.

5.2. Show the network connections status on **`serverb`**.

```
[student@serverb ~]$ nmcli connection
NAME   UUID                                  TYPE      DEVICE
bond0  491a07c3-37c4-4079-98cb-bc1931a0b1b8  bond      bond0
ens3   d27327cd-9e6e-45dd-b71f-4c0b99e62742  ethernet  ens3
ens4   7b941fb1-3f77-4a5c-9c73-bbbf9f26b8fb  ethernet  ens4
ens5   554a2f71-db4b-44db-b70a-a23e70dbba69  ethernet  ens5
ens6   3650e4c9-799c-4243-b0ad-c6006f74f453  ethernet  ens6
```

5.3. Show the network devices status on **`serverb`**.

```
[student@serverb ~]$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
ens3    ethernet  connected  ens3
bond0   bond      connected  bond0
ens4    ethernet  connected  ens4
ens5    ethernet  connected  ens5
ens6    ethernet  connected  ens6
lo      loopback  unmanaged  --
```

5.4. Verify the system interfaces status on **serverb**.

```
[student@serverb ~]$ ip address
...output omitted...
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
 default qlen 1000
    link/ether 2c:c2:60:27:fb:cf brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.20/24 brd 172.25.250.255 scope global noprefixroute ens4
...output omitted...
4: ens5: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bond0 state UP group default qlen 1000
    link/ether 2c:c2:60:22:6b:41 brd ff:ff:ff:ff:ff:ff
5: ens6: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bond0 state UP group default qlen 1000
    link/ether 2c:c2:60:22:6b:41 brd ff:ff:ff:ff:ff:ff
15: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state
 UP group default qlen 1000
    link/ether 2c:c2:60:22:6b:41 brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.21/24 brd 172.25.250.255 scope global noprefixroute bond0
...output omitted...
```

5.5. Log out of the web console on **serverb**.

Finish

From **workstation**, run **lab net-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab net-review finish
```

This concludes the guided exercise.

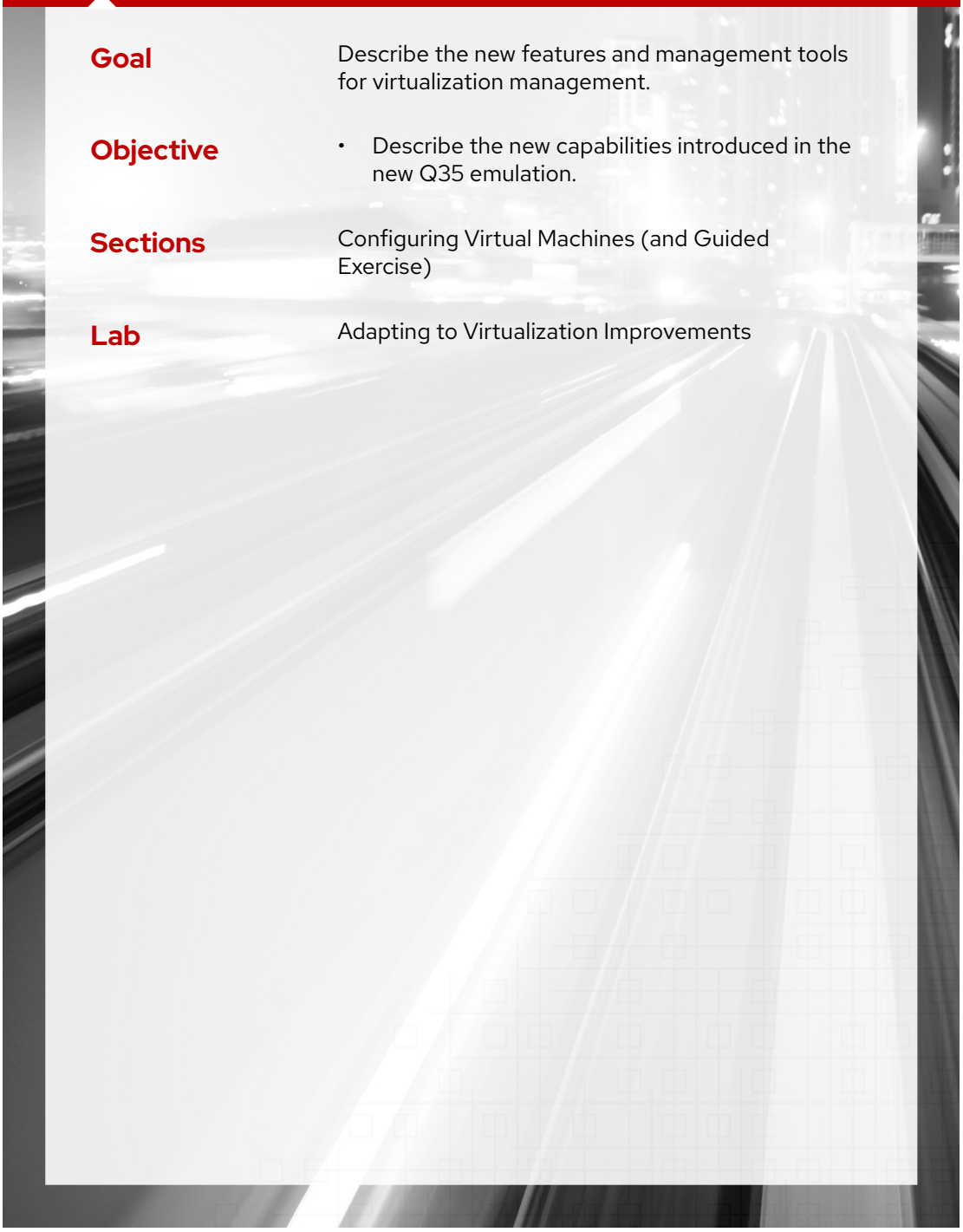
Summary

In this chapter, you learned:

- Firewalld uses nftables as its back end instead of iptables.
- The **`nft`** command replaces **`iptables`**, **`ip6tables`**, **`arptables`**, and **`ebtables`**.
- The **`iptables-restore-translate`** command translates iptables rules into nftables rules.
- Red Hat recommends using NetworkManager for network configuration.
- NetworkManager can now handle complex configurations.
- OpenStack and other layered products use NetworkManager.

Chapter 8

Adapting to Virtualization Improvements



Goal	Describe the new features and management tools for virtualization management.
Objective	• Describe the new capabilities introduced in the new Q35 emulation.
Sections	Configuring Virtual Machines (and Guided Exercise)
Lab	Adapting to Virtualization Improvements

Configuring Virtual Machines

Objectives

After completing this section, students should be able to describe the new capabilities introduced in the new Q35 emulation.

Updated QEMU Emulation

The *Kernel-based Virtual Machine (KVM)* kernel module and the *QEMU* emulator are the basis of virtualization in Red Hat Enterprise Linux 8. In this release, QEMU can now emulate the Intel Q35 motherboard chipset, which offers a better hardware platform for modern virtualized operating systems.

New Q35 Virtual Machine Type

- In addition to the previous Intel 440FX machine type, QEMU now emulates the Intel Q35 chipset and features.
- Q35 chipset emulation provides more current hardware devices than previously available.
- Q35 provides a PCI Express bus and supports secure boot.
- Q35 supports PCI Express pass-through and simplifies *physical to virtual (p2v)* migrations.

QEMU now emulates the Intel Q35 chipset and its associated Intel ICH9 (I/O controller hub) chipset.

The previous chipset provided only PCI ports (which are difficult to map to physical PCI-E ports), but the ICH9 chipset provides a *PCI Express (PCI-E)* bus. This supports PCI-E pass-through by directly mapping a PCI-E port inside the virtual machine to a PCI-E port on the host system.

Select the chipset to emulate when creating the virtual machine. If using the web console, **`virt-install`**, or **`virt-manager`**, the tool automatically chooses the Q35 chipset when a recent, supported operating system is selected.

Note

The web console replaces **`virt-manager`**, which is deprecated. Red Hat recommends the web console for virtual machine management. In a subsequent release, **`virt-manager`** will be removed.

For RHEL 7, RHEL 8, Microsoft Windows 2016, and Microsoft Windows 10 virtual machines, QEMU emulates the Q35 chipset. For RHEL 6 virtual machines, QEMU emulates the Intel 440FX chipset. The following screen capture shows the selection of the operating system in the web console.

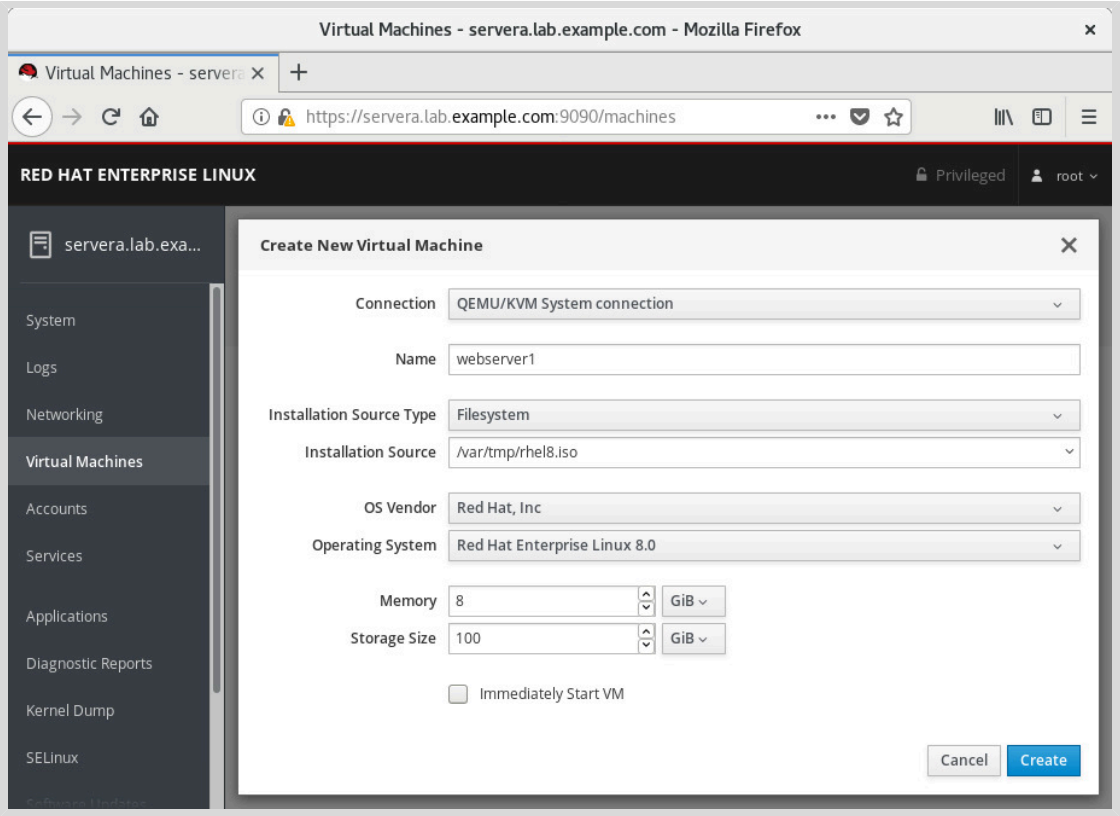


Figure 8.1: Operating system selection in the web console

When you import a virtual machine from a system with an older version of QEMU, you may have to force the previous Intel 440FX chipset emulation. This is because the virtualized operating system already uses drivers for that chipset. If you choose the Q35 chipset emulation, which presents a new hardware model, the virtualized operating system may not have the appropriate drivers installed to use the emulated hardware.

You get the chipset emulation that a virtual machine uses in its XML file definition, in the **os** section. The following example shows a virtual machine using the new Q35 chipset emulation.

```
[user@demo ~]$ virsh dumpxml webserver1
<domain type='kvm' id='5'>
  <name>webserver1</name>
  <uuid>f6fbe32f-ef6c-4d8a-8db0-96551c82444b</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/
domain/1.0">
      <libosinfo:os id="http://redhat.com/rhel/8.0"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>8388608</memory>
  <currentMemory unit='KiB'>8388608</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-q35-rhel7.6.0'>hvm</type>
```

```
    <boot dev='cdrom'/>
    <boot dev='hd'/>
  </os>
...output omitted...
```

The following example shows a virtual machine using the old Intel 440FX chipset emulation.

```
[user@demo ~]$ virsh dumpxml myrhel6
<domain type='kvm' id='7'>
  <name>myrhel6</name>
  <uuid>51ffc808-d3d0-4220-adc9-32adea33d218</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/
domain/1.0">
      <libosinfo:os id="http://redhat.com/rhel/5.0"/>
    </libosinfo:libosinfo>
  </metadata>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-rhel7.6.0'>hvm</type>
    <boot dev='cdrom'/>
    <boot dev='hd'/>
  </os>
...output omitted...
```

List the supported chipset emulations with the **`virsh capabilities`** command.

```
[user@demo ~]# virsh capabilities
  <guest>
    <os_type>hvm</os_type>
    <arch name='x86_64'>
      <wordsize>64</wordsize>
      <emulator>/usr/libexec/qemu-kvm</emulator>
      <machine maxCpus='240'>pc-i440fx-rhel7.6.0</machine>
      <machine canonical='pc-i440fx-rhel7.6.0' maxCpus='240'>pc</machine>
      <machine maxCpus='240'>pc-i440fx-rhel7.0.0</machine>
      <machine maxCpus='384'>pc-q35-rhel7.6.0</machine>
      <machine canonical='pc-q35-rhel7.6.0' maxCpus='384'>q35</machine>
...output omitted...
```

The previous output from a RHEL 8 system shows that QEMU supports both the Q35 and the Intel 440FX chipset emulation. The **pc** machine type is an alias for the i440FX emulation, and the **q35** machine type is an alias for the Q35 emulation.

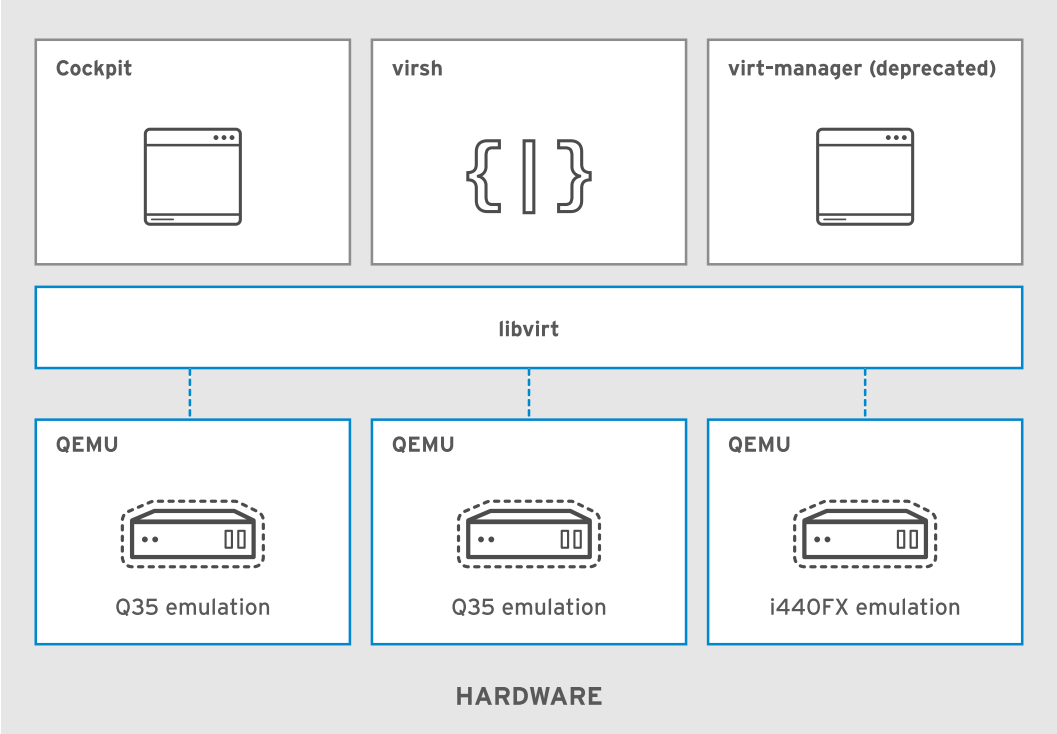


Figure 8.2: Virtualization overview

Virtual Machine Management in the Web Console

- The web console in RHEL 8 offers basic functionality for administering virtual machines. Install the *cockpit-machines* package to enable the web console component for managing virtual machines.
- For more advanced configurations, use the Libvirt tools such as **`virsh`** or **`virt-install`**.
- The **`virt-manager`** graphical interface is still available but deprecated, and will be removed in a subsequent release.

RHEL Virtualization Available as a Module

- For convenience, you can install the virtualization software using the new **`yum module`** command.
- The *virt* Yum module has one stream called **`rhel`** and a single default profile.
- A separate *virt* stream will be shipped in another repository for layered products such as Red Hat Virtualization (RHV), which are able to receive major updates on a different cadence than RHEL major releases usually allow.
- You can also install the virtualization packages individually using traditional **`yum`** commands.

The *virt* Yum module has one stream called **`rhel`** and a single default profile.

```
[user@demo ~]$ yum module list
Name          Stream          Profiles          Summary
virt          rhel [d][e]     common [d]        Virtualization module
...output omitted...
```

SCSI-3 Persistent Reservations with Virtio-SCSI

Supporting SCSI-3 Persistent Reservations with Virtio-SCSI

- On Red Hat Enterprise Linux 8, both QEMU and Libvirt support SCSI-3 persistent reservations on storage-devices presented to VMs through Virtio-SCSI backed by direct-attached LUNs.
- Virtual machines can share Virtio-SCSI storage devices and use SCSI-3 PRs to control access.
- Storage devices managed with **`device-mapper-multipath`** can be passed through to VMs to use SCSI-3 PRs, and the host manages PR actions across all paths.

References

For more information on virtualization, refer to the *Configuring and managing virtualization* guide at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_virtualization/

For more information on SCSI-3 Persistent Reservations, refer to the *How can I view, create, and remove SCSI persistent reservations and keys?* article at
https://access.redhat.com/solutions/43402

Guided Exercise

Configuring Virtual Machines

In this exercise, you will deploy a virtual machine using the web console, and configure features made available by the new Q35 emulation.

Outcomes

You should be able to:

- Install the required packages to enable virtualization
- Configure the web console for the management of virtual machines
- Create new virtual machines with the web console

Before You Begin

Log in to **workstation** as **student** using **student** as the password. On **workstation**, run the **lab virt-config start** command to verify that the environment is ready and prepare the systems for the exercise.

```
[student@workstation ~]$ lab virt-config start
```

Note

In case **workstation** hangs using the web console, restart the virtual machine, and log into the web console with a new browser.

1. Prepare **servera** for hosting virtual machines.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. Use **yum** to install the *virt* module.

```
[root@servera ~]# yum module install virt
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

   1.3. Confirm that the **servera** system supports virtualization.

```
[root@servera ~]# virt-host-validate
  QEMU: Checking for hardware virtualization                 : PASS
  QEMU: Checking if device /dev/kvm exists                   : PASS
```

```
  QEMU: Checking if device /dev/kvm is accessible             : PASS
  QEMU: Checking if device /dev/vhost-net exists              : PASS
  QEMU: Checking if device /dev/net/tun exists                : PASS
  QEMU: Checking for cgroup 'memory' controller support       : PASS
  QEMU: Checking for cgroup 'memory' controller mount-point   : PASS
  QEMU: Checking for cgroup 'cpu' controller support          : PASS
  QEMU: Checking for cgroup 'cpu' controller mount-point      : PASS
  QEMU: Checking for cgroup 'cpuacct' controller support      : PASS
  QEMU: Checking for cgroup 'cpuacct' controller mount-point  : PASS
  QEMU: Checking for cgroup 'cpuset' controller support       : PASS
  QEMU: Checking for cgroup 'cpuset' controller mount-point   : PASS
  QEMU: Checking for cgroup 'devices' controller support      : PASS
  QEMU: Checking for cgroup 'devices' controller mount-point  : PASS
  QEMU: Checking for cgroup 'blkio' controller support        : PASS
  QEMU: Checking for cgroup 'blkio' controller mount-point    : PASS
  QEMU: Checking for device assignment IOMMU support          : WARN (No ACPI
 DMAR table found, IOMMU either disabled in BIOS or not supported by this hardware
 platform)
```

You can safely ignore the warning message because **servera** is itself a virtual machine and its hardware emulation does not provide IOMMU support.

PCI pass-through, which allows you to attach a hardware device from the host system to your virtual machine, requires IOMMU.

2. Configure the web console on **servera** for managing virtual machines.

2.1. Confirm that the web console is running and enabled on **servera**.

```
[root@servera ~]# systemctl is-active cockpit.socket
active
[root@servera ~]# systemctl is-enabled cockpit.socket
enabled
```

If that is not the case, start and enable the web console.

```
[root@servera ~]# systemctl enable --now cockpit.socket
```

2.2. On **workstation**, open Firefox and navigate to https://servera.lab.example.com:9090/. If a certificate error appears, accept the self-signed certificate.

Select **Reuse my password for privileged tasks** and sign in using **student** as user name and **student** as the password.

Click on **servera.lab.example.com** in the left navigation bar and notice that the menu on the left does not include an entry for managing virtual machines.

2.3. On **servera**, install the *cockpit-machines* package.

```
[root@servera ~]# yum install cockpit-machines
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

2.4. In Firefox, refresh the web console's page. Notice the new **Virtual Machines** menu entry on the left.

3. Use the web console to deploy an RHEL 8 virtual machine on **`servera`** according to the following requirements.

Parameter	Value
VM name	**`myrhel8`**
RHEL 8 DVD ISO file	**`/var/tmp/boot.iso`** on **`servera`**
Operating system	Red Hat Enterprise Linux 8.0
Memory	1 GiB
Storage size	10 GiB
Kickstart URL	`http://172.25.250.254/min.ks`

3.1. In Firefox, click the **Virtual Machines** menu entry on the left of the web console. If you have not started the **`libvirtd`** Systemd service from the command line, then the web console offers to perform this step for you after the Yum *virt* module has been installed. Click **Start libvirt** to start and enable the **`libvirtd`** Systemd service.

3.2. Click **Create VM** and fill in the form according to the following table.

Parameter	Value
Name	**`myrhel8`**
Installation Source Type	**Filesystem**
Installation Source	**`/var/tmp/boot.iso`**
OS Vendor	Red Hat, Inc
Operating System	Red Hat Enterprise Linux 8.0
Memory	1 GiB
Storage Size	10 GiB

Click **Create**. If the state of the new virtual machine is **in transition**, refresh the web console's page. Click the **`myrhel8`** virtual machine and click **Install** to start the installation.

3.3. Use the kickstart at `http://172.25.250.254/min.ks` to perform an unattended installation.

In the console, use the **up** arrow key to highlight **Install Red Hat Enterprise Linux 8.0**. Press the **`Tab`** key. At the end of the line, at the bottom of the console, add **`ks=http://172.25.250.254/min.ks`** and press **`Enter`**.

Wait for the installation to complete. Because the virtual machine powers off after installation, click **Run** to start it.

4. Inspect the new virtual machine details.

4.1. Using the web console, in the console of the new virtual machine, log in as **root**, with **redhat** for the password.

Run the **dmidecode** command to confirm that QEMU uses the new Q35 chipset emulation.

```
[root@localhost ~]# dmidecode
...output omitted...
Handle 0x0100, DMI type 1, 27 bytes
System Information
 Manufacturer: Red Hat
 Product Name: KVM
 Version: RHEL-7.6.0 PC (Q35 + ICH9, 2009)
 Serial Number: Not Specified
 UUID: c0dad148-a985-4f74-a894-fc4cf2468661
 Wake-up Type: Power Switch
 SKU Number: Not Specified
 Family: Red Hat Enterprise Linux
...output omitted...
```

4.2. Run the **lspci** command to inspect further the emulated hardware.

```
[root@localhost ~]# lspci
00:00.0 Host bridge: Intel Corporation 82G33/G31/P35/P31 Express DRAM Controller
00:01.0 VGA compatible controller: Red Hat, Inc. QXL paravirtual graphic card (rev
 04)
00:02.0 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.1 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.2 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.3 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.4 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.5 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:02.6 PCI bridge: Red Hat, Inc. QEMU PCIe Root port
00:1b.0 Audio device: Intel Corporation 82801I (ICH9 Family) HD Audio Controller
 (rev 03)
00:1f.0 ISA bridge: Intel Corporation 82801IB (ICH9) LPC Interface Controller (rev
 02)
00:1f.2 SATA controller: Intel Corporation 82801IR/IO/IH (ICH9R/DO/DH) 6 port SATA
 Controller [AHCI mode] (rev 02)
00:1f.3 SMBus: Intel Corporation 82801I (ICH9 Family) SMBus Controller (rev 02)
01:00.0 Ethernet controller: Red Hat, Inc. Virtio network device (rev 01)
02:00.0 USB controller: Red Hat, Inc. QEMU XHCI Host Controller (rev 01)
03:00.0 Communication controller: Red Hat, Inc. Virtio console (rev 01)
04:00.0 SCSI storage controller: Red Hat, Inc. Virtio block device (rev 01)
05:00.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon (rev 01)
06:00.0 Unclassified device [00ff]: Red Hat, Inc. Virtio RNG (rev 01)
```

Notice that the Q35 chipset emulation provides PCI Express ports and a SATA controller, but no ISA bus.

4.3. Run the **systemctl status qemu-guest-agent** command to confirm that the installation process automatically deploys the *qemu-guest-agent* package and starts the service.

```
[root@localhost ~]# systemctl status qemu-guest-agent
* qemu-guest-agent.service - QEMU Guest Agent
   Loaded: loaded (/usr/lib/systemd/system/qemu-guest-agent.service; disabled;
 vendor preset: enabled)
   Active: active (running) since Thu 2019-02-07 03:41:41 EST; 1h 8min ago
 Main PID: 624 (qemu-ga)
    Tasks: 1 (limit: 6104)
   Memory: 908.0K
   CGroup: /system.slice/qemu-guest-agent.service
           └─624 /usr/bin/qemu-ga --method=virtio-serial --path=/dev/virtio-ports/
org.qemu.guest_agent.0 --blacklist=guest-file-open,guest-file-close,guest-f>

Feb 07 03:41:41 localhost.localdomain systemd[1]: Started QEMU Guest Agent.
```

The QEMU guest agent allows the host system to issue commands to the operating system running inside the virtual machine. For example, virtualization platforms use this feature to freeze the file systems inside the virtual machine when taking a snapshot.

4.4. In the web console, explore the **Overview**, **Disks**, and **Networks** tabs for the new virtual machine.

In the **Networks** tab, notice that you can only plug or unplug existing network devices, but you cannot create new devices. In those situations, use the **`virsh`** command or the deprecated **`virt-manager`** graphical tool.

4.5. On **`servera`**, use the **`virsh`** command to confirm that QEMU uses the Q35 chipset emulation for your new virtual machine.

```
[root@servera ~]# virsh dumpxml myrhel8
<domain type='kvm' id='2'>
  <name>myrhel8</name>
  <uuid>c0dad148-a985-4f74-a894-fc4cf2468661</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/xmlns/libvirt/
domain/1.0">
      <libosinfo:os id="http://redhat.com/rhel/8.0"/>
    </libosinfo:libosinfo>
    <cockpit_machines:data xmlns:cockpit_machines="https://github.com/cockpit-
project/cockpit/tree/master/pkg/machines">
      <cockpit_machines:has_install_phase>false</
cockpit_machines:has_install_phase>
      <cockpit_machines:install_source>/var/tmp/boot.iso</
cockpit_machines:install_source>
      <cockpit_machines:os_variant>rhel8.0</cockpit_machines:os_variant>
    </cockpit_machines:data>
  </metadata>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
```

```
    <type arch='x86_64' machine='pc-q35-rhel7.6.0'>hvm</type>
    <boot dev='hd'/>
  </os>
...output omitted...
```

4.6. When you are done exploring, delete the virtual machine from the web console. Select the virtual machine and click **Delete**. Also, delete the associated storage file.

4.7. Log off from **servera**.

```
[root@servera ~]# exit
[student@workstation ~]$
```

5. Libvirt selects the machine chipset to emulate from the operating system you choose through the **Operating System** field in the web console or the **--os-variant** option of the **virt-install** command.

Using the web console, create a new virtual machine, but this time, select **Red Hat Enterprise Linux 6.10 (Santiago)** as the operating system. Even though you are still using the RHEL 8 ISO to install the system, the virtualization platform emulates the older Intel 440FX chipset.

5.1. In Firefox, click the **Virtual Machines** menu entry on the left of the web console. Click **Create VM** and fill in the form according to the following table.

Parameter	Value
Name	**testi440fx**
Installation Source Type	**Filesystem**
Installation Source	**/var/tmp/boot.iso**
OS Vendor	Red Hat, Inc
Operating System	**Red Hat Enterprise Linux 6.10 (Santiago)**
Memory	1 GiB
Storage Size	10 GiB

Click **Create**. If the state of the new virtual machine is **in transition**, refresh the web console's page. Click the **testi440fx** virtual machine and click **Install** to start the installation.

5.2. Use the kickstart at `http://172.25.250.254/min.ks` to perform an unattended installation.

In the console, use the **up** arrow key to highlight **Install Red Hat Enterprise Linux 8.0**. Press the **Tab** key. At the end of the line, at the bottom of the console, add **ks=http://172.25.250.254/min.ks** and press **Enter**.

Wait for the installation to complete. Because the virtual machine powers off after installation, click **Run** to start it.

5.3. In the console of the new virtual machine, log in as **root**, with **redhat** for the password.

Run the **dmidecode** command to confirm that QEMU uses the old Intel 440FX chipset emulation.

```
[root@localhost ~]# dmidecode
...output omitted...
Handle 0x0100, DMI type 1, 27 bytes
System Information
 Manufacturer: Red Hat
 Product Name: KVM
 Version: RHEL 7.6.0 PC (i440FX + PIIX, 1996)
 Serial Number: Not Specified
 UUID: 6d944c55-5045-4d7a-88bc-10c5ecca03dd
 Wake-up Type: Power Switch
 SKU Number: Not Specified
 Family: Red Hat Enterprise Linux
...output omitted...
```

5.4. Run the **lspci** command to inspect further the emulated hardware.

```
[root@localhost ~]# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Red Hat, Inc. QXL paravirtual graphic card (rev
 04)
00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device
00:04.0 Audio device: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) High
 Definition Audio Controller (rev 01)
00:05.0 USB controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller
 #1 (rev 03)
00:05.1 USB controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller
 #2 (rev 03)
00:05.2 USB controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller
 #3 (rev 03)
00:05.7 USB controller: Intel Corporation 82801I (ICH9 Family) USB2 EHCI
 Controller #1 (rev 03)
00:06.0 Communication controller: Red Hat, Inc. Virtio console
00:07.0 SCSI storage controller: Red Hat, Inc. Virtio block device
00:08.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon
```

Notice this time that the Intel 440FX chipset emulation provides an ISA bus but no PCI Express (PCI-E) ports.

5.5. When you are done exploring, delete the virtual machine from the web console. Select the virtual machine and click **Delete**. Also, delete the associated storage file.

Finish

From **workstation**, run **lab virt-config finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab virt-config finish
```

This concludes the guided exercise.

Lab

Adapting to Virtualization Improvements

Performance Checklist

In this lab, you will configure a system to host virtual machines, deploy a virtual machine using the web console, and import a virtual machine definition file.

Outcomes

You should be able to:

- Install the required packages to enable virtualization
- Create new virtual machines with the web console
- Import a virtual machine from its XML definition file and disk image

Before You Begin

Log in to **workstation** as **student** using **student** as the password. On **workstation**, run the **lab virt-review start** command to verify that the environment is ready and prepare the systems for the exercise.

```
[student@workstation ~]$ lab virt-review start
```

Note

In case **workstation** hangs using the web console, restart the virtual machine, and log into the web console with a new browser.

1. On **servera**, install the required packages for virtualization and configure the web console to manage virtual machines. The **root** password on **servera** is **redhat** and the **student** password is **student**.
2. Use the web console to deploy an RHEL 8 virtual machine on **servera** according to the following requirements.

Parameter	Value
VM name	**server8**
RHEL 8 DVD ISO file	**/var/tmp/boot.iso** on **servera**
Operating system	Red Hat Enterprise Linux 8.0
Memory	1 GiB
First disk	**vda**: 10 GiB
Second disk	**vdb**: 15 GiB
Kickstart URL	`http://172.25.250.254/min.ks`

3. On **servera**, import the virtual machine from the Libvirt XML dump in **/root/myrhel6.xml**.

   That file describes a Red Hat Enterprise Linux 6 system running on a Red Hat Enterprise 6.7 host. The associated disk image is already in **/var/lib/libvirt/images/myrhel6.img**.

   You may have to update the virtual machine definition to make it work on your RHEL 8 system.

   Remember that you can use the **virsh** command to manage the virtual machines:

   - **virsh define *file.xml*** creates a new virtual machine from an XML file.
   - **virsh start *vm_name*** starts the virtual machine.
   - **virsh edit *vm_name*** starts an editor to update the virtual machine definition.

   In this exercise, for resource constraints, the disk file associated with the virtual machine does not contain the image of a bootable operating system. If you access the console of the virtual machine, you will notice that the system is in an error state: this is expected.

Finish

From **workstation**, run **lab virt-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab virt-review finish
```

This concludes the lab.

Solution

Adapting to Virtualization Improvements

Performance Checklist

In this lab, you will configure a system to host virtual machines, deploy a virtual machine using the web console, and import a virtual machine definition file.

Outcomes

You should be able to:

- Install the required packages to enable virtualization
- Create new virtual machines with the web console
- Import a virtual machine from its XML definition file and disk image

Before You Begin

Log in to **workstation** as **student** using **student** as the password. On **workstation**, run the **lab virt-review start** command to verify that the environment is ready and prepare the systems for the exercise.

```
[student@workstation ~]$ lab virt-review start
```

Note

In case **workstation** hangs using the web console, restart the virtual machine, and log into the web console with a new browser.

1. On **servera**, install the required packages for virtualization and configure the web console to manage virtual machines. The **root** password on **servera** is **redhat** and the **student** password is **student**.

   1.1. Log in to **servera** as the **root** user.

```
[student@workstation ~]$ ssh root@servera
```

   1.2. Use **yum** to install the *virt* module.

```
[root@servera ~]# yum module install virt
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

   1.3. Confirm that the **servera** system supports virtualization.

```
[root@servera ~]# virt-host-validate
  QEMU: Checking for hardware virtualization                 : PASS
  QEMU: Checking if device /dev/kvm exists                   : PASS
  QEMU: Checking if device /dev/kvm is accessible            : PASS
  QEMU: Checking if device /dev/vhost-net exists             : PASS
  QEMU: Checking if device /dev/net/tun exists               : PASS
  QEMU: Checking for cgroup 'memory' controller support      : PASS
  QEMU: Checking for cgroup 'memory' controller mount-point  : PASS
  QEMU: Checking for cgroup 'cpu' controller support         : PASS
  QEMU: Checking for cgroup 'cpu' controller mount-point     : PASS
  QEMU: Checking for cgroup 'cpuacct' controller support     : PASS
  QEMU: Checking for cgroup 'cpuacct' controller mount-point : PASS
  QEMU: Checking for cgroup 'cpuset' controller support      : PASS
  QEMU: Checking for cgroup 'cpuset' controller mount-point  : PASS
  QEMU: Checking for cgroup 'devices' controller support     : PASS
  QEMU: Checking for cgroup 'devices' controller mount-point : PASS
  QEMU: Checking for cgroup 'blkio' controller support       : PASS
  QEMU: Checking for cgroup 'blkio' controller mount-point   : PASS
  QEMU: Checking for device assignment IOMMU support         : WARN (No ACPI
 DMAR table found, IOMMU either disabled in BIOS or not supported by this hardware
 platform)
```

You can safely ignore the warning message because **servera** is itself a virtual machine and its hardware emulation does not provide IOMMU support.

1.4. Install the *cockpit-machines* package.

```
[root@servera ~]# yum install cockpit-machines
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

1.5. Confirm that the web console is running and enabled on **servera**.

```
[root@servera ~]# systemctl is-active cockpit.socket
active
[root@servera ~]# systemctl is-enabled cockpit.socket
enabled
```

If that is not the case, start and enable the web console.

```
[root@servera ~]# systemctl enable --now cockpit.socket
```

1.6. On **workstation**, open Firefox and navigate to `https://servera.lab.example.com:9090`. If a certificate error appears, accept the self-signed certificate.

Select **Reuse my password for privileged tasks** and sign in using **student** as user name and **student** as the password.

Click on **servera.lab.example.com** in the left navigation bar and notice the **Virtual Machines** menu entry on the left.

2. Use the web console to deploy an RHEL 8 virtual machine on **servera** according to the following requirements.

Parameter	Value
VM name	**server8**
RHEL 8 DVD ISO file	**/var/tmp/boot.iso** on **servera**
Operating system	Red Hat Enterprise Linux 8.0
Memory	1 GiB
First disk	**vda**: 10 GiB
Second disk	**vdb**: 15 GiB
Kickstart URL	`http://172.25.250.254/min.ks`

2.1. In the web console in Firefox, click the **Virtual Machines** menu entry on the left. If you have not started the **libvirtd** Systemd service from the command line, then the web console offers to perform this step for you after the Yum *virt* module has been installed. Click **Start libvirt** to start and enable the **libvirtd** Systemd service.

2.2. Click **Create VM** and fill the form according to the following table.

Parameter	Value
Name	**server8**
Installation Source Type	**Filesystem**
Installation Source	**/var/tmp/boot.iso**
OS Vendor	Red Hat, Inc
Operating System	Red Hat Enterprise Linux 8.0
Memory	1 GiB
Storage Size	10 GiB

Do not check **Immediately Start VM** because you need add the second disk before starting the virtual machine. As an alternative, you could also create that second disk after the virtual machine installation.

Click **Create**. If the state of the new virtual machine is **in transition**, refresh the page.

2.3. To create the second disk, click the **server8** virtual machine and navigate to the **Disks** tab. Click **Add Disk** and fill the form according to the following table.

Source	**Create New**
Pool	**default**
Target	**vdb**
Name	**disk2**
Size	15 GiB
Format	**qcow2**

Click **Add**.

2.4. Click **Install** to start the installation. Use the kickstart at `http://172.25.250.254/min.ks` to perform an unattended installation.

In the console, use the **up** arrow key to highlight **Install Red Hat Enterprise Linux 8.0**. Press the **Tab** key. At the end of the line, at the bottom of the console, add **ks=http://172.25.250.254/min.ks** and press **Enter**.

Wait for the installation to complete. Because the virtual machine powers off after installation, click **Run** to start it.

3. On **servera**, import the virtual machine from the Libvirt XML dump in **/root/myrhel6.xml**.

That file describes a Red Hat Enterprise Linux 6 system running on a Red Hat Enterprise 6.7 host. The associated disk image is already in **/var/lib/libvirt/images/myrhel6.img**.

You may have to update the virtual machine definition to make it work on your RHEL 8 system.

Remember that you can use the **virsh** command to manage the virtual machines:

- **virsh define *file.xml*** creates a new virtual machine from an XML file.
- **virsh start *vm_name*** starts the virtual machine.
- **virsh edit *vm_name*** starts an editor to update the virtual machine definition.

In this exercise, for resource constraints, the disk file associated with the virtual machine does not contain the image of a bootable operating system. If you access the console of the virtual machine, you will notice that the system is in an error state: this is expected.

3.1. On **servera**, inspect the **/root/myrhel6.xml** file.

```
[root@servera ~]# cat /root/myrhel6.xml
<domain type='kvm'>
  <name>myrhel6</name>
  <uuid>194cae8e-3ce1-5b13-aba2-2c0d36d4cb38</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.6.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
```

```
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' cache='none'/>
      <source file='/var/lib/libvirt/images/myrhel6.img'/>
      <target dev='vda' bus='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
    </disk>
...output omitted...
```

Notice that the **machine** attribute is **rhel6.6.0**. Run the **virsh capabilities** command to confirm that this machine type is not available anymore in RHEL8 for the **hvm** type and the **x86_64** architecture.

```
[root@servera ~]# virsh capabilities
<capabilities>
...output omitted...
  <guest>
    <os_type>hvm</os_type>
    <arch name='x86_64'>
      <wordsize>64</wordsize>
      <emulator>/usr/libexec/qemu-kvm</emulator>
      <machine maxCpus='240'>pc-i440fx-rhel7.6.0</machine>
      <machine canonical='pc-i440fx-rhel7.6.0' maxCpus='240'>pc</machine>
      <machine maxCpus='240'>pc-i440fx-rhel7.0.0</machine>
      <machine maxCpus='384'>pc-q35-rhel7.6.0</machine>
      <machine canonical='pc-q35-rhel7.6.0' maxCpus='384'>q35</machine>
      <machine maxCpus='240'>pc-i440fx-rhel7.5.0</machine>
      <machine maxCpus='240'>pc-i440fx-rhel7.1.0</machine>
      <machine maxCpus='240'>pc-i440fx-rhel7.2.0</machine>
      <machine maxCpus='255'>pc-q35-rhel7.3.0</machine>
      <machine maxCpus='384'>pc-q35-rhel7.4.0</machine>
      <machine maxCpus='240'>pc-i440fx-rhel7.3.0</machine>
      <machine maxCpus='240'>pc-i440fx-rhel7.4.0</machine>
      <machine maxCpus='384'>pc-q35-rhel7.5.0</machine>
      <domain type='qemu'/>
      <domain type='kvm'>
        <emulator>/usr/libexec/qemu-kvm</emulator>
      </domain>
    </arch>
    <features>
      <cpuselection/>
      <deviceboot/>
      <disksnapshot default='on' toggle='no'/>
      <acpi default='on' toggle='yes'/>
      <apic default='on' toggle='no'/>
```

```
    </features>
  </guest>

</capabilities>
```

Do not update the **/root/myrhel6.xml** file yet.

Also notice that the **/var/lib/libvirt/images/myrhel6.img** image file is the virtual machine disk. The exercise setup script has already deployed that file for you.

3.2. Create the virtual machine from the XML file.

```
[root@servera ~]# virsh define /root/myrhel6.xml
Domain myrhel6 defined from /root/myrhel6.xml
```

3.3. Start the **myrhel6** virtual machine.

```
[root@servera ~]# virsh start myrhel6
error: Failed to start domain myrhel6
error: internal error: process exited while connecting to monitor:
 2019-02-08T15:08:06.881725Z qemu-kvm: -machine rhel6.6.0,accel=kvm,usb=off,dump-
guest-core=off: unsupported machine type
Use -machine help to list supported machines
```

The virtual machine fails to start. This confirms that the **machine** attribute is incorrect.

3.4. Edit the virtual machine definition. Because the machine comes from an RHEL 6.7 virtualization host, select a machine that emulates the Intel 440FX chipset, which is the only one available on RHEL 6.7.

```
[root@servera ~]# virsh edit myrhel6
<domain type='kvm'>
  <name>myrhel6</name>
  <uuid>194cae8e-3ce1-5b13-aba2-2c0d36d4cb38</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc'>hvm</type>
    <boot dev='hd'/>
  </os>
...output omitted...
```

The **pc** machine is an alias for the Intel 440FX chipset emulation.

3.5. Try again to start the **myrhel6** virtual machine.

```
[root@servera ~]# virsh start myrhel6
Domain myrhel6 started

[root@servera ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 8     server8                        running
 9     myrhel6                        running
```

This time the virtual machine starts with no error.

In this exercise, for resource constraints, the disk file associated with the **myrhel6** virtual machine does not contain the image of a bootable operating system. If you access the console of the virtual machine, you see that the system is in an error state: this is expected.

3.6. If you have time, continue exploring the web console and the virtualization tools. When you are done, close Firefox and log off from **servera**.

Finish

From **workstation**, run **lab virt-review finish** to clean up any resources created for this exercise.

```
[student@workstation ~]$ lab virt-review finish
```

This concludes the lab.

Summary

In this chapter, you learned:

- QEMU now emulates the Intel Q35 chipset, offering a better hardware platform for modern virtualized operating systems.
- The *cockpit-machines* package adds basic functionality for administering virtual machines in the web console.
- The **`virt-manager`** graphical interface is still available but deprecated.