

## Red Hat System Administration II

Red Hat Enterprise Linux 10.0 RH134

Student Workbook

Edition 2

## Table of Contents

---

### Red Hat System Administration II

#### Document Conventions

- Admonitions
- Use of Artificial Intelligence
- Inclusive Language

#### Introduction

- Red Hat System Administration II
- Orientation to the Classroom Environment
- Performing Lab Exercises
- Viewing the Student Guide

### Chapter 1. Shell Scripting and the Command Line

- 1.1. Changing the Shell Environment
- 1.2. Guided Exercise Change the Shell Environment
- 1.3. Writing Simple Bash Scripts
- 1.4. Guided Exercise Write Simple Bash Scripts
- 1.5. Running Loops and Conditional Commands
- 1.6. Guided Exercise Run Loops and Conditional Commands
- 1.7. Lab Shell Scripting and the Command Line
- 1.8. Summary

### Chapter 2. Using Regular Expressions for Practical Applications

- 2.1. Matching Text with Regular Expressions
- 2.2. Guided Exercise Match Text with Regular Expressions
- 2.3. Quiz Matching Text with Regular Expressions
- 2.4. Summary

### Chapter 3. Scheduling User Tasks

- 3.1. Scheduling a Future User Job

- 3.2. Guided Exercise Schedule a Future User Job
- 3.3. Scheduling Recurring User Jobs
- 3.4. Guided Exercise Schedule Recurring User Jobs
- 3.5. Quiz Scheduling User Tasks
- 3.6. Summary

## **Chapter 4. Scheduling System Tasks**

- 4.1. Managing Repeating Jobs by Using Systemd Timer Units
- 4.2. Guided Exercise Manage Repeating Jobs by Using Systemd Timer Units
- 4.3. Managing Temporary Files
- 4.4. Guided Exercise Manage Temporary Files
- 4.5. Scheduling Recurring System Tasks with Cron
- 4.6. Guided Exercise Schedule a Recurring System Task with Cron
- 4.7. Quiz Scheduling User Tasks
- 4.8. Summary

## **Chapter 5. Analyzing and Storing Logs**

- 5.1. The System Log Architecture
- 5.2. Quiz The System Log Architecture
- 5.3. Interpreting and Managing Syslog Events
- 5.4. Guided Exercise Interpret and Manage Syslog Events
- 5.5. Finding and Interpreting System Journal Log Entries
- 5.6. Guided Exercise Find and Interpret System Journal Log Entries
- 5.7. Configuring a Persistent System Journal
- 5.8. Guided Exercise Configure a Persistent System Journal
- 5.9. Maintaining Synchronized Time
- 5.10. Guided Exercise Maintain Synchronized Time
- 5.11. Lab Analyze and Store Logs
- 5.12. Summary

## **Chapter 6. Managing Security with SELinux**

- 6.1. Operating SELinux

- 6.2. Guided Exercise Operate SELinux
- 6.3. Controlling SELinux File Contexts
- 6.4. Guided Exercise Control SELinux File Contexts
- 6.5. Tuning the SELinux Policy by Adjusting Booleans
- 6.6. Guided Exercise Tune the SELinux Policy by Adjusting Booleans
- 6.7. Investigating and Resolving SELinux Issues
- 6.8. Guided Exercise Investigate and Resolve SELinux Issues
- 6.9. Lab Manage Security with SELinux
- 6.10. Summary

## **Chapter 7. Archiving Files**

- 7.1. Managing Compressed Tar Archives
- 7.2. Guided Exercise Manage Compressed Tar Archives
- 7.3. Summary

## **Chapter 8. Transferring Files**

- 8.1. Transferring Files Between Systems
- 8.2. Guided Exercise Transfer Files Between Systems
- 8.3. Synchronizing Content Between Systems
- 8.4. Guided Exercise Synchronize Content Between Systems
- 8.5. Lab Transfer Files
- 8.6. Summary

## **Chapter 9. Tuning System Performance**

- 9.1. Setting a Tuning Profile
- 9.2. Guided Exercise Set a Tuning Profile
- 9.3. Influencing Process Scheduling
- 9.4. Guided Exercise Influence Process Scheduling
- 9.5. Lab Tune System Performance
- 9.6. Summary

## **Chapter 10. Managing Basic Storage**

- 10.1. Creating and Managing File Systems on Standard Partitions

- 10.2. Guided Exercise Create and Manage File Systems on Standard Partitions
- 10.3. Managing Swap Space
- 10.4. Guided Exercise Manage Swap Space
- 10.5. Lab Manage Basic Storage
- 10.6. Summary

## **Chapter 11. Managing Storage with Logical Volume Manager**

- 11.1. Creating Logical Volumes
- 11.2. Guided Exercise Create Logical Volumes
- 11.3. Extending a Logical Volume
- 11.4. Guided Exercise Extend a Logical Volume
- 11.5. Replacing Physical Volumes and Managing LVM
- 11.6. Quiz Replacing Physical Volumes and Managing LVM
- 11.7. Lab Manage Storage with Logical Volume Manager
- 11.8. Summary

## **Chapter 12. Controlling and Troubleshooting the Boot Process**

- 12.1. Managing the Boot Loader and Kernel Command Line
- 12.2. Guided Exercise Manage the Boot Loader and Kernel Command Line
- 12.3. Exploring the Boot Process and Selecting a Boot Target
- 12.4. Guided Exercise Explore the Boot Process and Select a Boot Target
- 12.5. Repairing Damaged File Systems at Boot Time
- 12.6. Guided Exercise Repair a Damaged File System at Boot Time
- 12.7. Lab Control and Troubleshoot the Boot Process
- 12.8. Summary

## **Chapter 13. Recovering Superuser Access**

- 13.1. Regaining Superuser Access and Resetting the Root Password
- 13.2. Guided Exercise Regain Superuser Access and Reset the Root Password
- 13.3. Summary

## **Chapter 14. Managing Network Security**

- 14.1. Managing Server Firewalls

- 14.2. Guided Exercise Manage a Server Firewall
- 14.3. SELinux Port Labeling
- 14.4. Guided Exercise SELinux Port Labeling
- 14.5. Lab Manage Network Security
- 14.6. Summary

## **Chapter 15. Accessing Network-attached Storage**

- 15.1. Mounting NFS File Systems
- 15.2. Guided Exercise Mount NFS File Systems
- 15.3. Automounting Storage Devices
- 15.4. Guided Exercise Automount Storage Devices
- 15.5. Lab Access Network-attached Storage
- 15.6. Summary

## **Chapter 16. Installing Red Hat Enterprise Linux**

- 16.1. Installing Red Hat Enterprise Linux Interactively
- 16.2. Guided Exercise Install Red Hat Enterprise Linux Interactively
- 16.3. Automating Red Hat Enterprise Linux Installation with Kickstart
- 16.4. Guided Exercise Automate Red Hat Enterprise Linux Installation with Kickstart
- 16.5. Quiz Installing Red Hat Enterprise Linux
- 16.6. Summary

## **Chapter 17. Managing Containers with Podman**

- 17.1. Introduction to Containers
- 17.2. Quiz Introduction to Containers
- 17.3. Running Containers with Podman
- 17.4. Guided Exercise Run Containers with Podman
- 17.5. Creating and Managing Container Images
- 17.6. Guided Exercise Create and Manage Container Images
- 17.7. Lab Manage Containers with Podman
- 17.8. Summary

## **Chapter 18. Working with Image-based Red Hat Enterprise Linux**

- 18.1. Image Mode for Red Hat Enterprise Linux
- 18.2. Quiz Image Mode for Red Hat Enterprise Linux
- 18.3. Creating Installable Images for Image Mode
- 18.4. Guided Exercise Create Installable Images for Image Mode
- 18.5. Installing Red Hat Enterprise Linux by Using Image Mode
- 18.6. Guided Exercise Install Red Hat Enterprise Linux by Using Image Mode
- 18.7. Managing Image Mode-based Systems
- 18.8. Guided Exercise Manage Image Mode-based Systems
- 18.9. Summary

## **Chapter 19. Comprehensive Review**

- 19.1. Comprehensive Review
- 19.2. Lab Fix Boot Issues and Maintain Servers
- 19.3. Lab Configure and Manage File Systems and Storage
- 19.4. Lab Configure and Manage Server Security
- 19.5. Lab Run Containers

## **Red Hat System Administration II**

---

### **Red Hat Enterprise Linux 10.0 RH134**

#### **Edition 2**

**Publication date 2025-08-21**

<b>Authors:</b>	Alex Callejas, Ashley D'Andrea, Hernán Del Negro, Shreya Dhang, Austin Garrigus, Patrick Gomez, Ashish Lingayat, Zoltan Molnar, Mauricio Santacruz Delgado
<b>Course Architects:</b>	Steven Bonneville, Ashish Lingayat
<b>DevOps Engineers:</b>	Trey Feagle, Artur Glogowski, Nikki Lucas, Travis Michette, Jordi Sola
<b>Editors:</b>	Julian Cable, Rachel Lee
<b>Contributors:</b>	Sajith Eyamkuzhy Sugathan, Dusty Powell, Wasim Raja, David Sacco, Jonika Shank, Chetan Tiwary, Roberto Velazquez

---

© 2025 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are © 2025 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed, please send email to [training@redhat.com](mailto:training@redhat.com) or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, OpenShift, Fedora, Hibernate, Ansible, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle American, Inc. and/or its affiliates.

XFS® is a registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is a trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

## Document Conventions

---

This section describes various conventions and practices that are used throughout all Red Hat Training courses.

## Admonitions

---

Red Hat Training courses use the following admonitions:

### References

References describe where to find external documentation that is 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 something that makes your life easier.

### Important

Important sections provide details of information that is easily missed: configuration changes that apply only to the current session, or services that need restarting before an update applies. Ignoring these admonitions will not cause data loss, but might cause irritation and frustration.

### Warning

Do not ignore warnings. Ignoring these admonitions will most likely cause data loss.

## Use of Artificial Intelligence

---

This course contains content or media that is generated or assisted by AI tools. All Red Hat Training deliverables have primarily human authorship, and all content was reviewed by a human domain expert.

## Inclusive Language

---

## **Document Conventions**

Red Hat Training is currently reviewing its use of language in various areas to help remove any potentially offensive terms. This process is ongoing and requires alignment with the products and services that are covered in Red Hat Training courses. Red Hat appreciates your patience during this process.

## Introduction

---

# Red Hat System Administration II

Perform key Linux system administration tasks at a deeper level than the *Red Hat System Administration I* (RH124) course, including installation, storage management, SELinux mandatory access control security, scheduled program execution, system log monitoring and simple tuning, and maintenance.

## Course Objectives

- Write and run simple shell scripts, and use shell scripting features to efficiently run commands at the shell prompt.
- Efficiently complete system administration tasks by using regular expressions to match text.
- Schedule programs to run in the future, either at a specific time or on a recurring basis, as a regular user.
- Schedule system programs that must run on a recurring basis to support daemons or operating system functions.
- Locate and interpret system logs for troubleshooting purposes, and ensure accurate timestamps for log events.
- Protect systems and manage security by using SELinux.
- Create compressed archives of files so that they can be backed up and transferred to other systems.
- Securely transfer files from one system to another.
- Improve system performance by setting a tuning profile and by adjusting the scheduling priority of specific processes.
- Manage storage devices by creating partitions, file systems, and swap spaces from the command line.
- Use Logical Volume Manager (LVM) to manage logical volumes that can contain file systems and swap spaces.
- Manage how the system boots to control which services start and to troubleshoot and repair boot-time problems.

## Introduction

- Gain administrative access to a system when the superuser password is unknown or is locked.
- Control network connections to services by using the system firewall, and network services that can bind to particular ports by using SELinux.
- Access network-attached storage that the Network File System (NFS) protocol provides, either manually or by using the automounter.
- Install Red Hat Enterprise Linux in package mode, either interactively or by using Kickstart.
- Manage containers and container images with the latest version of container management tools.
- Create, use, install, and upgrade containers and servers that use image-based installation management.
- Test the knowledge that you acquired from the *Red Hat System Administration II* (RH134) course and prepare for the *Red Hat Certified System Administrator* (EX200) exam.

## Audience

- System administrators, platform engineers, developers, and other IT professionals who have completed the *Red Hat System Administration I* (RH124) course, and are seeking to expand their skills in Linux system administration.

## Prerequisites

- Students should have completed the *Red Hat System Administration I* (RH124) course or be familiar with all topics that are included in that course before taking this course.
- Students who seek a more rapid overview of topics in preparation for the *Red Hat Certified System Administrator* (EX200) exam and who do not meet this requirement should consider attending the *Red Hat Certified System Administrator Rapid Track* (RH199) course instead.

## Orientation to the Classroom Environment

---

In this course, the main computer system that is used for hands-on learning activities is the workstation machine.

You also use three other machines for these activities: the servera, serverb, and serverc machines.

These machines are in the lab.example.com DNS domain.

## Introduction

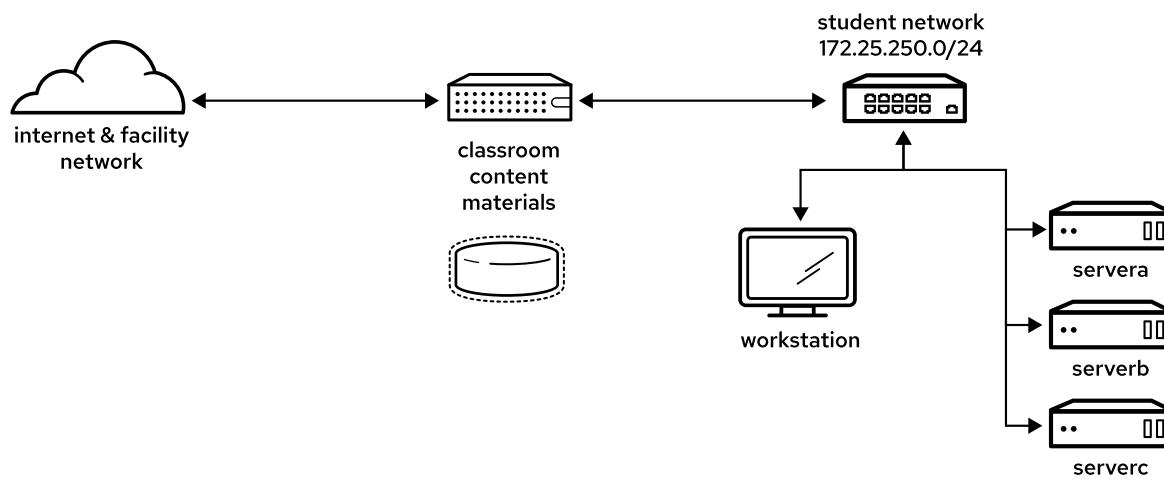


Figure 1. Classroom environment

Table 1. Classroom Machines

Machine name	IP addresses	Role
classroom.example.com	172.25.252.254	Server that hosts the required classroom materials
content.example.com	172.25.254.254	
materials.example.com		
utility.lab.example.com	172.25.250.8	Server that hosts course-specific support services
workstation.lab.example.com	172.25.250.9	Graphical workstation that students use
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"

## Controlling Your Systems

## Introduction

In an instructor-led training classroom, you are assigned a physical computer (`foundationX.ilt.example.com`), which accesses the virtual machines that run on that host. You are automatically logged in to the physical machine as the kiosk user, by using `redhat` as the password. The classroom systems that you work with are virtual machines that run on that host.

On the `foundationX` machine, you use the `rht-vmctl` and `rht-vmview` commands to work with the virtual machines. Run the commands in the following table as the kiosk user on the `foundationX` machine. You can use these commands with any of the virtual machines.

**Table 2. `rht-vmctl` Commands**

Action	Command
Start the virtual machine named <code>server</code> .	<code>rht-vmctl start server</code>
Display the system console for the virtual machine named <code>server</code> , to log in and work on that system.	<code>rht-vmview view server</code>
Reset the virtual machine named <code>server</code> to its initial course state and restart it.	<code>rht-vmctl reset server</code>
At the start of a lab exercise, if the "reset your servera" instruction appears, then run the <code>rht-vmctl reset servera</code> command on the <code>foundationX</code> machine as the kiosk user. Likewise, if the "reset your workstation" instruction appears, then run the <code>rht-vmctl reset workstation</code> command on the <code>foundationX</code> machine as the kiosk user.	
You are assigned remote computers in a Red Hat Online Learning (ROLE) classroom. Self-paced courses are accessed through a web application that is hosted at <a href="http://rol.redhat.com">rol.redhat.com</a> . Log in to this site with your Red Hat Customer Portal user credentials.	

## Controlling the Virtual Machines

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

## Introduction

The screenshot shows a user interface for managing lab environments. At the top, there are three expandable sections: 'SSH Private Key & Instructions' (with a 'DOWNLOAD SSH KEY' button), 'Classroom Webapp' (with a 'ACCESS CLASSROOM WEBAPP' button), and 'Lab Controls' (with a 'WATCH TUTORIAL' button). Below these are three primary buttons: 'DELETE' (red), 'STOP' (teal), and an information icon ('i'). A table follows, listing six virtual machines: 'classroom' (Active), 'servera' (Active), 'serverb' (Active), 'serverc' (Active), 'utility' (Active), and 'workstation' (Active). Each row has an 'ACTION' dropdown and an 'OPEN CONSOLE' button.

Virtual machine	State	Action	Open Console
classroom	Active	ACTION	OPEN CONSOLE
servera	Active	ACTION	OPEN CONSOLE
serverb	Active	ACTION	OPEN CONSOLE
serverc	Active	ACTION	OPEN CONSOLE
utility	Active	ACTION	OPEN CONSOLE
workstation	Active	ACTION	OPEN CONSOLE

Figure 2. An example course lab environment management page

Table 3. Machine States

Virtual machine state	Description
Building	The virtual machine is being created.
Active	The virtual machine is running and available. If the virtual machine just started, then it might still be starting services.
Stopped	The virtual machine is shut down. On starting, the virtual machine boots into the same state as before shutdown. The disk state is preserved.

## Introduction

Table 4. Classroom Actions

Button or action	Description
<b>CREATE</b>	Create the ROLE classroom. Creates and starts all the necessary virtual machines for this classroom.
<b>CREATING</b>	The ROLE classroom virtual machines are being created. Creation can take several minutes to complete.
<b>DELETE</b>	Delete the ROLE classroom. Deletes all virtual machines in the classroom. <b>All saved work on those systems' disks is lost.</b>
<b>START</b>	Start all virtual machines in the classroom.
<b>STARTING</b>	All virtual machines in the classroom are starting.
<b>STOP</b>	Stop all virtual machines in the classroom.

Table 5. Machine Actions

Button or action	Description
<b>OPEN CONSOLE</b>	Connect to the system console of the virtual machine in a new browser tab. You can log in directly to the virtual machine and run commands, when required. Normally, log in to the workstation virtual machine only, and from there, use ssh to connect to the other virtual machines.
<b>ACTION &gt; Start</b>	Start (power on) the virtual machine.
<b>ACTION &gt; Shutdown</b>	Gracefully shut down the virtual machine, and preserve disk contents.

Button or action	Description
<b>ACTION &gt; Power Off</b>	Forcefully shut down the virtual machine, and preserve disk contents. This action is equivalent to removing the power from a physical machine.
<b>ACTION &gt; Reset</b>	Forcefully shut down the virtual machine and reset the associated storage to its initial state. <b>All saved work on that system's disks is lost.</b>

At the start of an exercise, if you are instructed to reset a single virtual machine node, then click

**ACTION > Reset** for that specific virtual machine only.

At the start of an exercise, if you are instructed to reset *all* virtual machines, then click **ACTION > Reset** on every virtual machine in the list.

To return the classroom environment to its original state at the start of the course, you can click

**DELETE** to remove the entire classroom environment. After the lab is deleted, click **CREATE** to provision a new set of classroom systems.

#### **Warning**

The **DELETE** operation cannot be undone. All completed work in the classroom environment is lost.

## **The Auto-stop and Auto-destroy Timers**

The Red Hat Online Learning enrollment entitles you to a predefined allotment of computer time. To help to conserve your allotted time, the ROLE classroom uses timers, which shut down or delete the classroom environment when the appropriate timer expires.

To adjust the timers, locate the two plus-sign  buttons at the bottom of the course management page. Click the auto-stop  button to add another hour, up to a maximum of 12 hours for the auto-stop timer. Click the auto-destroy  button to add another day, up to a maximum of 14 days for the auto-destroy timer. Be careful to keep the timers set when you are working, so that your environment is not unexpectedly shut down. Be careful not to set the timers unnecessarily high, which could waste your subscription time allotment.

## Performing Lab Exercises

---

You might see the following lab activity types in this course:

- A *guided exercise* is a hands-on practice exercise that follows a presentation section. It walks you through a procedure to perform, step by step.
- A *quiz* is typically used when checking knowledge-based learning, or when a hands-on activity is impractical for some other reason.
- An *end-of-chapter lab* is a gradable hands-on activity to help you to check your learning. You work through a set of high-level steps, based on the guided exercises in that chapter, but the steps do not walk you through every command. A solution is provided with a step-by-step walk-through.

To prepare your lab environment at the start of each hands-on activity, run the `lab start` command with a specified activity name from the activity's instructions. Likewise, at the end of each hands-on activity, run the `lab finish` command with that same activity name to clean up after the activity.

Each hands-on activity has a unique name within a course.

The syntax for running an exercise script is as follows:

```
student@workstation:~$ lab action exercise
```

The *action* is a choice of the `install`, `select`, `list`, `start`, `grade`, or `finish` actions. All exercises support `start` and `finish`. Only end-of-chapter labs and comprehensive review labs support the `grade` action.

`install`

The `install` action installs the exercise files for the specified course or lesson.

`select`

The `select` action helps to select a course or a lesson from multiple installed packages.

`list`

This action lists all the currently installed packages in your lab environment.

## Introduction

### start

The `start` action verifies the required resources to begin an exercise. It might include configuring settings, creating resources, checking prerequisite services, and verifying necessary outcomes from previous exercises. Usually, you can perform an exercise at any time, even without performing preceding exercises. Occasionally, a previous activity might be required.

### grade

For gradable activities, the `grade` action directs the `lab` command to evaluate your work, and shows a list of grading criteria with the `PASS` or `FAIL` status for each one. To achieve the `PASS` status for all criteria, fix the failures and rerun the `grade` action.

### finish

The `finish` action cleans up resources that were configured during the exercise. You can perform an exercise as many times as you want.

The `lab` command supports tab completion. For example, to list all exercises that you can start, enter `lab start` and then press `Tab` twice. The `lab` command also includes suggestions to avoid potential issues when running the `start` and `finish` actions.

## Viewing the Student Guide

---

Commands and code blocks in the student guide are best viewed when your preferred viewer displays at least 80 characters per line. If you adjust your viewer to display fewer than 80 characters per line, then commands and code blocks might wrap at unexpected locations. An unexpected line break might lead to confusion and to running a command incorrectly or to introducing a mistake in a file.

The following line is 80 characters long. For the best viewing experience, adjust your viewer so that the following line does not wrap. If you are using a web browser, then you might adjust the zoom percentage. For an EPUB viewer, you might adjust the orientation and the font size.

```
=====
```

=

## Chapter 1.

# Shell Scripting and the Command Line

---

## Goal

Write and run simple shell scripts, and use shell scripting features to efficiently run commands at the shell prompt.

## Sections

- Changing the Shell Environment (and Guided Exercise)
- Writing Simple Bash Scripts (and Guided Exercise)
- Running Loops and Conditional Commands (and Guided Exercise)

## Lab

- Shell Scripting and the Command Line

## 1.1. Changing the Shell Environment

### Objectives

- Describe shell variables for use with commands, and export environment variables to modify the behavior of the shell and programs that the shell runs.

### Shell Variables

In the Bash shell, you can set *shell variables* to help to run commands or to modify the behavior of the shell. You can also export shell variables as *environment variables*, which are automatically copied to programs that are run from that shell. You can use variables for ease of running a command with a long argument, or to apply a common setting to commands that are run from that shell.

Shell variables are unique to a particular shell session. If two terminal windows are open, or there are two independent login sessions to the same remote server, then you are running two shells. Each shell has its own set of values for its shell variables.

### Variable Assignment

You assign a value to a shell variable with the following syntax:

```
user@host:~$ VARIABLENAME=value
```

Variable assignment cannot have spaces around the equal sign.

Variable names can contain uppercase or lowercase letters, digits, and the underscore character (\_).

For example, the following commands set shell variables to different values.

Set the COUNT variable to 40:

```
user@host:~$ COUNT=40
```

Set the first\_name variable to John:

```
user@host:~$ first_name=John
```

Set the full\_name variable to John Smith:

```
user@host:~$ full_name='John Smith'
```

Set the `file1` variable to `/tmp/abc`:

```
user@host:~$ file1=/tmp/abc
```

Set the `_ID` variable to `Training`:

```
user@host:~$ _ID=Training
```

Many variable naming conventions exist, depending on a variable's purpose and scope. For example, variables that are automatically set by the shell have names with all-uppercase characters. If you are setting your own variables, then you might want to use names with lowercase characters to prevent naming collisions.

This lesson uses uppercase letters to name shell variables.

If you modify a variable within a shell session, then it is changed only for that session. When a new shell is opened, the variable is the default value.

You can use the `set` command to list all shell variables that are currently set. (The `set` command also lists all shell functions, which you can ignore.) To improve readability, you can pipe the output to the `less` command so that you can view it one page at a time.

```
user@host:~$ set | less
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquot
e\
:force_fignore:globasciiranges:histappend:interactive_comments:login_shell:prog
c\
omp:promptvars:sourcepath
BASHRCSCORED=Y
...output omitted...
```

## Variable Data Types

The Bash shell supports the string, number, and array data types. Each data type achieves specific outcomes. For example, you can perform simple integer arithmetic on numbers, or look up elements of arrays.

 Note

Bash also supports the associative array data type, which is out of scope for this lesson.

String variables can include alphanumeric characters. When assigning a value to a string variable, you can enclose the value in single or double quotation marks.

Enclosing the value in quotation marks is necessary only if the value contains a space character, or if you do not escape all space characters. Additionally, you can escape a specific character by using the backslash character.

```
user@host:~$ bare=This\ string\ escapes\ all\ spaces.
```

If a string is enclosed in double quotation marks, then Bash can expand internal variables and commands.

```
user@host:~$ double_quote="This is an expandable string where the ${variable} variable would be replaced."
```

A string that is enclosed in single quotation marks, however, denotes a literal string, and Bash does not expand it.

```
user@host:~$ single_quote='This is a literal string with special characters like the $ character.'
```

Number variables include only numeric characters and are not enclosed in quotation marks. You can perform basic arithmetic on number variables, such as addition, subtraction, multiplication, and division, by using the Bash arithmetic expansion syntax `$((expression))`.

Set the `five` variable to 5:

```
user@host:~$ five=5
```

Set the `ten` variable to 10:

```
user@host:~$ ten=10
```

Add the `five` and `ten` variables:

```
user@host:~$ fifteen=$((five + ten))
```

Array variables are indexed sequences of data. You can assign a value to an array variable by enclosing space-separated elements in parentheses. You can access specific elements in an array by their index, or iterate over elements of the array. All elements of arrays are treated as strings.

```
user@host:~$ colors=('red' 'green' 'blue')
```

## Retrieving Values with Variable Expansion

You can use *variable expansion* to refer to the value of a variable that you set. To use variable expansion, precede the name of the variable with a dollar sign (\$).

In the following examples, the variable expansion occurs first and then the echo command prints the rest of the command line that is entered. For example, the following command sets the COUNT variable to 40.

```
user@host:~$ COUNT=40
```

If you enter the echo COUNT command, then the command prints the COUNT string.

```
user@host:~$ echo COUNT  
COUNT
```

If you enter instead the echo \$COUNT command, then the command prints the value of the COUNT variable.

```
user@host:~$ echo $COUNT  
40
```

You can expand a variable in a string with double quotation marks.

```
user@host:~$ echo "The count is: $COUNT"  
The count is: 40
```

Variable expansion is ignored in strings with single quotation marks.

```
user@host:~$ echo 'The count is: $COUNT'  
The count is: $COUNT
```

You can enclose the variable name in curly braces to reference it more explicitly.

In the following example, curly braces are not used. The echo command tries to expand the nonexistent COUNTx variable, which returns nothing.

```
user@host:~$ echo "Repeat $COUNTx"  
Repeat
```

If any trailing characters are next to a variable name, then delimit the variable name with curly braces.

The following example uses curly braces. The echo command now expands the COUNT variable correctly.

```
user@host:~$ echo "Repeat ${COUNT}x"  
Repeat 40x
```

You can also use the curly brace syntax to access array elements with the array[] syntax. Note that arrays begin with an index of 0.

```
user@host:~$ colors=("red" "green" "blue")
```

The following example accesses the element of an array at index 2.

```
user@host:~$ echo "The sky is ${colors[2]}"  
The sky is blue
```

You can access all elements in an array by using an at sign (@) or an asterisk (\*).

```
user@host:~$ echo "The human eye can see the following colors: ${colors[@]}"  
The human eye can see the following colors: red green blue
```

### Note

The curly brace syntax also enables many advanced operations, such as pattern replacement, default values, and length evaluation. Refer to the bash(1) man page for more information.

## Configure Bash with Shell Variables

Some shell variables are set when Bash starts. You can modify them to adjust the shell's behavior. Remember, this modification affects only the shell that you run the command in, not any other shells that you might be running on that server.

For example, the `HISTFILE`, `HISTFILESIZE`, and `HISTTIMEFORMAT` shell variables affect the shell history and the `history` command.

The `HISTFILE` variable specifies which file to save the shell history to, and defaults to the `~/.bash_history` file.

The `HISTFILESIZE` variable specifies how many commands to save in that file from the history.

The `HISTTIMEFORMAT` variable defines the time stamp format for every command in the history. This variable does not exist by default.

In the following example, the output of the `history` command does not show any timestamp:

```
user@host:~$ history
...output omitted...
 6  ls /etc
 7  uptime
 8  ls -l
 9  date
10  history
```

Set the timestamp format to the `%Y-%m-%d` date format and the `%H:%M:%S` 24-hour notation:

```
user@host:~$ HISTTIMEFORMAT="%F %T "
```

You can verify that the timestamp is now shown in the output of the `history` command:

```
user@host:~$ history
...output omitted...
 6 2022-05-03 04:58:11 ls /etc
 7 2022-05-03 04:58:13 uptime
 8 2022-05-03 04:58:15 ls -l
 9 2022-05-03 04:58:16 date
10 2022-05-03 04:58:18 history
11 2022-05-03 04:59:10 HISTTIMEFORMAT="%F %T "
12 2022-05-03 04:59:12 history
```

Another example is the PS1 variable, which controls the appearance of the shell prompt. If you change this value, then it changes the appearance of your shell prompt.

Red Hat recommends ending the prompt with a trailing space to help differentiate the prompt from commands. Various special character expansions that the prompt supports are listed in the "PROMPTING" section of the bash(1) man page.

For example, you can set the bash command prompt to the bash\$ prompt:

```
user@host:~$ PS1="bash\$ "
bash$
```

You can also set the bash command prompt to appear in the following format:

```
bash$ PS1="[\u@\h \W]\$ "
[user@host ~]$
```

The \u prompt string displays the username of the current user.

The \h prompt string displays the hostname until the first period character ( . ).

The \W prompt string displays the basename of the \$PWD variable, and the \$HOME variable is abbreviated with a tilde (~).

The \\$ prompt string displays # if the effective UID is 0; otherwise, this prompt string displays \$.

## Configuring Programs with Environment Variables

The shell provides an *environment* for the programs that you run from that shell. Among other items, this environment includes information about the current working directory on the file system, the

command-line options that are passed to the program, and the values of environment variables. You can use environment variables to change the behavior of programs or their default settings.

To list all the environment variables for a shell, use the `env` command:

```
user@host:~$ env
...output omitted...
LANG=en_US.UTF-8
HISTCONTROL=ignoredups
HOSTNAME=host.example.com
XDG_SESSION_ID=4
...output omitted...
```

If a shell variable is not an environment variable, then only the shell can use it. However, if a shell variable is an environment variable, then the shell and any programs that run from that shell can use that variable.

#### Note

The `HISTFILE`, `HISTFILESIZE`, and `PS1` variables from the previous section do not need to be exported as environment variables, because only the shell itself uses them, not the programs that you run from the shell.

You can assign any variable that is defined in the shell as an environment variable by marking it for export with the `export` command.

The following example sets Vim as the text editor.

```
user@host:~$ EDITOR=vim
```

Export the `EDITOR` variable.

```
user@host:~$ export EDITOR
```

You can set and export a variable in one step:

```
user@host:~$ export EDITOR=vim
```

The shell automatically sets the HOME variable to the file name of the user's home directory when the shell starts. You can use this variable to help programs to determine where to save files.

The LANG variable sets the locale encoding. This variable adjusts the preferred language for program output; the character set; the formatting of dates, numbers, and currency; and the sort order for programs. If the LANG variable is set to en\_US.UTF-8, then the locale uses US English with UTF-8 Unicode character encoding. Or, if this variable is set to fr\_FR.UTF-8, then the locale uses French UTF-8 Unicode encoding.

In the following example, verify that the date command uses the en\_US.UTF-8 format:

```
user@host:~$ date  
Tue May 27 17:04:37 UTC 2025
```

Export the LANG variable to the fr\_FR.UTF-8 value:

```
user@host:~$ export LANG=fr_FR.UTF-8
```

Verify that the date command now shows the new format that the LANG variable sets:

```
user@host:~$ date  
mar. 27 mai 2025 17:04:51 UTC
```

The PATH environment variable contains a list of colon-separated directories that define where executable commands exist:

```
user@host:~$ echo $PATH  
/home/user/.local/bin:/home/user/bin:/usr/share/Modules/bin:/usr/local/bin:/usr  
/bin:/usr/local/sbin:/usr/sbin
```

When you run a command such as the ls command, the shell looks for the ls executable file in each of those directories in order, and runs the first matching file that it finds. (On a typical Red Hat system, this file is the /usr/bin/ls executable file.)

You can append or prepend directories to your PATH variable. For example, you want to run some executable programs or scripts such as regular commands in the /home/user/sbin directory. You can append the /home/user/sbin directory to your PATH variable for the current session as follows:

```
user@host:~$ export PATH=${PATH}:/home/user/sbin
```

## Set the Default Text Editor

The EDITOR environment variable specifies your default text editor for command-line programs. Many programs use the vi or vim editor if not specified, and you can override this preference:

```
user@host:~$ export EDITOR=nano
```

## Setting Variables Automatically

When Bash starts, several text files run with shell commands that initialize the shell environment. To set shell or environment variables automatically when your shell starts, you can edit these Bash startup scripts.

The exact scripts that run depend on whether the shell is interactive or noninteractive, and is a login or a non-login shell.

A user directly enters commands into an interactive shell, whereas a noninteractive shell, such as a script, runs in the background without user intervention. For interactive login shells, the /etc/profile and ~/ .bash\_profile files configure the Bash environment. The /etc/profile file also sources the /etc/bashrc file, and the ~/ .bash\_profile file sources the ~/ .bashrc file. Noninteractive shells invoke any files that the BASH\_ENV variable defines. This variable is not defined by default.

A login shell is invoked when a user logs in locally via the terminal or remotely via the SSH protocol.

A non-login shell is invoked from within an existing login session, such as when you open a terminal from the GNOME GUI. For interactive non-login shells, only the /etc/bashrc and ~/ .bashrc files configure the Bash environment. Whereas the /etc/profile and /etc/bashrc files apply to the whole system, the ~/ .bash\_profile and ~/ .bashrc files are user-specific.

For example, to change the default editor when you log in via SSH, you can modify the EDITOR variable in your ~/ .bash\_profile file:

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs
export EDITOR=nano
```

### Important

The best way to adjust settings that affect all user accounts is to create a file with a `.sh` extension with the changes, and add this file to the `/etc/profile.d` directory. To create the files in the `/etc/profile.d` directory, log in as the `root` user.

## Bash Aliases

Bash aliases are shortcuts to other Bash commands. For example, if you often type a long command, then you can create a shorter alias to invoke it. Use the `alias` command to create aliases. Consider the following example, which creates a `hello` alias for an `echo` command.

```
user@host:~$ alias hello='echo "Hello, this is a long string."'
```

You can then run the `hello` command and it invokes the `echo` command.

```
user@host:~$ hello
Hello, this is a long string.
```

Add aliases to the user's `~/.bashrc` file so they are available in any interactive shell.

## Unsetting and Unexporting Variables and Aliases

To unset and unexport a variable, use the `unset` command:

```
user@host:~$ echo $file1  
/tmp/tmp.z9pXW0HqcC  
user@host:~$ unset file1  
user@host:~$ echo $file1  
  
user@host:~$
```

To unexport a variable without unsetting it, use the `export -n` command:

```
user@host:~$ export -n PS1
```

To unset an alias, use the `unalias` command:

```
user@host:~$ unalias hello
```

## References

`bash(1)`, `builtins(1)`, and `env(1)` man pages

[https://www.gnu.org/software/bash/manual/html\\_node/Bash-Variables.html](https://www.gnu.org/software/bash/manual/html_node/Bash-Variables.html) [Bash Reference Manual: Bash Variables]

[https://www.gnu.org/software/bash/manual/html\\_node/Arithmetic-Expansion.html](https://www.gnu.org/software/bash/manual/html_node/Arithmetic-Expansion.html) [Bash Reference Manual: Arithmetic Expansion]

## 1.2. Guided Exercise

### Change the Shell Environment

---

Set shell variables for use with commands, and export environment variables to modify the behavior of the shell.

#### Outcomes

- Edit a user profile.
- Create a shell variable.
- Create an environment variable.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start scripts-env
```

#### Instructions

1. Modify the PS1 shell variable on the servera machine to include the time by modifying the `~/.bashrc` configuration file to the '`[\u@\h \t \w]$`' value.

- 1.1. Log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 1.2. Edit the `~/.bashrc` configuration file to add the PS1 shell variable and its value. Set the value of the shell variable, including a trailing space, inside quotation marks.

```
...output omitted...
export PATH
PS1='[\u@\h \t \w]$ '
...output omitted...
```

- 1.3. Use the source command to load the modified `~/.bashrc` file.

```
[student@servera ~]$ source ~/.bashrc
[student@servera 14:45:05 ~]$
```

- 2.** Define a variable named `file` that contains the `examplefile` string, and then use the variable in the `ls` and `rm` commands.

- 2.1.** Define the `file` variable with a value of `examplefile`. The `examplefile` file exists in the student home directory.

```
[student@servera 14:47:05 ~]$ file=examplefile
```

- 2.2.** Retrieve the value of the `file` variable.

```
[student@servera 14:48:35 ~]$ echo $file
examplefile
```

- 2.3.** Use the `file` variable as the operand of the `ls -l` command to list files with `examplefile` as the name.

```
[student@servera 14:59:07 ~]$ ls -l $file
-rw-r--r--. 1 student student 0 Jan 23 14:59 examplefile
```

- 2.4.** Use the `rm` command with the `file` variable to delete the `examplefile` file.

```
[student@servera 14:59:10 ~]$ rm $file
```

- 2.5.** Verify that the `examplefile` file is not present.

```
[student@servera 14:59:15 ~]$ ls -l $file
ls: cannot access 'examplefile': No such file or directory
```

- 3.** Use the `export` command to set the `vim` editor to the `EDITOR` variable and export it as an environment variable.

- 3.1.** Export the `EDITOR` variable.

```
[student@servera 14:46:40 ~]$ export EDITOR=vi
```

- 3.2.** Verify the value of the `EDITOR` variable.

```
[student@servera 14:46:55 ~]$ echo $EDITOR  
vim
```

4. Return to the workstation system as the student user.

```
[student@servera 14:47:11 ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish scripts-env
```

## 1.3. Writing Simple Bash Scripts

### Objectives

- Create and execute a simple Bash shell script to run like a regular program to run predefined commands.

### Create and Execute Bash Shell Scripts

You can accomplish many system administration tasks by using command-line tools. More complex tasks often require chaining together multiple commands and passing the results between them. By using the Bash shell environment and scripting features, you can combine Linux commands into *shell scripts* to solve real-world problems.

A Bash shell script is a sequence of commands along with programming logic to control decision-making in the overall task. You can write shell scripts in the form of script files to handle repetitive tasks. When well-written, a shell script is a powerful command-line tool on its own, and you can use it with other scripts.

Shell scripting proficiency is essential for system administrators in any operational environment. You can use shell scripts to improve the efficiency and accuracy of routine task completion.

Although you can use any text editor, advanced editors such as `vim` or `emacs` understand Bash shell syntax and can provide color-coded highlighting. This highlighting helps to identify common scripting errors such as improper syntax, unmatched quotes, parentheses, brackets, and braces, and other structural mistakes.

### Specify the Command Interpreter

The first line of a script file begins with the `#!` notation, which is commonly referred to as shebang or hash-bang, from the names of those two characters, sharp or hash and bang.

This notation indicates which command interpreter to process the script under, when you execute the script directly.

For Bash syntax script files, the first line is the following directive:

```
#!/usr/bin/bash
```

This line indicates that the `/usr/bin/bash` executable file processes the remaining lines of the script. If the script is named `script.sh`, then the previous `#!` line directs your system to execute the equivalent of the following command:

```
user@host:~$ /usr/bin/bash script.sh
```

## Execute a Bash Shell Script

A shell script file must have execute permissions to run it as an ordinary command. Use the `chmod` command to modify the file permissions. Use the `chown` command, if needed, to grant execute permission only for specific users or groups.

If the script is stored in a directory that is listed in the shell's PATH environment variable, then you can run the shell script by using only its file name, similar to running compiled commands. The parsing of the PATH variable runs the first matching file name that is found. The name of the shell scripts must not match the command names that are present on the system. If a file exists within a directory that your PATH variable specifies, then you can determine its location by using the `which` command.

Alternatively, run a script in your current working directory by using the `.` directory prefix, such as `./scriptname`.

Use the `which` command to determine the location of the `hello` script:

```
user@host:~$ which hello
~/bin/hello
```

Verify that the `~/bin` directory is present in the PATH variable:

```
user@host:~$ echo $PATH
/home/user/.local/bin:/home/user/bin:/sbin:/bin:/usr/sbin:/usr/bin:...
```

## Positional Parameters

A *positional parameter* is a parameter that is indicated by one or more digits. You use positional parameters to pass text input to a shell script when it is invoked.

Positional parameters, like shell variables, are expanded with a dollar sign (\$) prefix. Use `$0` to refer to the command to run the shell script itself. Use `$1` to refer to the first parameter; use `$2` to refer to the

second parameter; and so on. When a positional parameter that consists of more than a single digit is expanded, it must be enclosed in braces, for example  `${10}` .

Positional parameters are not variables themselves, and thus they are not directly assignable in the same manner as variables. Positional parameters can, however, be reassigned in a single step by using the `set` builtin command.

The following Bash script named `params.sh` illustrates the use of positional parameters:

```
#!/usr/bin/bash

echo "Parameter 0: $0"
echo "Parameter 1: $1"
echo "Parameter 2: $2"
echo "Parameter 10 (incorrect): $10" ①
echo "Parameter 10 (correct): ${10}"

set -- "$1" "$2" "$4" "$3" ②

echo
echo "Parameter 3 (after set): $3"
echo "Parameter 4 (after set): $4"
echo "Parameter 10 (after set): ${10}"
```

- ① The positional parameter 10 is expanded incorrectly due to the lack of braces.
- ② The `set` builtin command is used to swap the values of the positional parameters 3 and 4. Note that positional parameters 5 through 10 are not referenced and are removed as a result.

The following command prints the output for corresponding positional parameters:

```
user@host:~$ ./params.sh 1st 2nd 3rd 4th 5th 6th 7th 8th 9th 10th
Parameter 0: ./params.sh ①
Parameter 1: 1st
Parameter 2: 2nd
Parameter 10 (incorrect): 1st②
Parameter 10 (correct): 10th ③

Parameter 3 (after set): 4th ④
Parameter 4 (after set): 3rd
Parameter 10 (after set): ⑤
```

- ① Positional parameter 0 represents the script exactly as it was called. If the script is called by using an absolute path, such as the /home/user/params.sh path, then the /home/user/params.sh path is printed instead.
- ② In this incorrect expansion of positional parameter 10, braces are not used. Bash interprets this parameter as positional parameter 1, followed by the number 0.
- ③ In this correct expansion of the positional parameter 10, braces are used. Bash interprets this parameter correctly as positional parameter 10.
- ④ The values of positional parameters 3 and 4 are reversed due to the swap that the set builtin command made.
- ⑤ The value of positional parameter 10 is empty because it was removed during the use of the set builtin command.

#### Note

Positional parameters are not necessarily equivalent to *arguments* or *options*. Each positional parameter can be enclosed in quotes and is separated by whitespace; however, such parameters can be complicated to parse.

You can use the getopt builtin command, with a case statement, to parse options along with their related arguments, such as -o yaml.

For more information, refer to the `builtins(1)` man page.

## Quote Special Characters

Some characters and words have a special meaning to the Bash shell. To use these characters for their literal values rather than for their special meanings, you escape them in the script. Use the backslash character (\), single quotation marks (' '), or double quotation marks (" ") to remove (or escape) the special meaning of these characters.

The backslash character removes the special meaning of the single character that immediately follows the backslash. For example, to use the echo command to display the # not a comment literal string, the # hash character must not be interpreted as a comment.

```
user@host:~$ echo # not a comment
```

The following example shows the backslash character (\) modifying the hash character so it is not interpreted as a comment:

```
user@host:~$ echo \# not a comment  
# not a comment
```

To escape more than one character in a text string, either use the backslash character multiple times, or enclose the whole string in single quotation marks (' ') to interpret literally.

```
user@host:~$ echo # not a comment #
```

The backslash character preserves one character in the following example:

```
user@host:~$ echo \# not a comment #  
# not a comment
```

The backslash character preserves two characters in the following example:

```
user@host:~$ echo \# not a comment \#  
# not a comment #
```

Single quotation marks preserve the literal meaning of all characters that they enclose, as in the following example:

```
user@host:~$ echo '# not a comment #'  
# not a comment #
```

Use the following example to parse the variables by using various characters:

```
user@host:~$ var=$(hostname -s); echo $var
host
```

Use double quotation marks to suppress *globbing* (file name pattern matching) and shell expansion, but still allow command and variable substitution.

```
user@host:~$ echo "***** hostname is ${var} *****"
***** hostname is host *****
```

Variable substitution is conceptually similar to command substitution, and might use optional brace syntax. The question mark (?) is included inside the quotations, because it is a *metacharacter* that also needs escaping from expansion.

```
user@host:~$ echo "Will variable $var evaluate to $(hostname -s)?"
Will variable host evaluate to host?
```

Use double quotation marks to make the backslash character inoperative:

```
user@host:~$ echo \"Hello, world\"
"Hello, world"
```

Use single quotation marks to interpret all enclosed text literally:

```
user@host:~$ echo 'Hello, world'
"Hello, world"
```

Besides suppressing globbing and shell expansion, single quotation marks also direct the shell to suppress command and variable substitution. The question mark (?) is included inside the quotations, because it is a *metacharacter* that also needs escaping from expansion.

```
user@host:~$ echo 'Will variable $var evaluate to $(hostname -s)?'
Will variable $var evaluate to $(hostname -s)?
```

## Provide Output from a Shell Script

The echo command displays arbitrary text by passing the text as an argument to the command. By default, the text is sent to *standard output* (STDOUT). You can send text elsewhere by using output redirection.

In the following simple Bash script, the echo command displays the "Hello, world" message:

```
user@host:~$ cat ~/bin/hello
#!/usr/bin/bash

echo "Hello, world"
```

When you run the script, the output redirects to STDOUT, which defaults to the screen device:

```
user@host:~$ hello
Hello, world
```

The echo command is widely used in shell scripts to display informational or error messages. Messages help indicate a script's progress, and can be directed to standard output or standard error, or be redirected to a log file for archiving. When you display error messages, good programming practice is to redirect error messages to STDERR to separate them from normal program output.

In the following example, the hello script redirects STDERR:

```
user@host:~$ cat ~/bin/hello
#!/usr/bin/bash

echo "Hello, world"
echo "ERROR: Houston, we have a problem." >&2
```

In the following example, the hello script error message is redirected to the hello.log file:

```
user@host:~$ hello 2> hello.log
Hello, world
```

View the contents of the hello.log file.

```
user@host:~$ cat hello.log
ERROR: Houston, we have a problem.
```

The echo command is also helpful to debug a problematic shell script. Adding echo statements in a script, to display variable values and other runtime information, can help to clarify how a script is functioning.

## References

`bash(1)`, `builtins(1)`, `echo(1)`, `echo(1p)`, and `getopts(1p)` man pages

[Adding Arguments and Options to Your Bash Scripts](#)

## 1.4. Guided Exercise

### Write Simple Bash Scripts

Create and execute a simple Bash shell script to run like a regular program and redirect the output to a file.

#### Outcomes

- Write and execute a simple Bash script.
- Redirect the output of a simple Bash script to a file.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start scripts-write
```

#### Instructions

1. Log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Create and execute the `firstscript.sh` Bash script.

1. Use a text editor to create the `firstscript.sh` file in your home directory.

```
#!/usr/bin/bash
echo "This is my first bash script" > ~/output.txt
echo "" >> ~/output.txt
echo ##### >> ~/output.txt
```

2. Make the `firstscript.sh` file executable by using the `chmod` command.

```
[student@servera ~]$ chmod +x firstscript.sh
```

3. Execute the `firstscript.sh` script.

```
[student@servera ~]$ ./firstscript.sh
```

**2.4.** Review the output file that the script generates.

```
[student@servera ~]$ cat output.txt
This is my first bash script

#####
```

**3.** Add more commands to the `firstscript.sh` script, execute it, and review the output.

**3.1.** Use a text editor to modify the `firstscript.sh` file to match the following script:

```
#!/usr/bin/bash
#
echo "This is my first bash script" > ~/output.txt
echo "" >> ~/output.txt
echo "#####" >> ~/output.txt
echo "LIST BLOCK DEVICES" >> ~/output.txt
echo "" >> ~/output.txt
lsblk >> ~/output.txt
echo "" >> ~/output.txt
echo "#####" >> ~/output.txt
echo "FILESYSTEM FREE SPACE STATUS" >> ~/output.txt
echo "" >> ~/output.txt
df -h >> ~/output.txt
echo "#####" >> ~/output.txt
```

**3.2.** Execute the `firstscript.sh` script.

```
[student@servera ~]$ ./firstscript.sh
```

**3.3.** Review the output file that the script generated.

```
[student@servera ~]$ cat output.txt
This is my first bash script

#####
LIST BLOCK DEVICES

NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   10G  0 disk
└─sda1   8:1    0   1M   0 part
└─sda2   8:2    0  200M  0 part /boot/efi
└─sda3   8:3    0  9.8G  0 part /
sdb      8:16   0   5G   0 disk
sdc      8:32   0   5G   0 disk
sdd      8:48   0   5G   0 disk
sr0     11:0   1  514K  0 rom

#####
FILESYSTEM FREE SPACE STATUS

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3       9.8G  3.1G  6.7G  32% /
devtmpfs        4.0M   0    4.0M   0% /dev
tmpfs           981M   0   981M   0% /dev/shm
tmpfs           393M   14M  379M   4% /run
tmpfs           1.0M   0   1.0M   0% /run/credentials/systemd...
/dev/sda2       200M  8.4M  192M   5% /boot/efi
tmpfs           1.0M   0   1.0M   0% /run/credentials/getty@...
tmpfs           1.0M   0   1.0M   0% /run/credentials/serial...
tmpfs           197M  4.0K  197M   1% /run/user/1000
#####
```

#### 4. Return to the workstation machine when finished.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish scripts-write
```

## 1.5. Running Loops and Conditional Commands

### Objectives

- Describe Bash control structures to run a command in a loop, to run different tasks based on the value of a variable, or that run depending on whether a condition is true or false.

### Use Loops to Iterate Commands

System administrators often encounter repetitive tasks in their daily activities. A repetitive task example is running a command multiple times on a target, such as checking a process every minute for 10 minutes to know whether it completed. Another example is running a command once each for multiple targets, such as backing up many databases on a system. The *for loop* is a Bash looping construct to use for task iterations.

#### Process Items from the Command Line

In Bash, the *for* loop construct uses the following syntax:

```
for VARIABLE in LIST; do
    COMMAND VARIABLE
done
```

The loop processes the strings that you provide in *LIST*, and exits after processing the last string in the list. The *for* loop temporarily stores each list string as the value of *VARIABLE*, and then executes the block of commands that use the variable. The variable name is arbitrary. Typically, you reference the variable value with commands in the command block.

Provide the list of strings for the *for* loop from a list that the user enters directly, or that is generated from shell expansion, such as variable, brace, or file name expansion, or command substitution.

The following example uses the *for* loop to print the value of the *HOST* variable:

```
user@host:~$ for HOST in host1 host2 host3; do echo $HOST; done
host1
host2
host3
```

The following example uses the *for* loop to print the value of the *HOST* variable by using braces ({}):

```
user@host:~$ for HOST in host{1,2,3}; do echo $HOST; done
host1
host2
host3
```

The following example uses the `for` loop to print the value of the `HOST` variable by using the range of values in braces:

```
user@host:~$ for HOST in host{1..3}; do echo $HOST; done
host1
host2
host3
```

Another example uses the `for` loop to print the value of the `HOST` variable by using the range of values in braces:

```
user@host:~$ for FILE in file{a..c}; do ls $FILE; done
filea
fileb
filec
```

Use the `for` loop to print the installed kernel packages along with the date in a string:

```
user@host:~$ for PACKAGE in $(rpm -qa | grep kernel); \
do echo "$PACKAGE was installed on \
$(date -d @$($rpm -q --qf "%{INSTALLTIME}\n" $PACKAGE))"; done
kernel-tools-libs-5.14.0-70.2.1.el9_0.x86_64 was installed on Thu...
kernel-tools-5.14.0-70.2.1.el9_0.x86_64 was installed on Thu Mar...
kernel-core-5.14.0-70.2.1.el9_0.x86_64 was installed on Thu Mar...
kernel-modules-5.14.0-70.2.1.el9_0.x86_64 was installed on Thu...
kernel-5.14.0-70.2.1.el9_0.x86_64 was installed on Thu Mar 24...
```

Use the `for` loop to print even numbers from 1 through 10:

```
user@host:~$ for EVEN in $(seq 2 2 10); do echo "$EVEN"; done
2
4
6
8
10
```

## Conditional Commands

Simple shell scripts represent a collection of commands that are executed from beginning to end. Programmers incorporate decision-making into shell scripts by using conditional commands. A script can execute specific routines when stated conditions are met.

### Bash Script Exit Codes

An *exit code* is a value from 0 to 255 which indicates how a process exited. All commands produce an exit code on completion.

After a script interprets and processes all its content, the script process exits with the 0 exit code and passes control back to the parent process that called it. However, a script can exit before it finishes processing all commands, such as when the script encounters an error condition. Use the `exit` command, along with an optional exit code, to immediately leave the script and skip processing the remainder of the script, and returning the exit code to the shell.

An exit code value of 0 represents a successful script completion with no errors. All other nonzero values indicate an error exit code. The `errno -l` command provides a list of standardized error codes and their meanings. You can retrieve the exit code of the most recent command from the built-in `$?` variable.

The following script returns exit code 0 on successful execution:

```
[user@host bin]$ cat hello
#!/usr/bin/bash
echo "Hello, world"
exit 0
```

After executing the script, print the `$?` variable value to determine the exit code:

```
[user@host bin]$ echo $?
0
```

The `/bin/true` command always returns the 0 exit code.

```
[user@host bin]$ /bin/true
[user@host bin]$ echo $?
0
```

The `/bin/false` command always returns the 1 exit code.

```
[user@host bin]$ /bin/false  
[user@host bin]$ echo $?  
1
```

When a script's exit command is used without an exit code argument, the script returns the exit code of the last command that was run within the script.

## Test for Conditions

You can test whether conditional statements are true by using the `test` command or the `[[ CONDITION ]]` syntax. Create conditional statements by comparing values with operators such as greater than (`-gt`), less than (`-lt`), equal (`-eq`), or not equal (`-ne`). Additionally, you can verify other statuses, such as whether a regular file (`-f`) or directory (`-d`) exists or whether the current user has read permission (`-r`).

### Note

Shell scripting uses many other operators. The `test(1)` man page lists all conditional expression operators with descriptions. The `bash(1)` man page also explains operator use and evaluation, although it can be complex to read. Red Hat recommends learning shell scripting through quality books and courses that are dedicated to shell programming.

The following examples demonstrate the `test` command with Bash numeric comparison operators.

The following example runs successfully and prints the exit code as `0`.

```
user@host:~$ test 1 -gt 0 ; echo $?  
0
```

The following example runs unsuccessfully and prints the exit code as `1`.

```
user@host:~$ test 0 -gt 1 ; echo $?  
1
```

The following examples demonstrate the Bash `test` command syntax and numeric comparison operators:

```
user@host:~$ [[ 1 -eq 1 ]]; echo $? ①
0
user@host:~$ [[ 1 -ne 1 ]]; echo $? ②
1
user@host:~$ [[ 8 -gt 2 ]]; echo $? ③
0
user@host:~$ [[ 2 -ge 2 ]]; echo $? ④
0
user@host:~$ [[ 2 -lt 2 ]]; echo $? ⑤
1
```

- ① Determines whether the operands are equal.
- ② Determines whether the operands are not equal.
- ③ Determines that the left operand is greater than the right operand.
- ④ Determines that the left operand is greater than or equal to the right operand.
- ⑤ Determines that the left operand is less than the right operand.

The following examples demonstrate the Bash string comparison operators:

```
user@host:~$ [[ abc = abc ]]; echo $? ①
0
user@host:~$ [[ abc == def ]]; echo $? ②
1
user@host:~$ [[ abc != def ]]; echo $? ③
0
```

- ① Determines whether the operands are equal.
- ② Determines whether the operands are equal to match case sensitivity.
- ③ Determines whether the operands are not equal.

The following examples demonstrate Bash string unary (one argument) operators:

```
user@host:~$ STRING=''; [[ -z "$STRING" ]]; echo $?
0
user@host:~$ STRING='abc'; [[ -n "$STRING" ]]; echo $?
0
```

### Note

The space characters inside the brackets are mandatory, because they separate the words and elements within the test expression. The shell's command parsing routine divides the command elements into words and operators by recognizing spaces and other metacharacters, according to built-in parsing rules. For full treatment of this advanced concept, see the getopt(3) man page. The left square bracket character ([ ]) is itself a built-in alias for the test command. Shell words, whether they are commands, subcommands, options, arguments, or other token elements, are always delimited by spaces.

## While Loops and Until Loops

While loops and until loops present a way to continue performing a set of commands provided that a condition is met. A while loop executes the body of the loop as long as the condition remains true (a zero exit code). However, an until loop executes as long as the condition remains false (a non-zero exit code). Both while loops and until loops test that the condition is true before performing any iterations in the loop. If the condition prevents an iteration of the loop before the while or until statement, then the loop body does not execute.

The `while(test)/do/done` construct has the following syntax:

```
while <CONDITION>; do
    <STATEMENT>
    ...
    <STATEMENT>
done
```

With this construct, provided that the script meets the given condition, it executes the code in the statement block. Common test conditions in the `while(test)/do/done` statements include the previously discussed numeric, string, and file tests. The done statement at the end closes the `while(test)/do/done` construct.

The following code section demonstrates a `while(test)/do/done` construct to print the value of the variable `x` (which is initialized to 1) and to increment `x` when it is less than or equal to 5:

```
user@host:~$ x=1; while [[ $x -le 5 ]]; do
    echo "x is $x"
    x=$((x + 1))
done; echo "Finished"
x is 1
x is 2
x is 3
x is 4
x is 5
Finished
```

The *until(test)/do/done* construct has the following syntax:

```
until <CONDITION>; do
    <STATEMENT>
    ...
    <STATEMENT>
done
```

With this construct, as long as the script does not meet the given condition, then it executes the code in the statement block. Common test conditions in the *until(test)/do/done* statements include the previously discussed numeric, string, and file tests. The *done* statement at the end closes the *until(test)/do/done* construct.

The following code section demonstrates an *until(test)/do/done* construct to print the value of the variable *x* (which is initialized to 5) and to decrement *x* until *x* is less than 1:

```
user@host:~$ x=5; until [[ $x -lt 1 ]]; do
    echo "x is $x"
    x=$((x - 1))
done; echo "Finished"
x is 5
x is 4
x is 3
x is 2
x is 1
Finished
```

The exit code of the *while* and *until* commands is the exit code of the last executed statement. The exit code is zero if no statement was executed.

## If Then Conditionals

The simplest conditional structure is the *if/then* construct, with the following syntax:

```
if <CONDITION>; then
    <STATEMENT>
    ...
    <STATEMENT>
fi
```

With this construct, if the script meets the given condition, then it executes the code in the statement block. The script does not act if the given condition is not met. Common test conditions in the *if/then* statements include the previously discussed numeric, string, and file tests. The *fi* statement at the end closes the *if/then* construct.

The following code section demonstrates an *if/then* construct to start the *psacct* service if it is not active:

```
user@host:~$ systemctl is-active psacct > /dev/null 2>&1
user@host:~$ if [[ $? -ne 0 ]]; then sudo systemctl start psacct; fi
```

## Else Statements

You can further expand the *if/then* construct to take different sets of actions depending on whether a condition is met. Use the *if/then/else* construct to accomplish this behavior, as in this example:

```
if <CONDITION>; then
    <STATEMENT>
    ...
    <STATEMENT>
else
    <STATEMENT>
    ...
    <STATEMENT>
fi
```

The following code section demonstrates an *if/then/else* statement to start the *psacct* service if it is not active, and to stop this service if it is active:

```
user@host:~$ systemctl is-active psacct > /dev/null 2>&1
user@host:~$ if [[ $? -ne 0 ]]; then
    sudo systemctl start psacct
else
    sudo systemctl stop psacct
fi
```

## Elif Statements

You can also expand an `if/then/else` construct to test more than one condition and to execute a different set of actions when the construct meets a specific condition.

The next example shows the construct for an added condition:

```
if <CONDITION>; then
    <STATEMENT>
    ...
    <STATEMENT>
elif <CONDITION>; then
    <STATEMENT>
    ...
    <STATEMENT>
else
    <STATEMENT>
    ...
    <STATEMENT>
fi
```

In this conditional structure, Bash tests the conditions as they are ordered in the script. When a condition is true, Bash executes the actions that are associated with the condition and then skips the remainder of the conditional structure. If none of the conditions are true, then Bash executes the actions in the `else` clause.

The following example demonstrates an `if/then/elif/then/else` statement to run the `mysql` client if the `mariadb` service is active, or to run the `psql` client if the `postgresql` service is active, or to run the `sqlite3` client if both the `mariadb` and the `postgresql` service are inactive:

```
user@host:~$ systemctl is-active mariadb > /dev/null 2>&1
user@host:~$ MARIADB_ACTIVE=$?
user@host:~$ sudo systemctl is-active postgresql > /dev/null 2>&1
user@host:~$ POSTGRESQL_ACTIVE=$?
user@host:~$ if [[ "$MARIADB_ACTIVE" -eq 0 ]]; then
    mysql
elif [[ "$POSTGRESQL_ACTIVE" -eq 0 ]]; then
    psql
else
    sqlite3
fi
```

## References

[bash\(1\) man page](#)

## 1.6. Guided Exercise

### Run Loops and Conditional Commands

---

Use a for loop and if/then/else Bash control structures to run a command with different tasks.

#### Outcomes

- Create a for loop to iterate through a list of items from the command line and in a shell script.
- Create an if/then/else statement that executes different commands based on a condition being true or false.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start scripts-loops
```

#### Instructions

1. Use the ssh and hostname commands to print the hostname of the servera machine and the serverb machine to standard output.

- 1.1. Print the hostname of the servera machine.

```
student@workstation:~$ ssh student@servera hostname  
servera
```

- 1.2. Print the hostname of the serverb machine.

```
student@workstation:~$ ssh student@serverb hostname  
serverb
```

2. Create a for loop to execute the hostname command on the servera and serverb machines.

```
student@workstation:~$ for host in servera serverb; do
    ssh student@${host} hostname
done
servera
serverb
```

- 3.** Create a shell script in the /home/student/bin directory to execute the same for loop. Insert an if/then/else statement at the end of the loop that prints the message, The host is servera if the \$HOST variable is equal to servera, or otherwise print the message, The host is not servera. Ensure that the /home/student/bin directory is included in the PATH environment variable.

- 3.1.** Create the /home/student/bin directory to store the shell script, if the directory does not exist.

```
student@workstation:~$ mkdir ~/bin
```

- 3.2.** Verify that the bin subdirectory of your home directory is in your PATH environment variable.

```
student@workstation:~$ echo $PATH
/home/student/.local/bin:`/home/student/bin`:/sbin:/bin:/usr/sbin:...
```

- 3.3.** Use a text editor to create a shell script named printhostname.sh in the /home/student/bin directory to perform the for loop, and add the following content in the file:

```
#!/usr/bin/bash
# Execute for loop to print server hostname.
for host in servera serverb; do
    ssh student@${host} hostname
    if [[ "${host}" == "servera" ]]; then
        echo "The host is servera"
    else
        echo "The host is not servera"
    fi
done
```

- 3.4.** Give the script execute permissions.

```
student@workstation:~$ chmod +x ~/bin/printhostname.sh
```

**3.5.** Run the script from your home directory.

```
student@workstation:~$ ./printhostname.sh  
servera  
The host is servera  
serverb  
The host is not servera
```

**3.6.** Verify that the exit code of your script is 0.

```
student@workstation:~$ echo $?  
0
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish scripts-loops
```

## 1.7. Lab

# Shell Scripting and the Command Line

Write and run a simple shell script to use a Bash control structure with a for loop.

## Outcomes

- Create a Bash script and redirect its output to a file.
- Use loops to simplify your code.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start scripts-review
```

## Instructions

1. Create the executable /home/student/bin/bash-lab.sh script file on the workstation machine. Add a #! directive to execute under the /bin/bash interpreter. The script must be empty besides the #! directive.
2. Edit your newly created script file to store the following information from the servera and serverb machines on the workstation machine. Store the output of the listed commands from the following table in the /home/student/output-servera and /home/student/output-serverb files respectively on the workstation machine. Print a sequence of hash signs (#) to differentiate the output of the successive commands in the output file.

Command	Notes
hostname --fqdn	Store the entire output.
echo "#####"	Append the hash signs to differentiate the following command.

Command	Notes
hostname --ip-address	Store the entire output.
echo "#####"	Append the hash signs to differentiate the following command.
uname --machine	Store the entire output.

Save the required information to the `output-servera` and `output-serverb` files in the `/home/student` directory on workstation.

#### Note

Remember to use a loop to simplify your script. You can use shell variables to store the output file name and SSH destination.

3. Execute the `/home/student/bin/bash-lab.sh` script and review the output content on the workstation machine. The `/home/student/bin` directory is already included in the student user's PATH environment variable.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade scripts-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish scripts-review
```

## Solution

---

Write and run a simple shell script to use a Bash control structure with a for loop.

## Outcomes

- Create a Bash script and redirect its output to a file.
- Use loops to simplify your code.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start scripts-review
```

## Instructions

1. Create the executable /home/student/bin/bash-lab.sh script file on the workstation machine. Add a #! directive to execute under the /bin/bash interpreter. The script must be empty besides the #! directive.

- 1.1. On the workstation machine, create the /home/student/bin/ directory if it does not exist.

```
student@workstation:~$ mkdir -p /home/student/bin
```

- 1.2. Use a text editor to create and edit the /home/student/bin/bash-lab.sh script file, and add the following content:

```
#!/usr/bin/bash
```

- 1.3. Make your script file executable by using the chmod command.

```
student@workstation:~$ chmod +x ~/bin/bash-lab.sh
```

2. Edit your newly created script file to store the following information from the servera and serverb machines on the workstation machine. Store the output of the listed commands from the following table in the /home/student/output-servera and /home/student/output-

serverb files respectively on the workstation machine. Print a sequence of hash signs (#) to differentiate the output of the successive commands in the output file.

Command	Notes
hostname --fqdn	Store the entire output.
echo #####	Append the hash signs to differentiate the following command.
hostname --ip-address	Store the entire output.
echo #####	Append the hash signs to differentiate the following command.
uname --machine	Store the entire output.

Save the required information to the output-servera and output-serverb files in the /home/student directory on workstation.

#### Note

Remember to use a loop to simplify your script. You can use shell variables to store the output file name and SSH destination.

- 2.1.** Use a text editor to append the following lines to the /home/student/bin/bash-lab.sh script file. The number of hash signs is arbitrary.

#### Note

The following output is an example of how you can achieve the requested script. In Bash scripting, you can take different approaches and obtain the same result.

```
#!/usr/bin/bash
USER='student'
PREFIX='/home/student/output'

for SERVER in servera serverb; do
    OUTPUT=${PREFIX}-${SERVER}
    DESTINATION=${USER}@${SERVER}

    ssh ${DESTINATION} "hostname --fqdn" > ${OUTPUT}
    echo ##### >> ${OUTPUT}
    ssh ${DESTINATION} "hostname --ip-address" >> ${OUTPUT}
    echo ##### >> ${OUTPUT}
    ssh ${DESTINATION} "uname --machine" >> ${OUTPUT}
done
```

- 3.** Execute the /home/student/bin/bash-lab.sh script and review the output content on the workstation machine. The /home/student/bin directory is already included in the student user's PATH environment variable.

- 3.1.** On the workstation machine, execute the /home/student/bin/bash-lab.sh script without the bash command and without a path.

```
student@workstation:~$ bash-lab.sh
...output omitted...
```

- 3.2.** Review the content of the /home/student/output-servera file.

```
student@workstation:~$ cat /home/student/output-servera
servera.lab.example.com
#####
172.25.250.10
#####
x86_64
```

- 3.3.** Review the content of the /home/student/output-serverb file.

```
student@workstation:~$ cat /home/student/output-serverb
serverb.lab.example.com
#####
172.25.250.11
#####
x86_64
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade scripts-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish scripts-review
```

## 1.8. Summary

---

In this lesson, you learned how to use shell variables, bash scripts, loops, and conditional commands.

Shell variables can help you to run commands, and are unique to a shell session. You can modify the behavior of the shell or the processes with environment variables.

You can create and execute Bash scripts to accomplish administration tasks. Loops are used to iterate through a list of items from the command line and in a shell script. Conditional structures are used to incorporate decision-making into shell scripts.

## Chapter 2.

# Using Regular Expressions for Practical Applications

---

## Goal

Efficiently complete system administration tasks by using regular expressions to match text.

## Sections

- Matching Text with Regular Expressions (and Guided Exercise)
- Matching Text with Regular Expressions (Quiz)

## 2.1. Matching Text with Regular Expressions

---

### Objectives

- Create regular expressions to use with command-line utilities, to match lines in a file or in the output from a program or a shell pipeline.

### Write Regular Expressions

Regular expressions provide a pattern matching mechanism to find specific content. The `grep`, `less`, and `vim` commands are some of the commands that can use regular expressions.

Programming languages such as C, Python, and Rust also support regular expressions, and might differ slightly in syntax.

Regular expressions are a unique language, with their own syntax and rules. This section introduces regular expression syntax as implemented in Bash, with examples.

### A Simple Regular Expression

The simplest regular expression is an exact match of the string to search. An exact match occurs when the characters in the regular expression match the type and order of the string.

Imagine that a user is looking through the following file for all occurrences of the `cat` pattern:

```
cat
dog
concatenate
dogma
category
educated
boondoggle
vindication
chilidog
```

The `cat` string is an exact match of the c character, followed by the a and t characters with no other characters in between. Searching the file with the `cat` string as the regular expression returns the following matches:

```
cat
concatenate
category
educated
vindication
```

## Match the Start and End of a Line

A regular expression matches the search string anywhere on the line in which it occurs: at the beginning, middle, or end of the word or line. Use a *line anchor* metacharacter to control where to look for a match on a line.

To match only at the beginning of a line, use the caret character (^). To match only at the end of a line, use the dollar sign (\$).

Using the same file as in the previous example, the ^cat regular expression matches two lines.

```
cat
category
```

The cat\$ regular expression finds only one match, where the cat characters occur at the end of a line.

```
cat
```

Locate lines in the file that end with dog by using an end-of-line anchor to create the dog\$ regular expression. This example matches two lines:

```
dog
chilidog
```

To locate a line that contains only the exact search expression, use both the beginning and end-of-line anchors. For example, to locate the word cat when it is both at the beginning and at the end of a line simultaneously, use ^cat\$.

```
cat
```

## Basic and Extended Regular Expressions

The two types of regular expressions are basic regular expressions and extended regular expressions.

One difference between basic and extended regular expressions is in the behavior of the |, +, ?, (, ), {, and } special characters. In basic regular expression syntax, these characters have a special meaning *only* if they are prefixed with a backslash \ character. In extended regular expression syntax, it is exactly the opposite: these characters have a special meaning *unless* they are prefixed with a backslash character.

Other minor differences apply to how the ^, \$, and \* characters are handled.

The grep, sed, and vim commands use basic regular expressions. The grep command -E option, the sed command -E option, and the less command use extended regular expressions.

## Wildcard and Multiplier Usage in Regular Expressions

Regular expressions use a period character (.) as a wildcard to match any single character on a single line. The c.t regular expression searches for a string that contains c, followed by any single character, followed by t.

Example matches might include cat, concatenate, vindication, cut, and c\$t.

With an unrestricted wildcard, you cannot predict the character that matches the wildcard. To match specific characters, replace the unrestricted wildcard with appropriate characters.

The use of bracket characters, such as in the c[aou]t regular expression, matches patterns that start with c, followed by a, o, or u, followed by t. Possible matching expressions can have the cat, cot, or cut strings.

*Multipliers* are another mechanism for use as a wildcard. Multipliers apply to the previous character or wildcard in the regular expression. An often used multiplier is the asterisk (\*) character. When used in a regular expression, the asterisk multiplier matches zero or more occurrences of the multiplied expression. You can use the asterisk with expressions, in addition to characters.

For example, the c[aou]\*t regular expression might match coat or coot. A regular expression of c.\*t matches cat, coat, culvert, and even ct (matching zero characters between c and t). Any string that starts with c, is followed by zero or more characters, and ends with t must be a match.

Another type of multiplier indicates a more precise number of characters in the pattern. An example of an explicit multiplier is the 'c.{2}t' regular expression, which matches any word that begins with c, followed by exactly any two characters, and ends with t. The 'c.{2}t' expression would match two words in the following example:

```
cat
coat
convert
cart
covert
cypher
```

### Note

This course introduced two metacharacter text parsing mechanisms: shell *pattern matching* (which is also known as *file globbing* or *file name expansion*), and *regular expressions*. Both mechanisms use similar metacharacters, such as the asterisk character, with differences in metacharacter interpretation and rules.

Pattern matching is a shell technique to specify multiple file names on the command line. Regular expressions represent any form or pattern in text strings, no matter how complex. Regular expressions are internally supported by many text processing commands and languages such as awk, grep, python, rust, sed, and many other applications.

Table 6. Basic and Extended Regular Expression Syntax

Basic syntax	Extended syntax	Description
.		The period (.) matches any single character.
?		The preceding item is optional and is matched at most once.
*		The preceding item is matched zero or more times.
+		The preceding item is matched one or more times.
\{n\}	{n}	The preceding item is matched exactly n times.
\{n,\}	{n,}	The preceding item is matched n or more times.

Basic syntax	Extended syntax	Description
\{,m\}	{,m}	The preceding item is matched at most m times.
\{n,m\}	{n,m}	The preceding item is matched at least n times, but not more than m times.
[:alnum:]		Alphanumeric characters: [:alpha:] and [:digit:]; in the 'C' locale and ASCII character encoding, this expression is the same as [0-9A-Za-z].
[:alpha:]		Alphabetic characters: [:lower:] and [:upper:]; in the 'C' locale and ASCII character encoding, this expression is the same as [A-Za-z].
[:blank:]		Blank characters: space and tab.
[:cntrl:]		Control characters. In ASCII, these characters have octal codes 000 through 037, and 177 (DEL).
[:digit:]		Digits: 0 1 2 3 4 5 6 7 8 9.
[:graph:]		Graphical characters: [:alnum:] and [:punct:].
[:lower:]		Lowercase letters; in the 'C' locale and ASCII character encoding: a b c d e f g h i j k l m n o p q r s t u v w x y z.
[:print:]		Printable characters: [:alnum:], [:punct:], and space.

Basic syntax	Extended syntax	Description
[:punct:]		Punctuation characters; in the 'C' locale and ASCII character encoding: ! " # \$ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ' {   } ~.
[:space:]		Space characters: in the 'C' locale, it is tab, newline, vertical tab, form feed, carriage return, and space.
[:upper:]		Uppercase letters: in the 'C' locale and ASCII character encoding, it is: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z.
[:xdigit:]		Hexadecimal digits: 0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f.
\b		Match the empty string at the start or end of a word.
\B		Match the empty string provided that it is not at start or end of a word.
\<		Match the empty string at the start of a word.
\>		Match the empty string at the end of a word.
\w		Match the word constituent. Synonym for [_[:alnum:]]
\W		Match the non-word constituent. Synonym for [^_[:alnum:]]

Basic syntax	Extended syntax	Description
\s		Match white space. Synonym for [[:space:]]
\S		Match non-white space. Synonym for [^[:space:]]

## Match Regular Expressions from the Command Line

The grep command uses regular expressions to isolate matching data. You can use the grep command to match data in a single file or in multiple files. When you use grep to match data in multiple files, it prints the file name followed by a colon character and then the lines that match the regular expression.

### Isolate Data with the Grep Command

The grep command specifies a regular expression and a file to parse for matches.

```
user@host:~$ grep '^computer' /usr/share/dict/words
computer
computerese
computerise
computerite
computerizable
computerization
computerize
computerized
computerizes
computerizing
computerlike
computernik
computers
```

#### Note

It is recommended practice to use single quotation marks to encapsulate the regular expression to protect any shell metacharacters (such as the \$, \*, and { } characters). Encapsulating the regular expression ensures that the command, not the shell, interprets the characters.

The grep command can process output from other commands by using a pipe operator character (|). The following example shows the grep command parsing lines from the output of another command.

```
root@host:~# ps aux | grep chrony
chrony 915  0.0  0.0 10140  2516 ?      S  20:19 0:00 /usr/sbin/chrony -F 2
root    2297  0.0  0.0 228220 1936 pts/1 S+ 20:24 0:00 grep --color=auto chrony
```

## Grep Command Options

The grep command has many options for controlling how it parses lines.

Table 7. Table of Common Grep Options

Option	Function
-i	Use the provided regular expression and do not enforce case sensitivity (run case-insensitive).
-v	Display only lines that do <i>not</i> contain matches to the regular expression.
-r	Search for data that matches the regular expression recursively in a group of files or directories.
-A NUMBER	Display NUMBER of lines after the regular expression match.
-B NUMBER	Display NUMBER of lines before the regular expression match.
-e	If multiple -e options are used, then multiple regular expressions can be supplied and are used with a logical OR.
-E	Use extended instead of basic regular expression syntax when parsing the provided regular expression.

View the man pages to find other options for the grep command.

## Examples of the Grep Command

The following examples use various configuration files and log files.

Regular expressions are case-sensitive by default. Use the grep command `-i` option to run a case-insensitive search. The following example shows an excerpt of the `/etc/httpd/conf/httpd.conf` configuration file:

```
user@host:~$ cat /etc/httpd/conf/httpd.conf
...output omitted...
ServerRoot "/etc/httpd"

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on a specific IP address, but note that if
# httpd.service is enabled to run at boot time, the address may not be
# available when the service starts. See the httpd.service(8) man
# page for more information.
#
#Listen 12.34.56.78:80
Listen 80
...output omitted...
```

The following example searches for the `serverroot` regular expression in the `/etc/httpd/conf/httpd.conf` configuration file.

```
user@host:~$ grep -i serverroot /etc/httpd/conf/httpd.conf
# with "/", the value of ServerRoot is prepended -- so 'log/access_log'
# with ServerRoot set to '/www' will be interpreted by the
# ServerRoot: The top of the directory tree under which the server's
# ServerRoot at a non-local disk, be sure to specify a local disk on the
# same ServerRoot for multiple httpd daemons, you will need to change at
ServerRoot "/etc/httpd"
```

Use the grep command `-v` option to *reverse* search the regular expression. This option displays only the lines that do *not* match the regular expression.

In the following example, all lines, regardless of case, that *do not* contain the `server` regular expression are returned.

```
user@host:~$ grep -v -i server /etc/hosts
127.0.0.1 localhost.localdomain localhost
172.25.254.254 classroom.example.com classroom
172.25.254.254 content.example.com content
172.25.254.254 materials.example.com materials
### rht-vm-hosts file listing the entries to be appended to /etc/hosts

172.25.250.9    workstation.lab.example.com workstation
172.25.250.254  bastion.lab.example.com bastion
172.25.250.220  utility.lab.example.com utility
172.25.250.220  registry.lab.example.com registry
```

To view a file without the distraction of comment lines, use the grep command -v option. In the following example, the regular expression matches and excludes all the lines that begin with a hash character (#) or a semicolon (;) in the /etc/systemd/system/multi-user.target.wants/rsyslog.service file. In that file, the hash character at the beginning of a line indicates a general comment, whereas the semicolon refers to a commented variable value.

```
user@host:~$ grep -v '^#[;]' \
/etc/systemd/system/multi-user.target.wants/rsyslog.service
[Unit]
Description=System Logging Service
Wants=network.target network-online.target
After=network.target network-online.target
Documentation=man:rsyslogd(8)
Documentation=https://www.rsyslog.com/doc/

[Service]
Type=notify
EnvironmentFile=-/etc/sysconfig/rsyslog
ExecStart=/usr/sbin/rsyslogd -n $SYSLOGD_OPTIONS
ExecReload=/usr/bin/kill -HUP $MAINPID
UMask=0066
StandardOutput=null
Restart=on-failure

...output omitted...

LimitNOFILE=16384

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

The grep command -e option can search for more than one regular expression at a time. The following example, which uses a combination of the less and grep commands, locates all occurrences of pam\_unix, user root, and Accepted publickey in the /var/log/secure log file.

```
root@host:~# grep -e 'pam_unix' -e 'user root' -e 'Accepted publickey' \
/var/log/secure | less
May 19 19:29:09 host passwd[3108]: pam_unix(passwd:chauthtok): password changed
for root
May 19 19:30:07 host sshd-session[5941]: Accepted publickey for devops from
10.80.0.254 port 50198 ssh2: RSA
SHA256:30Iqz4c4G034GexgXwfkmfxpeRa/wYsXxQxzY0xGeH0
May 19 19:30:07 host (systemd)[5946]: pam_unix(systemd-user:session): session
opened for user devops(uid=1001) by devops(uid=0)
May 19 19:30:07 host sshd-session[5941]: pam_unix(sshd:session): session opened
for user devops(uid=1001) by devops(uid=0)
...output omitted...
May 19 19:35:53 host sudo[24067]: pam_unix(sudo:session): session closed for
user root
:
```

To search for text in a file that you opened with the `less` or `vim` command, first enter the slash character (/) and then type the pattern to find. Press `Enter` to start the search. Press `N` to find the next match.

```
user@host:~$ sudo less /var/log/messages
...output omitted...
May 19 19:28:51 host kernel: SCSI subsystem initialized
May 19 19:28:51 host kernel: ACPI: bus type USB registered
May 19 19:28:51 host kernel: usbcore: registered new interface driver usbfs
May 19 19:28:51 host kernel: usbcore: registered new interface driver hub
May 19 19:28:51 host kernel: usbcore: registered new device driver usb
May 19 19:28:51 host kernel: pps_core: LinuxPPS API ver. 1 registered
/usb$
```

```
root@host:~# vim /etc/chrony.conf
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.rhel.pool.ntp.org iburst
#server 1.rhel.pool.ntp.org iburst
#server 2.rhel.pool.ntp.org iburst
#server 3.rhel.pool.ntp.org iburst
server 172.25.254.254 iburst

# Ignore stratum in source selection.
stratumweight 0

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

...output omitted...

# Serve time even if not synchronized to any NTP server.
#local stratum 10
/[Nn][Tt][Pp]
```

## References

regex(7) and grep(1) man pages

## 2.2. Guided Exercise

### Match Text with Regular Expressions

Use regular expressions to match text in files and in the shell pipeline.

#### Outcomes

- Efficiently search for text in log files and configuration files.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start regexes-regex
```

#### Instructions

1. Log in to the servera machine as the student user and switch to the root user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Use the /etc/passwd and /etc/group files to find the UID and GID for the apache user and group.

Use the grep command to query for information that matches the expression.

```
[root@servera ~]# grep apache /etc/passwd /etc/group
/etc/passwd:apache:x:48:48:Apache:/usr/share/httpd/:sbin/nologin
/etc/group:apache:x:48:
```

3. Check the status for the httpd service and get some contextual information for your findings. Use the systemctl status command to get the status of the httpd service.

Use the grep command with the -B and -A options to get the two lines before and three lines after the line that matches the Active expression. Pay attention to case sensitivity when using the

grep command.

```
[root@servera ~]# systemctl status httpd | grep Active -B 2 -A 3
● httpd.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Active: active (running) since Wed 2025-05-28 22:47:18 UTC; 3min 48s ago
      Invocation: c7f967ed86a44e85ab5bc9845f601127
        Docs: man:httpd.service(8)
    Main PID: 1842 (httpd)
```

4. Search the /etc/httpd/conf/httpd.conf configuration file to find the location of the ServerRoot and DocumentRoot directories for the httpd server.

Use the grep command -i option to ignore case distinctions.

```
[root@servera ~]# grep -i -e ^serverroot -e ^documentroot \
/etc/httpd/conf/httpd.conf
ServerRoot "/etc/httpd"
DocumentRoot "/var/www/html"
```

5. Confirm that the httpd service stores the event logs in the /var/log/messages file.

- 5.1. Use the grep command to search for the lines that match the expression. Redirect the output to the /tmp/httpd.log file.

```
[root@servera ~]# grep httpd /var/log/messages > /tmp/httpd.log
```

- 5.2. Use the cat command to display the contents of the /tmp/httpd.log file.

```
[root@servera ~]# cat /tmp/httpd.log
May 28 ... servera systemd[1]: Starting httpd.service - The Apache HTTP
Server...
May 28 ... servera (httpd)[1842]: httpd.service: Referenced but unset ...
May 28 ... servera httpd[1842]: Server configured, listening on: port 80
May 28 ... servera systemd[1]: Started httpd.service - The Apache HTTP Server.
```

6. Use the ps ax command to confirm that the httpd server is currently running.

Use the grep command to limit the output to the necessary lines.

```
[root@servera ~]# ps ax | grep httpd
 1842 ?      Ss    0:00 /usr/sbin/httpd -DFOREGROUND
 1843 ?      S    0:00 /usr/sbin/httpd -DFOREGROUND
 1844 ?      Sl    0:00 /usr/sbin/httpd -DFOREGROUND
 1845 ?      Sl    0:00 /usr/sbin/httpd -DFOREGROUND
 1846 ?      Sl    0:00 /usr/sbin/httpd -DFOREGROUND
 2548 pts/2   S+   0:00 grep --color=auto httpd
```

7. Check the email address that is configured for the server administrator in the /etc/httpd/conf/httpd.conf file.

Use the vim command to open the file and then the slash character (/) to search the file.

Look for the line that starts with the ServerAdmin expression.

```
[root@servera ~]# vim /etc/httpd/conf/httpd.conf

...output omitted...
# 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition. These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#
#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed. This address appears on some server-generated pages, such
# as error documents. e.g. admin@your-domain.com
#
ServerAdmin root@localhost
...output omitted...

/^ServerAdmin
```

8. Return to the workstation machine as the student user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish regexes-regex
```

## 2.3. Quiz

### Matching Text with Regular Expressions

Review the following information and use it to answer the quiz questions.

The system has a words.txt file with the following content:

```
bash
RedHat
systemd
init
kernel
tty6
passwd
shadow
mount
lsmod
12345
grep
cron
fstab
404
```

Choose the correct answers to the following questions:

1. Which expression matches only the lines that contain at least one digit?

- a. `[:alpha:]`
- b. `[:digit:]`
- c. `[a-z0-9]`
- d. `^[:digit:]`

2. Which expression matches only the RedHat line?

- a. `^[A-Z]`
- b. `[A-Z]$`
- c. `^*[a-z]`
- d. `^A-Z`

**3.** Which expression matches lines that contain at least three consecutive lowercase letters?

- a.** [a-z][a-z][a-z]
- b.** [a-z]\*3
- c.** [a-z][0-9][a-z]
- d.** [a-z]\{3,\}

**4.** Which lines of text match the '^[:digit:]\*'\$' expression?

- a.** 12345, 404
- b.** tty6
- c.** tty6, 12345, 404
- d.** No lines match this expression.

**5.** Which command lists all lines that do not contain the `kernel` word, regardless of case?

- a.** grep -v 'kernel' words.txt
- b.** grep -i 'kernel' words.txt
- c.** grep -iv 'kernel' words.txt
- d.** grep --exclude 'kernel' words.txt

**6.** Which command shows three lines before and three lines after each match for `cron`?

- a.** grep -A3B3 'cron' words.txt
- b.** grep -AB3 'cron' words.txt
- c.** grep -C6 'cron' words.txt
- d.** grep -A3 -B3 'cron' words.txt

## Solution

---

1. Which expression matches only the lines that contain at least one digit?

- a. `[:alpha:]`
- b. `[:digit:]`
- c. `[a-z0-9]`
- d. `^[:digit:]`

2. Which expression matches only the RedHat line?

- a. `^[A-Z]`
- b. `[A-Z]$`
- c. `^a-z]`
- d. `^A-Z`

3. Which expression matches lines that contain at least three consecutive lowercase letters?

- a. `[a-z][a-z][a-z]`
- b. `[a-z]*3`
- c. `[a-z][0-9][a-z]`
- d. `[a-z]\{3,\}`

4. Which lines of text match the '`^[:digit:]*$`' expression?

- a. **12345, 404**
- b. `tty6`
- c. `tty6, 12345, 404`
- d. No lines match this expression.

5. Which command lists all lines that do not contain the `kernel` word, regardless of case?

- a. `grep -v 'kernel' words.txt`
- b. `grep -i 'kernel' words.txt`
- c. **grep -iv 'kernel' words.txt**
- d. `grep --exclude 'kernel' words.txt`

**6.** Which command shows three lines before and three lines after each match for cron?

- a. grep -A3B3 'cron' words.txt
- b. grep -AB3 'cron' words.txt
- c. grep -C6 'cron' words.txt
- d. **grep -A3 -B3 'cron' words.txt**

## 2.4. Summary

---

In this lesson, you learned to use regular expressions in various programming languages and tools to search, replace, and validate text. Regular expressions are powerful for tasks such as data validation, parsing, and text manipulation.

The grep command is a valuable tool for searching text by using regex patterns, for data analysis. You can use this tool to find lines in files or command outputs that match specific patterns.

You can also use regex in text pagers and editors, such as less and vim, to search and manipulate text interactively.

## **Chapter 3.**

### **Scheduling User Tasks**

---

#### **Goal**

Schedule programs to run in the future, either at a specific time and date or on a recurring basis, as a regular user.

#### **Sections**

- Scheduling a Future User Job (and Guided Exercise)
- Scheduling Recurring User Jobs (and Guided Exercise)
- Scheduling User Tasks (Quiz)

## 3.1. Scheduling a Future User Job

### Objectives

- Describe the commands to schedule to run at one specified future time.

### Deferred User Tasks

Sometimes you might need to run one or more commands at a specific future time. An example is a user who schedules a long-running maintenance task to occur in the middle of the night. Another example is a system administrator who is working on a firewall configuration and queues a safety job to reset the firewall settings to a former working state in ten minutes' time. The system administrator then deactivates the job before it runs, unless the new firewall configuration worked.

These scheduled commands are known as *tasks* or *jobs*, and the *deferred* term indicates that these tasks run in the future.

One solution for Red Hat Enterprise Linux users to schedule deferred tasks is the `at` command, which is installed and enabled by default. The `at` package provides the `atd` system daemon and the `at` and `atq` commands to interact with the daemon.

Any user can queue jobs for the `atd` daemon by using the `at` command. The `atd` daemon provides queues from a to z and from A to Z. The queue with highest letters run with high priority.

### Schedule Deferred User Tasks

Use the `at` *TIMESPEC* command to start entering a new job to schedule. The `at` command reads from `STDIN` (standard input, for example your keyboard) to obtain the commands to run. When manually entering commands, complete the input by pressing `Ctrl + D` on an empty line. You can use input redirection from a script file for entering more complex commands.

For example, use the `at` command to schedule the `myfile` script to run in five minutes' time, without needing to type the commands manually in a terminal window.

```
user@host:~$ at now +5min < myfile
warning: commands will be executed using /bin/sh
job 2 at Wed May 14 20:40:00 2025
```

The `at` command *TIMESPEC* argument accepts natural time specifications for when to run a job. For example, specify a time as `02:00pm`, `15:59`, `midnight`, or even `teatime`, followed by an optional

date or number of days in the future.

The TIMESPEC argument expects time and date specifications in that order. If you provide the date and not the time, then the time defaults to the current time. If you provide the time and not the date, then the date is considered to be matched, and the jobs run when the time next matches.

The following example shows a job schedule without providing the date. The at command schedules the job for today or tomorrow depending whether the time has passed.

```
user@host:~$ at 21:03 < myfile
warning: commands will be executed using /bin/sh
job 3 at Wed May 14 21:03:00 2025
```

The at command schedules the job for tomorrow when the time is reached.

```
user@host:~$ at 21:00 < myfile
warning: commands will be executed using /bin/sh
job 4 at Thu May 15 21:00:00 2025
```

The man pages for the at command and other documentation sources use lowercase to write the natural time specifications. You can use lowercase, sentence case, or uppercase.

Here are examples of time specifications that you can use:

- now +5min
- teatime tomorrow (teatime is 16:00)
- noon +4 days
- 5pm august 3 2025

For other valid time specifications, refer to the local timespec document in the references.

## Inspect and Manage Deferred User Jobs

For an overview of the pending jobs for the current user, use the atq or the at -l command.

```
user@host:~$ atq
28  Mon May 19 05:13:00 2025 a user
29  Tue May 20 16:00:00 2025 h user
30  Wed May 21 12:00:00 2025 a user
```

In the preceding output, every line represents a different scheduled future job. The following description applies to the first line of the output:

- **28** is the unique job number.
- **Mon May 16 05:13:00 2022** is the execution date and time for the scheduled job.
- **a** indicates that the job is scheduled with the default queue a.
- **user** is the owner and user for the job.

#### **Important**

Unprivileged users can view and manage only their own jobs. The root user can view and manage all jobs for all users.

Use the `at -c JOBNUMBER` command to inspect the commands that run when the atd daemon executes a job. This command shows the job's environment, which is set from the user's environment when they created the job, and the command syntax to run.

## Remove Jobs from a Schedule

The `atrm JOBNUMBER` command removes a scheduled job. Remove a scheduled job when you no longer need it, for example, when a remote firewall configuration succeeded and you do not need to reset it.

#### **References**

`at(1)` and `atd(8)` man pages

`/usr/share/doc/at/timespec`

## 3.2. Guided Exercise

### Schedule a Future User Job

Schedule commands to run at one specified future time.

#### Outcomes

- Schedule a job to run at a specified future time.
- Inspect the commands that a scheduled job runs.
- Delete the scheduled jobs.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start scheduling-at
```

#### Instructions

1. Log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Schedule a job to run in two minutes from now. Save the output of the date command to the /home/student/myjob.txt file.
  - 2.1. Pass the date >> /home/student/myjob.txt string as the input to the at command, so that the job runs in two minutes from now.

```
[student@servera ~]$ echo "date >> /home/student/myjob.txt" | at now +2min
warning: commands will be executed using /bin/sh
job 1 at Wed May 28 18:05:00 2025
```

- 2.2. List the scheduled jobs.

```
[student@servera ~]$ atq  
1 Wed May 28 18:05:00 2025 a student
```

- 2.3.** Monitor the deferred jobs queue in real time. After the atd daemon executes, it removes the job from the queue.

The command updates the output of the atq command every two seconds, by default. After the atd daemon removes the deferred job from the queue, press **Ctrl + C** to exit the watch command and return to the shell prompt.

```
[student@servera ~]$ watch atq  
Every 2.0s: atq servera: Wed May 28 18:04:42 2025  
1 Wed May 28 18:05:00 2025 a student  
Ctrl + C
```

- 2.4.** Verify that the contents of the /home/student/myjob.txt file match the output of the date command.

The output matches the output of the date command, which confirms that the scheduled job executed successfully.

```
[student@servera ~]$ cat myjob.txt  
Wed May 28 18:05:00 UTC 2025
```

- 3.** Interactively schedule a job in the g queue that runs at teatime (16:00). The job prints the It's teatime message and appends the message to the /home/student/tea.txt file.

```
[student@servera ~]$ at -q g teatime  
warning: commands will be executed using /bin/sh  
at Thu May 29 16:00:00 2025  
at> echo "It's teatime" >> /home/student/tea.txt  
at> Ctrl + D  
job 2 at Thu May 29 16:00:00 2025
```

- 4.** Interactively schedule another job with the b queue that runs at 16:05. The job prints The cookies are good message and appends the message to the /home/student/cookies.txt file.

```
[student@servera ~]$ at -q b 16:05
warning: commands will be executed using /bin/sh
at Thu May 29 16:05:00 2025
at> echo "The cookies are good" >> /home/student/cookies.txt
at> Ctrl + D
job 3 at Thu May 29 16:05:00 2025
```

**5.** Inspect the commands in the pending jobs.**5.1.** View the job numbers of the pending jobs.

Use the job numbers from the output on your system, rather than the job numbers in these examples.

```
[student@servera ~]$ atq
2      Thu May 29 16:00:00 2025 g student
3      Thu May 29 16:05:00 2025 b student
```

**5.2.** View the commands in the pending job number 2. Replace the job number if it changed for you.

The job executes an echo command that appends the It's teatime message to the /home/student/tea.txt file.

```
[student@servera ~]$ at -c 2
...output omitted...
echo "It's teatime" >> /home/student/tea.txt
...output omitted...
```

**5.3.** View the commands in the pending job number 3. Replace the job number if it changed for you.

The job executes an echo command that appends the message The cookies are good to the /home/student/cookies.txt file.

```
[student@servera ~]$ at -c 3
...output omitted...
echo "The cookies are good" >> /home/student/cookies.txt

...output omitted...
```

6. Remove the job that runs at teatime (16:00) by using the atrm command.

```
[student@servera ~]$ atrm 2
```

7. Verify that the scheduled job to run at teatime (16:00) no longer exists.

- 7.1. View the list of pending jobs, and confirm that the scheduled job to run at teatime (16:00) no longer exists.

```
[student@servera ~]$ atq
3      Thu May 29 16:05:00 2025 b student
```

8. Return to the workstation machine as the student user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish scheduling-at
```

## 3.3. Scheduling Recurring User Jobs

### Objectives

- Schedule commands to run on a recurring schedule by setting up Cron jobs as a regular user.

### Recurring User Jobs

Recurring jobs run repeatedly on a schedule. Red Hat Enterprise Linux provides the `crond` daemon, which is enabled and started by default.

The `crond` daemon reads multiple configuration files: one per user, and a set of system-wide files. Each user has a personal file that they edit with the `crontab` command `-e` option.

When executing recurring jobs, these configuration files provide detailed control to users and administrators. If the scheduled job is not written to use redirection, then the `crond` daemon emails any generated output or errors to the job owner.

### Schedule Recurring User Jobs

Use the `crontab` command to manage scheduled jobs. The following list shows the commands that a user can use to manage their jobs:

Table 8. Examples of the `crontab` Command

Command	Intended use
<code>crontab -l</code>	List the jobs for the current user.
<code>crontab -r</code>	Remove all jobs for the current user.
<code>crontab -e</code>	Edit jobs for the current user.
<code>crontab filename</code>	Remove all jobs, and replace them with jobs that are read from the <code>filename</code> file. This command uses <code>stdin</code> input when no file is specified.

A privileged user might use the `crontab` command `-u` option to manage jobs for another user. The `crontab` command is never used to manage system jobs, and using the `crontab` command as the root user is not recommended due to the ability to exploit personal jobs that are configured to run as the root user.

## User Job Format

The `crontab` command `-e` option invokes the `vim` editor by default unless the `EDITOR` environment variable is set for another editor. Each job must use a unique line in the `crontab` file. Follow these recommendations for valid entries when writing recurring jobs:

- Empty lines for ease of reading
- Comments on lines that start with the number sign (#)
- Environment variables with a `NAME=value` format, which affects all lines after the line where they are declared

Standard variable settings include the `SHELL` variable, to declare the shell that interprets the remaining lines of the `crontab` file. The `MAILTO` variable determines who should receive the emailed output.

### Note

The ability to send an email requires additional system configuration for a local mail server or an SMTP relay.

The fields in the `crontab` file appear in the following order:

- Minutes
- Hours
- Day of month
- Month
- Day of week
- Command

The command executes when the *Day of month* or *Day of week* fields use the same value other than the \* character. For example, to run a command on the 11th day of every month, and every Friday at 12:15 (24-hour format), use the following job format:

```
15 12 11 * Fri command
```

The first five fields all use the same syntax rules:

- Use the \* character to execute in every possible instance of the field.
- A number to specify the number of minutes or hours, a date, or a day of the week. For days of the week, 0 equals Sunday, 1 equals Monday, 2 equals Tuesday, and so on, and 7 also equals Sunday.
- Use x-y for a range, which includes the x and y values.
- Use x , y for lists. Lists might include ranges as well, for example, 5 , 10-13 , 17 in the Minutes column, for a job to run at 5, 10, 11, 12, 13, and 17 minutes past the hour.
- The \*/x indicates an interval of x; for example, \*/7 in the Minutes column runs a job every seven minutes.

Additionally, 3-letter English abbreviations are used for months or days of the week, for example, Jan, Feb, and Mon, Tue.

The last field contains the full command with options and arguments to execute with the default shell. If the command contains an unescaped percentage sign (%), then that percentage sign is treated as a newline character, and everything after the percentage sign passes to the command as stdin input.

### Important

When specifying a range, the first number must be less than or equal to the second one. This applies to a number's matching English abbreviation as well.

For example, the range Tue-Fri (numerically expressed as 2-5) is valid, however the range Fri-Tue (numerically written as 5-2) is not valid because the first number 5 is greater than the second number 2. The second range can be correctly expressed as a list of two ranges: Sun-Tue, Fri-Sat (numerically expressed as 0-2, 5-6).

## Examples of Recurring User Jobs

The following job executes the /usr/local/bin/yearly\_backup command at 09:00:00 on 3 February, every year. February is represented as the number 2 in the example, because it is the second month of the year.

```
0 9 3 2 * /usr/local/bin/yearly_backup
```

The following job sends an email that contains the Chime word to the owner of this job at every fifth minute past every hour from 09:00 through 16:59 on Fridays in July.

```
*/5 9-16 * Jul 5 echo "Chime"
```

The preceding 9-16 range of hours means that the job timer starts at 09:00 and continues until 16:59. The job starts executing at 09:00 with the last execution at 16:55, because five minutes after 16:55 is 17:00, which is beyond the given scope of hours.

If a range is specified for the hours instead of a single value, then all hours within the range might match. Therefore, with the hours of 9-16, this example matches every five minutes from 09:00 through 16:55.

#### Note

This example job sends the output as an email, because the crond daemon recognizes that the job allowed the output to go to the STDIO channel without redirection. Because the cron jobs run in a background environment without an output device (which is known as a *controlling terminal*), the crond daemon buffers the output and creates an email to send it to the specified user in the configuration. For system jobs, the email is sent to the root account.

The following job runs the /usr/local/bin/daily\_report command every day from Monday to Friday at two minutes before midnight.

```
58 23 * * 1-5 /usr/local/bin/daily_report
```

The following job executes the mutt command to send the Checking in mail message to the developer@example.com recipient every day from Monday to Friday at 09:00.

```
0 9 * * 1-5 mutt -s "Checking in" developer@example.com % Hi there, just  
checking in.
```

#### References

crond(8), crontab(1), and crontab(5) man pages

## 3.4. Guided Exercise

### Schedule Recurring User Jobs

Schedule and inspect recurring jobs by setting up Cron jobs as a regular user.

#### Outcomes

- Schedule recurring jobs to run as a nonprivileged user.
- Inspect the commands that a scheduled recurring job runs.
- Remove scheduled recurring jobs.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start scheduling-cron
```

#### Instructions

1. Log in to the servera machine as the student user .

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Schedule a recurring job as the student user that appends the current date and time to the /home/student/my\_first\_cron\_job.txt file every two minutes. Use the date command to display the current date and time. The job must run only from one day before to one day after the current time, not on any other day.

- 2.1. Use the date command to display the current date and time. Note the day of the week, which you need for the next steps.

```
[student@workstation ~]$ date  
Wed May 28 18:31:40 UTC 2025  
[student@servera ~]$
```

- 2.2.** You can also use the date -d "last day" +%a command to display the day before the current time, and the date -d "next day" +%a command to display the day after the current time.

```
[student@servera ~]$ date -d "last day" +%a  
Tue  
[student@servera ~]$ date -d "next day" +%a  
Thu
```

- 2.3.** Open the crontab file with the default text editor.

```
[student@servera ~]$ crontab -e
```

- 2.4.** Insert the following line. Replace the range of days to be one day before to one day after the current day. For example, if the current day is Wednesday (Wed) then the range of days is Tue-Thu.

```
*/2 * * * Tue-Thu /usr/bin/date >> /home/student/my_first_cron_job.txt
```

- 2.5.** Save the changes and exit the editor. When the editor exits, the following output is shown:

```
no crontab for student - using an empty one  
crontab: installing new crontab  
[student@servera ~]$
```

- 3.** Use the crontab -l command to list the scheduled recurring jobs. Inspect the command that you scheduled to run as a recurring job in the preceding step.

Verify that the job runs the /usr/bin/date command and appends its output to the /home/student/my\_first\_cron\_job.txt file.

```
[student@servera ~]$ crontab -l  
*/2 * * * Tue-Thu /usr/bin/date >> /home/student/my_first_cron_job.txt
```

- 4.** Instruct your shell prompt to sleep until the /home/student/my\_first\_cron\_job.txt file is created because of the successful execution of the recurring job that you scheduled. Wait for your

shell prompt to return.

The while command uses ! test -f to continue to run a loop, and sleeps for one second until the my\_first\_cron\_job.txt file is created in the /home/student directory.

```
[student@servera ~]$ while ! test -f my_first_cron_job.txt; do sleep 1s; done
```

5. Verify that the contents of the /home/student/my\_first\_cron\_job.txt file match the output of the date command.

```
[student@servera ~]$ cat my_first_cron_job.txt  
Wed May 28 18:34:01 UTC 2025
```

6. Remove all the scheduled recurring jobs for the student user, and verify that no recurring jobs exist for the student user.

- 6.1. Remove all the scheduled recurring jobs for the student user.

```
[student@servera ~]$ crontab -r  
...output omitted...
```

- 6.2. Verify that no recurring jobs exist for the student user.

```
[student@servera ~]$ crontab -l  
no crontab for student
```

- 6.3. Return to the workstation machine as the student user.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish scheduling-cron
```

## 3.5. Quiz

### Scheduling User Tasks

---

Choose the correct answers to the following questions:

1. Which command displays all the user jobs that you scheduled to run as deferred jobs?  
 a. atq  
 b. atrm  
 c. at -c  
 d. at --display
  
2. Which command removes the deferred user job with the job number 5?  
 a. at -c 5  
 b. atrm 5  
 c. at 5  
 d. at --delete 5
  
3. Which command displays all the scheduled recurring user jobs for the currently logged-in user?  
 a. crontab -r  
 b. crontab -l  
 c. crontab -u  
 d. crontab -V
  
4. Which job format executes the /usr/local/bin/daily\_backup command hourly from 9 AM to 6 PM on all days from Monday through Friday?  
 a. 00 \* \* \* Mon-Fri /usr/local/bin/daily\_backup  
 b. \* \*/9 \* \* Mon-Fri /usr/local/bin/daily\_backup  
 c. 00 \*/18 \* \* /usr/local/bin/daily\_backup  
 d. 00 09-18 \* \* Mon-Fri /usr/local/bin/daily\_backup

## Solution

---

1. Which command displays all the user jobs that you scheduled to run as deferred jobs?  
 a. `atq`  
 b. `atrm`  
 c. `at -c`  
 d. `at --display`
  
2. Which command removes the deferred user job with the job number 5?  
 a. `at -c 5`  
 b. `atrm 5`  
 c. `at 5`  
 d. `at --delete 5`
  
3. Which command displays all the scheduled recurring user jobs for the currently logged-in user?  
 a. `crontab -r`  
 b. `crontab -l`  
 c. `crontab -u`  
 d. `crontab -V`
  
4. Which job format executes the `/usr/local/bin/daily_backup` command hourly from 9 AM to 6 PM on all days from Monday through Friday?  
 a. `00 * * * Mon-Fri /usr/local/bin/daily_backup`  
 b. `* */9 * * Mon-Fri /usr/local/bin/daily_backup`  
 c. `00 */18 * * * /usr/local/bin/daily_backup`  
 d. `00 09-18 * * Mon-Fri /usr/local/bin/daily_backup`

## 3.6. Summary

---

In this lesson, you learned how to create and manage deferred and recurring jobs in the system.

The `at` command schedules a job to run one time in the future, so users can manage their own job queues. The `atd` daemon is responsible for monitoring and executing these jobs at the specified time.

The `cron` daemon manages recurring jobs, which are scheduled by using the `crontab` command.

The `crontab` command enables users to create and edit their own cron jobs, which follow a specific syntax to define the execution schedule.

## Chapter 4.

### Scheduling System Tasks

---

#### Goal

Schedule system programs that must run on a recurring basis to support daemons or operating system functions.

#### Sections

- Managing Repeating Jobs by Using Systemd Timer Units (and Guided Exercise)
- Managing Temporary Files (and Guided Exercise)
- Scheduling Recurring System Tasks with Cron (and Guided Exercise)
- Scheduling User Tasks (Quiz)

## 4.1. Managing Repeating Jobs by Using Systemd Timer Units

### Objectives

- Investigate, interpret, and manage recurring system tasks that are scheduled as the systemd timer units.

### Systemd Timer

Starting from Red Hat Enterprise Linux 10, the systemd daemon and service manager replaces the traditional cron daemon for most of the scheduling system tasks, via the systemd timer units.

A systemd timer unit is defined by a unit configuration file with a file name that ends with .timer. Timer units activate a unit of another type, which is usually a service. By default, the unit name matches the timer unit name. Timer units enable timer-based activation of other units and describe when to run another unit. The systemd timer unit logs timer events in system journals for ease of debugging.

#### Note

Fuller discussion of the systemd timer units is beyond the scope of this section. For more information, see the `systemd(1)` man page.

### List Timer Units

To list all timer units that are loaded in memory, use the `systemctl list-units -t timer` command. This command lists timer units that are active, pending, or failed. You can use the `--all` option to include inactive timer units.

```
root@host:~# systemctl list-units -t timer
UNIT                  LOAD   ACTIVE SUB      DESCRIPTION
dnf-makecache.timer  loaded  active waiting dnf makecache --timer
fstrim.timer          loaded  active waiting Discard unused ...
fwupd-refresh.timer   loaded  active waiting Refresh fwupd ...
logrotate.timer       loaded  active waiting Daily rotation ...
plocate-updatedb.timer loaded  active waiting Update the plocate ...
raid-check.timer      loaded  active waiting Weekly RAID setup ...
systemd-tmpfiles-clean.timer loaded  active waiting Daily Cleanup ...
...output omitted...
```

The column headings are as follows:

**UNIT**

The full name of the unit.

**LOAD**

Indicates whether the unit definition was loaded.

**ACTIVE**

Indicates whether the unit is active at a high level.

**SUB**

Shows the unit-type-specific detailed state of the unit; possible values vary by unit type.

**DESCRIPTION**

A brief description of the unit's purpose.

## List Timer Unit Files

You can list all installed timer unit files with the `systemctl list-unit-files` command. In contrast to the `systemctl list-units` command, the `systemctl list-unit-files` command also includes unit templates.

```
root@host:~# systemctl list-unit-files -t timer
UNIT FILE                      STATE   PRESET
dnf-makecache.timer             enabled  enabled
fstrim.timer                    enabled  enabled
fwupd-refresh.timer             enabled  enabled
insights-client.timer          disabled  disabled
logrotate.timer                 enabled  enabled
mdadm-last-resort@.timer       static   -
mdcheck_continue.timer          disabled  disabled
mdcheck_start.timer             disabled  disabled
mdmonitor-oneshot.timer        disabled  disabled
plocate-updatedb.timer          enabled  enabled
podman-auto-update.timer       disabled  disabled
raid-check.timer                enabled  enabled
rhc-canonical-facts.timer      disabled  disabled
systemd-sysupdate-reboot.timer disabled  disabled
systemd-sysupdate.timer         disabled  disabled
systemd-tmpfiles-clean.timer   static   -
```

16 unit files listed.

The column headings are as follows:

#### UNIT FILE

The full name of the unit file.

#### STATE

Indicates the current enablement state of the unit.

#### PRESET

Indicates whether the unit is enabled by default.

## Start and Stop Systemd Timer Units

Use the `systemctl start` command to start a timer unit. To start the log rotation timer unit named `logrotate`, use the following command:

```
root@host:~# systemctl start logrotate.timer
```

Use the `systemctl stop` command to stop a timer unit. To stop the log rotation timer unit named `logrotate`, use the following command:

```
root@host:~# systemctl stop logrotate.timer
```

## Enable and Disable Systemd Timer Units

Use the `systemctl enable` command to enable a systemd timer unit at system boot time. To enable the log rotation timer unit named `logrotate`, use the following command:

```
root@host:~# systemctl enable logrotate.timer
```

The `systemctl enable` command alone does not start the systemd unit. To enable and start the systemd unit in one step, include the `--now` switch:

```
root@host:~# systemctl enable --now logrotate.timer
```

Use the `systemctl disable` command to disable a timer unit at system boot time. To disable the log rotation timer unit named `logrotate`, use the following command:

```
root@host:~# systemctl disable logrotate.timer
```

The `systemctl disable` command alone does not stop the systemd unit. To disable and stop the systemd unit in one step, include the `--now` switch:

```
root@host:~# systemctl disable --now logrotate.timer
```

## Timer Unit Status

You can use the `systemctl status` command to display the current status of a timer unit and its corresponding service unit.

To show the current status of the `logrotate` timer and service units, use the following command:

```
root@host:~# systemctl status logrotate.timer
● logrotate.timer - Daily rotation of log files
    Loaded: loaded (/usr/lib/.../logrotate.timer; enabled; preset: enabled)
    Active: active (waiting) since Wed 2025-05-28 23:28:22 UTC; 54min ago
      Invocation: 3a95de54d4c04c539d43b3808c27e99a
        Trigger: Fri 2025-05-30 00:48:43 UTC; 24h left
      Triggers: ● logrotate.service
        Docs: man:logrotate(8)
               man:logrotate.conf(5)

May 28 ... workstation systemd[1]: Started logrotate.timer - Daily ...
```

## Example Timer Unit Configuration File

The sysstat package provides the systemd timer unit, which is the sysstat-collect.timer service, to collect system statistics every 10 minutes.

The following output shows the contents of the /usr/lib/systemd/system/sysstat-collect.timer configuration file:

```
...output omitted...
[Unit]
Description=Run system activity accounting tool every 10 minutes

[Timer]
OnCalendar=*:00/10

[Install]
WantedBy=sysstat.service
```

The [Unit] and [Install] sections contain configuration items that are common to all the systemd units. The timer-specific options are defined in the [Timer] section.

In this example, the OnCalendar=`*:00/10` option signifies that this timer unit activates the corresponding sysstat-collect.service service unit every 10 minutes.

You can specify more complex time intervals. For example, when the OnCalendar option is set to a value of `2025-07-* 12:35,37,39:16`, the timer unit activates the corresponding service unit at the 12:35:16, 12:37:16, and 12:39:16 times, on every day of July 2025.

You can also specify relative timers by using the OnUnitActiveSec option. For example, when the OnUnitActiveSec option is set to a value of `15min`, the timer unit triggers the corresponding service

unit to start 15 minutes after the last time that the service unit was activated.

For more information about timer unit options, refer to the `systemd.timer(5)` man page.

### Important

Do not modify any unit configuration files in the `/usr/lib/systemd/system` directory. The `systemd` unit overrides all configuration changes that are made in the `/usr/lib/systemd/system` directory. To modify a `systemd` unit's configuration, copy its unit configuration file from the `/usr/lib/systemd/system` directory to the `/etc/systemd/system` directory, and then modify the copied file to prevent any update to the provider package from overriding the changes. If two files exist with the same name in the `/usr/lib/systemd/system` and `/etc/systemd/system` directories, then the `systemd` timer unit parses the file in the `/etc/systemd/system` directory.

After you change the timer unit configuration file, use the `systemctl daemon-reload` command to ensure that the `systemd` timer unit loads the changes.

```
root@host:~# systemctl daemon-reload
```

After reloading the `systemd` daemon configuration, use the `systemctl` command to enable and immediately activate the timer unit.

```
root@host:~# systemctl enable --now UNITNAME.timer
```

### References

`systemctl(1)`, `systemd.directives(7)`, `systemd.time(7)`, `systemd.timer(5)`, `systemd.syntax(7)`, and `systemd.unit(5)` man pages

For more information about `systemd` unit files, refer to the *Using Systemd Unit Files to Customize and Optimize Your System* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_systemd\\_unit\\_files\\_to\\_customize\\_and\\_optimize\\_your\\_system/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_systemd_unit_files_to_customize_and_optimize_your_system/index)

Upstream `systemd` project site: <https://systemd.io/>

## 4.2. Guided Exercise

### Manage Repeating Jobs by Using Systemd Timer Units

Schedule and inspect recurring system tasks that are scheduled as the systemd timer units.

#### Outcomes

- Update the systemd timer unit that gathers system activity data.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start systasks-timers
```

#### Instructions

- Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Install the sysstat package. List the systemd unit files for the sysstat package that were installed in the /usr/lib/systemd/system directory. Examine the sysstat-collect.timer timer unit file.

##### 2.1. Install the sysstat package.

```
[root@servera ~]# dnf install sysstat
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

##### 2.2. List the systemd unit files for the sysstat package that were installed in the /usr/lib/systemd/system directory.

```
[root@servera ~]# ls -1 /usr/lib/systemd/system/sysstat*
/usr/lib/systemd/system/sysstat-collect.service
/usr/lib/systemd/system/sysstat-collect.timer
/usr/lib/systemd/system/sysstat-rotate.service
/usr/lib/systemd/system/sysstat-rotate.timer
/usr/lib/systemd/system/sysstat-summary.service
/usr/lib/systemd/system/sysstat-summary.timer
/usr/lib/systemd/system/sysstat.service
```

- 2.3.** Examine the sysstat-collect.timer timer unit file in the /usr/lib/systemd/system directory.

The timer unit named sysstat-collect.timer triggers the matching service unit named sysstat-collect.service every 10 minutes.

```
# /usr/lib/systemd/system/sysstat-collect.timer
# (C) 2014 Tomasz Torcz <tomek@pipebreaker.pl>
#
# sysstat-12.7.6 systemd unit file:
#       Activates activity collector every 10 minutes

[Unit]
Description=Run system activity accounting tool every 10 minutes

[Timer]
OnCalendar=*\:00/10

[Install]
WantedBy=sysstat.service
```

- 2.4.** Examine the matching sysstat-collect.service service unit file in the /usr/lib/systemd/system directory.

The service unit file named sysstat-collect.service collects system activity data with the /usr/lib64/sa/sa1 shell script.

```
# /usr/lib/systemd/system/sysstat-collect.service
# (C) 2014 Tomasz Torcz <tomek@pipebreaker.pl>
#
# sysstat-12.7.6 systemd unit file:
#       Collects system activity data
#       Activated by sysstat-collect.timer unit

[Unit]
Description=system activity accounting tool
Documentation=man:sa1(8)
After=sysstat.service

[Service]
Type=oneshot
User=root
ExecStart=/usr/lib64/sa/sa1 1 1
```

- 3.** Unit files in the `/etc/systemd/system` directory override the unit files that are installed in the `/usr/lib/systemd/system` directory.

Copy the `sysstat-collect.timer` timer unit configuration file in the `/usr/lib/systemd/system` directory to the `/etc/systemd/system` directory. Edit the timer unit configuration file in the `/etc/systemd/system` directory so that system activity data is collected every two minutes. Notify the `systemd` daemon of the changes.

- 3.1.** Copy the `/usr/lib/systemd/system/sysstat-collect.timer` file to the `/etc/systemd/system/sysstat-collect.timer` file.

```
[root@servera ~]# cp /usr/lib/systemd/system/sysstat-collect.timer \
/etc/systemd/system/sysstat-collect.timer
```

- 3.2.** Edit the `/etc/systemd/system/sysstat-collect.timer` timer unit configuration file so that system activity data is collected every two minutes. Use the `vim` command to edit the configuration file.

Replace any occurrence of the `10 minutes` string with `2 minutes` throughout the unit configuration file, including any occurrences in commented lines.

```
...output omitted...
# Activates activity collector every 2 minutes

[Unit]
Description=Run system activity accounting tool every 2 minutes

[Timer]
OnCalendar=*:00/2

[Install]
WantedBy=sysstat.service
```

**3.3.** Notify the systemd daemon of changes by reloading the daemon.

```
[root@servera ~]# systemctl daemon-reload
```

**3.4.** Enable and activate the sysstat-collect.timer unit.

```
[root@servera ~]# systemctl enable --now sysstat-collect.timer
```

**4.** The sysstat-collect.service service unit collects the system activity data in a binary file in the /var/log/sa directory. Verify that the binary file in that directory is modified within two minutes.

**4.1.** Wait until the binary file is created in the /var/log/sa directory.

The while command, `ls /var/log/sa | wc -l`, returns 0 when the file does not exist, or returns 1 when the file exists. The while command pauses for one second when the file does not exist. The while loop exits when the file exists.

```
[root@servera ~]# while [ $(ls /var/log/sa | wc -l) -eq 0 ]; \
do sleep 1s; done
```

**4.2.** Verify that the binary file in the /var/log/sa directory is modified within two minutes.

```
[root@servera ~]# ls -l /var/log/sa
total 4
-rw-r--r--. 1 root root 2540 Apr  5 04:08 sa05
```

**5.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish systasks-timers
```

## 4.3. Managing Temporary Files

### Objectives

- Configure software that uses a timer unit to schedule tasks to clear temporary files on the system, or to ensure that certain files or directories exist.

### The Systemd-tempfiles Tool

Most critical applications and services use temporary files and directories. Some applications and users use the `/tmp` directory to hold transient working data, whereas other applications use task-specific locations such as daemon- and user-specific volatile directories under the `/run` directory, which exist only in memory. When the system reboots or loses power, memory-based file systems are self-cleaning.

Commonly, daemons and scripts operate correctly only when their expected temporary files and directories exist. Additionally, purging temporary files on persistent storage is necessary to prevent disk space issues or stale working data.

Red Hat Enterprise Linux includes the `systemd-tmpfiles` tool, which provides a structured and configurable method to manage temporary directories and files.

At system boot, one of the first `systemd` service units to launch is the `systemd-tmpfiles-setup` service. This service runs the `systemd-tmpfiles` command, which reads instructions from the `/usr/lib/tmpfiles.d/*.conf`, `/run/tmpfiles.d/*.conf`, and `/etc/tmpfiles.d/*.conf` configuration files. These configuration files list files and directories that the `systemd-tmpfiles-setup` service is instructed to create, delete, or secure with permissions.

### Clean Temporary Files with a Systemd Timer

To prevent long-running systems from filling up their disks with stale data, a `systemd` timer unit named `systemd-tmpfiles-clean.timer` triggers at a regular interval the `systemd-tmpfiles-clean.service` unit, which executes the `systemd-tmpfiles --clean` command.

A `systemd` timer unit configuration has a `[Timer]` section to indicate how to start the service with the same name as the timer.

Use the following `systemctl` command to view the contents of the `systemd-tmpfiles-clean.timer` unit configuration file:

```
user@host:~$ systemctl cat systemd-tmpfiles-clean.timer
# /usr/lib/systemd/system/systemd-tmpfiles-clean.timer
# SPDX-License-Identifier: LGPL-2.1-or-later
#
# This file is part of systemd.
...output omitted...

[Unit]
Description=Daily Cleanup of Temporary Directories
Documentation=man:tmpfiles.d(5) man:systemd-tmpfiles(8)
ConditionPathExists=!/etc/initrd-release

[Timer]
OnBootSec=15min
OnUnitActiveSec=1d
```

In the preceding configuration, the `OnBootSec=15min` parameter indicates that the `systemd-tmpfiles-clean.service` unit gets triggered 15 minutes after the system boots up. The `OnUnitActiveSec=1d` parameter indicates that any further trigger to the `systemd-tmpfiles-clean.service` unit happens 24 hours after the service unit was last activated.

Change the parameters in the `systemd-tmpfiles-clean.timer` unit configuration file to meet your requirements. For example, a `30min` value for the `OnUnitActiveSec` parameter triggers the `systemd-tmpfiles-clean.service` unit 30 minutes after the service unit was last activated. As a result, the `systemd-tmpfiles-clean.service` unit gets triggered every 30 minutes after the changes are recognized.

After you change the timer unit configuration file, use the `systemctl daemon-reload` command to ensure that the `systemd` daemon loads the new configuration.

```
root@host:~# systemctl daemon-reload
```

You can view the status of the timer unit, including whether it is loaded in memory or active, by using the `systemctl status` command:

```
root@host:~# systemctl status systemd-tmpfiles-clean.timer
● systemd-tmpfiles-clean.timer - Daily Cleanup of Temporary Directories
    Loaded: loaded (/usr/lib/.../systemd-tmpfiles-clean.timer; static)
    Active: active (waiting) since Fri 2025-05-30 13:59:31 UTC; 19min ago
      Invocation: 497008f1a6594041812a660b64f3c550
        Trigger: Sat 2025-05-31 14:14:31 UTC; 23h left
      Triggers: ● systemd-tmpfiles-clean.service
        Docs: man:tmpfiles.d(5)
               man:systemd-tmpfiles(8)

May 30 13:59:31 host systemd[1]: Started systemd-tmpfiles-clean...
```

## Clean Temporary Files Manually

The `systemd-tmpfiles --clean` command parses the same configuration files as the `systemd-tmpfiles --create` command; instead of creating files and directories, it purges all files that were not accessed, changed, or modified more recently than the maximum age that is defined in the configuration file.

For detailed information about the format of the configuration files for the `systemd-tmpfiles` service, refer to the `tmpfiles.d(5)` man page.

The syntax consists of the following columns: Type, Path, Mode, UID, GID, Age, and Argument. Type refers to the action for the `systemd-tmpfiles` service to take; for example, the `d` option to create a directory if it does not exist, or the `Z` option to recursively restore SELinux contexts, file permissions, and ownership.

The following example purges a configuration in the `/run/systemd/seats` directory:

```
d /run/systemd/seats 0755 root root -
```

- Creates the `/run/systemd/seats` directory if it does not exist.
- If the directory exists, takes no action.
- Sets the `root` user and the `root` group as owners.
- Sets the file permissions to `0755` (`rwxr-xr-x`).
- Does not set an age or any additional arguments.
- The `systemd-tmpfiles` service does not purge this directory automatically.

The following example removes files from the `/home/student` directory that were not accessed, changed, or modified for more than one day:

```
D /home/student 0700 student student 1d
```

- Creates the `/home/student` directory if it does not exist.
- If the directory exists, then removes all its contents.
- Sets the `student` user and the `student` group as owners.
- Sets the file permissions to `0700` (`rwx-----`).
- Sets the age to one day.
- Does not set any additional arguments.
- When the system runs the `systemd-tmpfiles --clean` command, all files that you did not access, change, or modify for more than one day are removed from the `/home/student` directory.

The following example creates a symbolic link to the `/etc/fstab` directory:

```
L /run/fstablink - root root - /etc/fstab
```

- Creates the `/run/fstablink` symbolic link to point to the `/etc/fstab` directory.
- Sets the `root` user and the `root` group as owners.
- Never automatically purges this line.
- Does not set the permissions or age.
- Sets the argument to `/etc/fstab` to indicate the source of the link.

## Configuration File Precedence

The `systemd-tmpfiles-clean` service configuration files can exist in three places:

- `/etc/tmpfiles.d/*.conf`
- `/run/tmpfiles.d/*.conf`
- `/usr/lib/tmpfiles.d/*.conf`

Use the files in the `/etc/tmpfiles.d/` directory to configure custom temporary locations, and to override vendor-provided defaults. The files in the `/run/tmpfiles.d/` directory are volatile files, which daemons normally use to manage their own runtime temporary files. Relevant RPM packages provide the files in the `/usr/lib/tmpfiles.d/` directory; therefore do not edit these files.

If a file in the `/run/tmpfiles.d/` directory has the same file name as a file in the `/usr/lib/tmpfiles.d/` directory, then the service uses the file in the `/run/tmpfiles.d/`

directory. If a file in the `/etc/tmpfiles.d/` directory has the same file name as a file in either the `/run/tmpfiles.d/` or the `/usr/lib/tmpfiles.d/` directories, then the service uses the file in the `/etc/tmpfiles.d/` directory.

Given these precedence rules, you can override vendor-provided settings by copying the relevant file to the `/etc/tmpfiles.d/` directory and editing it. By using these configuration locations correctly, you can manage administrator-configured settings from a central configuration management system, and package updates do not overwrite your configured settings.

#### Note

When testing new or modified configurations, apply only the commands from a single configuration file at a time. Specify the name of the single configuration file on the `systemd-tmpfiles` command line.

#### References

`systemd-tmpfiles(8)`, `tmpfiles.d(5)`, `stat(1)`, `stat(2)`, and `systemd.timer(5)` man pages

## 4.4. Guided Exercise

### Manage Temporary Files

Use a timer unit to schedule a task that clears temporary files on the system.

#### Outcomes

- Configure `systemd-tmpfiles` to remove unused temporary files from the `/tmp` directory.
- Configure `systemd-tmpfiles` to periodically purge files from another directory.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start systasks-tempfiles
```

#### Instructions

- Log in to the `servera` system as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Configure the `systemd-tmpfiles` service to clean the `/tmp` directory of any unused files from the last five days. Ensure that a package update does not overwrite the configuration files.

- Copy the `/usr/lib/tmpfiles.d/tmp.conf` file to the `/etc/tmpfiles.d` directory.

```
[root@servera ~]# cp /usr/lib/tmpfiles.d/tmp.conf \
/etc/tmpfiles.d/tmp.conf
```

- Replace the existing age of the temporary files in the relevant configuration line with the new age of 5 days (5d).

Ensure that all other lines are removed from the file and that only the following line is present:

In the configuration, the q type is the same as the d type. By using the q or d type, the `systemd-tmpfiles` service creates the `/tmp` directory if it does not exist. The directory's octal permissions must be set to 1777. Both the owning user and group of the `/tmp` directory must be set to the `root` user. The `/tmp` directory must not contain any unused temporary files from the last five days.

```
q /tmp 1777 root root 5d
```

- 2.3.** Use the `systemd-tmpfiles --clean` command to verify the `/etc/tmpfiles.d/tmp.conf` file configuration. If the exit code is 0, then the configuration settings are correct.

```
[root@servera ~]# systemd-tmpfiles --clean /etc/tmpfiles.d/tmp.conf
[root@servera ~]# echo $?
0
```

- 3.** Add a new `/etc/tmpfiles.d/momentary.conf` configuration file that ensures that the `/run/momentary` directory exists, and that ownership is set to the `root` user and the `root` group. The octal permissions for the directory must be 0700. The configuration must purge from this directory any files that remain unused in the last 30 seconds.

- 3.1.** Create the `/etc/tmpfiles.d/momentary.conf` file with the following content.

```
d /run/momentary 0700 root root 30s
```

- 3.2.** Verify the `/etc/tmpfiles.d/momentary.conf` file configuration. The command creates the `/run/momentary` directory if it does not exist.

If the command does not return any errors, then the configuration settings are correct.

```
[root@servera ~]# systemd-tmpfiles --create /etc/tmpfiles.d/momentary.conf
[root@servera ~]# echo $?
0
```

- 3.3.** Verify that the `systemd-tmpfiles` command creates the `/run/momentary` directory with the appropriate permissions, owner, and group owner.

The octal permission for the `/run/momentary` directory is set to 0700, and the user and group ownership are set to `root`.

```
[root@servera ~]# ls -ld /run/momentary  
drwx-----. 2 root root 40 Apr  4 06:35 /run/momentary
```

- 4.** Verify that the `systemd-tmpfiles --clean` command removes any file from the `/run/momentary` directory that is unused in the last 30 seconds, based on the `systemd-tmpfiles` configuration for the directory.

- 4.1.** Create an empty `/run/momentary/test` file.

```
[root@servera ~]# touch /run/momentary/test
```

- 4.2.** Configure your shell prompt not to return for 30 seconds.

```
[root@servera ~]# sleep 30
```

- 4.3.** After your shell prompt returns, clean all stale files in the `/run/momentary` directory, based on the referenced rule in the `/etc/tmpfiles.d/momentary.conf` configuration file.

The command removes the `/run/momentary/test` file because it remains unused for 30 seconds. This behavior is based on the referenced rule in the `/etc/tmpfiles.d/momentary.conf` configuration file.

```
[root@servera ~]# systemd-tmpfiles --clean /etc/tmpfiles.d/momentary.conf
```

- 4.4.** Verify that the `/run/momentary/test` file does not exist.

```
[root@servera ~]# ls -l /run/momentary/test  
ls: cannot access '/run/momentary/test': No such file or directory
```

- 5.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

**Finish**

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish systasks-tempfiles
```

## 4.5. Scheduling Recurring System Tasks with Cron

### Objectives

- Schedule commands to run on a recurring schedule by using the system crontab file and the Cron directories.

### Recurring System Jobs

System administrators often need to run recurring jobs. It is best to run these jobs from system accounts rather than from user accounts. Schedule these jobs with system-wide crontab files instead of with the crontab command. Job entries in the system-wide crontab files are similar to the users' crontab entries. The system-wide crontab files have an extra field before the command field to specify the user that runs the command.

The /etc/crontab file has a syntax diagram in the comments.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue...
# | | | | |
# * * * * * user-name command to be executed
```

The /etc/crontab file and other files in the /etc/cron.d directory define the recurring system jobs. Always create custom crontab files in the /etc/cron.d directory to schedule recurring system jobs. Place the custom crontab file in the /etc/cron.d directory to prevent a package update from overwriting the /etc/crontab file. Packages that require recurring system jobs place their crontab files in the /etc/cron.d directory with the job entries. Administrators also use this location to group related jobs into a single file.

The crontab system also includes repositories for scripts to run every hour, day, week, or month. These repositories are placed in the /etc/cron.hourly, /etc/cron.daily, /etc/cron.weekly, or /etc/cron.monthly directories. These directories contain executable shell scripts, not crontab files.

### Note

Use the `chmod +x script_name` command to make a script executable. A script must be executable to run.

## Default System Crontab Files

Every script in the /etc/cron.hourly directory is a job that is run via the /etc/cron.d/0hourly secondary system crontab file. This crontab file instructs the crond daemon to execute the `run-parts` command on the /etc/cron.hourly directory, as the root user, at one minute past every hour. The `run-parts` command runs every executable file it finds inside the directory.

```
root@host:~# cat /etc/cron.d/0hourly
# Run the hourly jobs
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
01 * * * * root run-parts /etc/cron.hourly
```

## Creating a System Cron Job

To set up a system cron job, add a crontab file in the /etc/cron.d directory.

For example, the following crontab file named /etc/cron.d/timestamp\_example appends a temporary file named /tmp/timestamp\_example.txt with the output of the date command every two minutes.

```
# Example timestamp cronjob
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
*/2 * * * * root date >> /tmp/timestamp_example.txt
```

The \*/2 notation indicates a step value of two. In the minute field, this notation is interpreted as "every two minutes".

After several minutes, the `/tmp/timestamp_example.txt` file shows evidence of the date command being executed every two minutes:

```
root@host:~# cat /tmp/timestamp_example.txt
Fri May 30 15:48:01 UTC 2025
Fri May 30 15:50:01 UTC 2025
Fri May 30 15:52:01 UTC 2025
Fri May 30 15:54:01 UTC 2025
...output omitted...
```

### Important

Apart from comments, the `/etc/crontab` file is otherwise empty by default. Never edit the `/etc/crontab` file directly; use this file solely as a reference. Place individual crontab files in the `/etc/cron.d` directory instead.

System packages can also install related crontab files in the `/etc/cron.d` directory. To reduce the chance of a file name collision, include a string in the crontab file name that uniquely identifies it.

## Run Periodic Commands with Anacron

The `anacron` command uses the `run-parts` script to execute daily, weekly, or monthly jobs from the `/etc/anacrontab` configuration file on a recurring basis. Also, the `anacron` command is responsible for running standardized system jobs on predefined schedules.

Whereas the `cron` daemon can run time-sensitive tasks in systems that are presumed to be always running, the `anacron` command ensures that scheduled jobs always run and are not skipped accidentally because the system was turned off or hibernated.

For example, when a system job that runs daily was not executed at a specified time because the system was rebooting, then the job is completed when the system becomes ready. A delay might occur before the job starts, if specified in the `Delay in minutes` parameter in the `/etc/anacrontab` file.

Files in the `/var/spool/anacron` directory determine the daily, weekly, or monthly jobs. When the `cron` daemon starts a job from the `/etc/anacrontab` file, it updates the timestamps of those files. With this timestamp, you can determine the last time that the job executed. The syntax of the `/etc/anacrontab` file is different from the regular `crontab` configuration files.

The /etc/anacrontab file contains four fields per line, as shown in the following example:

```
# /etc/anacrontab: configuration file for anacron  
  
# See anacron(8) and anacrontab(5) for details.  
  
SHELL=/bin/sh  
PATH=/sbin:/bin:/usr/sbin:/usr/bin  
MAILTO=root  
# the maximal random delay added to the base delay of the jobs  
RANDOM_DELAY=45  
# the jobs will be started during the following hours only  
START_HOURS_RANGE=3-22  
  
#period in days    delay in minutes    job-identifier    command  
1      5            cron.daily         nice run-parts /etc/cron.daily  
7      25           cron.weekly        nice run-parts /etc/cron.weekly  
@monthly 45        cron.monthly       nice run-parts /etc/cron.monthly
```

### Period in days

Defines the interval in days for the job to run on a recurring schedule. This field accepts an integer or a macro value. For example, the @daily macro is equivalent to the 1 integer, which executes the job daily. Similarly, the @weekly macro is equivalent to the 7 integer, which executes the job weekly.

### Delay in minutes

Defines the time that the crond daemon must wait before it starts the job.

### Job identifier

Identifies the unique name of the job in the log messages.

### Command

The command to be executed.

The /etc/anacrontab file also contains environment variable declarations with the NAME=value syntax. The START\_HOURS\_RANGE variable specifies the time interval for the jobs to run. Jobs do not start outside this range. If a job does not run within this time interval on a particular day, then the job must wait until the next day for execution.

## Relationship Between Cron and Anacron

The `crond.service` command scheduler service and the `anacron` command work together as follows:

- Recall that the `/etc/cron.d` directory includes an `@hourly` crontab file that, at one minute past every hour, executes the `run-parts /etc/cron.hourly` command as the root user.
- The `/etc/cron.hourly` directory contains a shell script named `@anacron` that, in turn, conditionally runs the `/usr/sbin/anacron -s` command.
- When executed, the `/usr/sbin/anacron -s` command reads the `/etc/anacrontab` configuration file and executes the nice `run-parts` command on the `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly` files at particular time intervals, including a random delay of up to 45 minutes.

#### Note

The `crontabs` command is a legacy name for the `run-parts` script and the system crontab.

#### References

`crontab(5)`, `crontabs(4)`, `anacron(8)`, `anacrontab(5)`, and `crond(8)` man pages

## 4.6. Guided Exercise

### Schedule a Recurring System Task with Cron

Schedule a recurring system task by using the system crontab file in the Cron directory.

#### Outcomes

- Schedule a recurring system job to count the number of active users.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start systasks-syscron
```

#### Instructions

- Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Create a crontab file named usercount in the /etc/cron.d directory that generates a log message to indicate the number of active users in the system. Configure the crontab file to run as the root user every three minutes. Use the logger "There are `w -h | wc -l` active users" command to generate the log message. Confirm that the message is being logged by using the tail -f /var/log/messages | grep --line-buffered "There are" command.

- 2.1. Create the /etc/cron.d/usercount crontab file with the following content.

The `w -h | wc -l` command retrieves a list of active users in the system and then counts them.

```
# Log the number of active users
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
*/3 * * * * root logger "There are `w -h | wc -l` active users"
```

- 2.2.** Wait at least three minutes and confirm that the message is being logged. Use the `tail -f /var/log/messages | grep --line-buffered "There are"` command.

The number of active users might differ on your system.

```
[root@servera ~]# tail -f /var/log/messages |\
grep --line-buffered "There are"
May 30 17:39:01 workstation root[5750]: There are 4 active users
```

- 3.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish systasks-syscron
```

## 4.7. Quiz

### Scheduling User Tasks

---

Choose the correct answers to the following questions:

- 1.** Which directory is used to configure custom temporary locations, and to override vendor-provided defaults for the `systemd-tmpfiles-clean` service?  
 **a.** `/usr/lib/tmpfiles.d/`  
 **b.** `/var/tmpfiles.d/`  
 **c.** `/etc/tmpfiles.d/`  
 **d.** `/run/tmpfiles.d/`
  
- 2.** Which configuration file defines the settings for the system jobs that run daily, weekly, and monthly?  
 **a.** `/etc/crontab`  
 **b.** `/etc/anacrontab`  
 **c.** `/etc/inittab`  
 **d.** `/etc/sysconfig/crond`
  
- 3.** Which `systemd` unit regularly triggers the cleanup of temporary files?  
 **a.** `systemd-tmpfiles-clean.timer`  
 **b.** `systemd-tmpfiles-clean.service`  
 **c.** `dnf-makecache.timer`  
 **d.** `unbound-anchor.timer`
  
- 4.** Which command enables and starts a `systemd` unit in one step?  
 **a.** `systemd --enable --start`  
 **b.** `systemctl --now`  
 **c.** `systemctl enable --now`  
 **d.** `systemd enable --start`

**5.** Which command ensures that the systemd daemon loads a new configuration?

- a.** `sysctl reload`
- b.** `systemctl daemon --reload`
- c.** `systemd reload-daemon`
- d.** `systemctl daemon-reload`

**6.** Which directory are custom crontab files placed in?

- a.** `/etc/crontab.d`
- b.** `/etc/cron.d`
- c.** `/etc/cron.custom`
- d.** `/etc/crontab`

**7.** What crontab file notation indicates a step value of 7?

- a.** `*/7`
- b.** seven
- c.** 7
- d.** `7 * * * *`

## Solution

1. Which directory is used to configure custom temporary locations, and to override vendor-provided defaults for the `systemd-tmpfiles-clean` service?

- a. `/usr/lib/tmpfiles.d/`
- b. `/var/tmpfiles.d/`
- c. `/etc/tmpfiles.d/`
- d. `/run/tmpfiles.d/`

2. Which configuration file defines the settings for the system jobs that run daily, weekly, and monthly?

- a. `/etc/crontab`
- b. `/etc/anacrontab`
- c. `/etc/inittab`
- d. `/etc/sysconfig/crond`

3. Which `systemd` unit regularly triggers the cleanup of temporary files?

- a. `systemd-tmpfiles-clean.timer`
- b. `systemd-tmpfiles-clean.service`
- c. `dnf-makecache.timer`
- d. `unbound-anchor.timer`

4. Which command enables and starts a `systemd` unit in one step?

- a. `systemd --enable --start`
- b. `systemctl --now`
- c. `systemctl enable --now`
- d. `system.d enable --start`

5. Which command ensures that the `systemd` daemon loads a new configuration?

- a. `sysctl reload`
- b. `systemctl daemon --reload`
- c. `systemd reload-daemon`
- d. `systemctl daemon-reload`

**6.** Which directory are custom crontab files placed in?

- a.** ./etc/crontab.d
- b.** ./etc/cron.d
- c.** ./etc/cron.custom
- d.** ./etc/crontab

**7.** What crontab file notation indicates a step value of 7?

- a.** \*/7
- b.** seven
- c.** 7
- d.** 7 \* \* \* \*

## 4.8. Summary

---

In this lesson, you learned about methods that are used to run recurring system tasks automatically.

Most recurring system tasks now use systemd timer units. Timer units activate other types of units, usually service units, to carry out tasks at specific dates and times, or intervals.

The `systemd-tmpfiles` tool provides a structured way to manage temporary directories and files. Cleaning and maintaining temporary files is crucial to help prevent disk space issues and keep working data current.

The `crond` daemon monitors crontab files to run recurring, time-sensitive tasks on always-running systems. The `crond` daemon is also configured to use the `anacron` command to help ensure that scheduled jobs are not skipped due to the system not running at a specific time, and generally to run standardized system jobs on a preset schedule.

## Chapter 5.

### Analyzing and Storing Logs

---

#### Goal

Locate and interpret system logs for troubleshooting purposes, and ensure accurate timestamps for log events.

#### Sections

- The System Log Architecture (and Quiz)
- Interpreting and Managing Syslog Events (and Guided Exercise)
- Finding and Interpreting System Journal Log Entries (and Guided Exercise)
- Configuring a Persistent System Journal (and Guided Exercise)
- Maintaining Synchronized Time (and Guided Exercise)

#### Lab

- Analyze and Store Logs

## 5.1. The System Log Architecture

---

### Objectives

- Describe the primary mechanisms that log system events in Red Hat Enterprise Linux: Rsyslog and the system journal.

### System Logging

The operating system kernel and other processes record a log of events that happen when the system is running. These logs are used to audit the system and to troubleshoot problems. You can use text utilities such as the `less` and `tail` commands to inspect these logs.

Red Hat Enterprise Linux uses a standard logging system that is based on the `syslog` protocol to log the system messages. Many programs use the logging system to record events and to organize them into log files. The `systemd-journald` and `rsyslog` services handle the `syslog` messages in Red Hat Enterprise Linux.

The `systemd-journald` service is at the heart of the operating system event logging architecture.

The `systemd-journald` service collects event messages from many sources:

- System kernel
- Output from the early stages of the boot process
- Standard output and standard error from daemons
- The `syslog` events

The `systemd-journald` service restructures the logs into a standard format and writes them into a structured, indexed system journal. By default, this journal is stored on a file system that does not persist across reboots.

The `rsyslog` service reads the `syslog` messages that the `systemd-journald` service receives from the journal when they arrive. The `rsyslog` service then processes the `syslog` events, and records them to its log files or forwards them to other services according to its own configuration.

 **Note**

In RHEL 10, the operating system sends the local `syslog` messages to the `/dev/log` file, which is a symbolic link to a special socket file read by the `systemd-journald` service. Those messages are recorded to the system journal by the `systemd-journald` service.

Meanwhile, the `rsyslog` service does not read the `/dev/log` file directly. Instead, the `rsyslog` service uses its `imjournal` module to read the `syslog` messages from the system journal as they arrive, and then saves those log messages into files based on the configuration files.

The `rsyslog` service sorts and writes the `syslog` messages to the log files that do persist across reboots in the `/var/log` directory. The `rsyslog` service also sorts the `syslog` messages to specific log files according to the type of program that sent each message and the priority of each message.

In addition to the `syslog` message files, the `/var/log` directory contains log files from other services on the system.

The following table lists some useful files in the `/var/log` directory:

**Table 9. Selected System Log Files**

Log file	Type of stored messages
<code>/var/log/messages</code>	Most <code>syslog</code> messages are logged here. Exceptions include messages about authentication and email processing, scheduled job execution, and purely debugging-related messages.
<code>/var/log/secure</code>	Messages about security and authentication events
<code>/var/log/maillog</code>	Messages about the mail server
<code>/var/log/cron</code>	Messages about scheduled job execution

Log file	Type of stored messages
/var/log/boot.log	Console messages about system startup that are hidden by the graphical startup screen, and the syslog messages that are sent to the local7 facility.

The boot log stored in the /var/log/boot.log file can be populated from two sources.

The splash screen, which the Plymouth application provides and that is displayed when the system starts, and hides many diagnostic messages by default. Plymouth dumps these diagnostic messages to the /var/log/boot.log file.

The rsyslog service also saves any log messages that is sent to its local7 facility (typically messages about the boot process) as a line in the same /var/log/boot.log file.

Some applications do not use the syslog service to manage their log messages. For example, the Apache HTTP Server saves log messages to files in a subdirectory of the /var/log directory.

## References

`systemd-journald.service(8)`, `rsyslogd(8)`, and `rsyslog.conf(5)` man pages

For more information, refer to the *Troubleshooting Problems by Using Log Files* chapter of the *Red Hat Enterprise Linux 10 Risk Reduction and Recovery Operations* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/risk\\_reduction\\_and\\_recovery\\_operations/index#troubleshooting-problems-by-using-log-files](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/risk_reduction_and_recovery_operations/index#troubleshooting-problems-by-using-log-files)

## 5.2. Quiz

### The System Log Architecture

---

Choose the correct answers to the following questions:

- 1.** Which log file stores most syslog messages, except for the ones about authentication, mail, scheduled jobs, and debugging?

- a.** ./var/log/maillog
- b.** ./var/log/boot.log
- c.** ./var/log/messages
- d.** ./var/log/secure

- 2.** Which log file stores syslog messages about security and authentication operations in the system?

- a.** ./var/log/maillog
- b.** ./var/log/boot.log
- c.** ./var/log/messages
- d.** ./var/log/secure

- 3.** Which service sorts and organizes syslog messages into files in the /var/log directory?

- a.** rsyslog
- b.** systemd-journald
- c.** syslog
- d.** tuned

- 4.** Which directory accommodates the human-readable syslog files?

- a.** ./sys/kernel/debug
- b.** ./var/log/journal
- c.** ./run/log/journal
- d.** ./var/log

**5.** Which file stores syslog messages about the mail server?

- a.** /var/log/lastlog
- b.** /var/log/maillog
- c.** /var/log/tallylog
- d.** /var/log/boot.log

**6.** Which file stores syslog messages about scheduled jobs?

- a.** /var/log/cron
- b.** /var/log/tallylog
- c.** /var/log/spooler
- d.** /var/log/secure

**7.** Which file stores console messages about system startup?

- a.** /var/log/messages
- b.** /var/log/cron
- c.** /var/log/boot.log
- d.** /var/log/secure

## Solution

---

1. Which log file stores most syslog messages, except for the ones about authentication, mail, scheduled jobs, and debugging?  
 a. /var/log/maillog  
 b. /var/log/boot.log  
 c. /var/log/messages  
 d. /var/log/secure
2. Which log file stores syslog messages about security and authentication operations in the system?  
 a. /var/log/maillog  
 b. /var/log/boot.log  
 c. /var/log/messages  
 d. /var/log/secure
3. Which service sorts and organizes syslog messages into files in the /var/log directory?  
 a. rsyslog  
 b. systemd-journald  
 c. syslog  
 d. tuned
4. Which directory accommodates the human-readable syslog files?  
 a. /sys/kernel/debug  
 b. /var/log/journal  
 c. /run/log/journal  
 d. /var/log
5. Which file stores syslog messages about the mail server?  
 a. /var/log/lastlog  
 b. /var/log/maillog  
 c. /var/log/tallylog  
 d. /var/log/boot.log

**6.** Which file stores syslog messages about scheduled jobs?

- a. /var/log/cron
- b. /var/log/tallylog
- c. /var/log/spooler
- d. /var/log/secure

**7.** Which file stores console messages about system startup?

- a. /var/log/messages
- b. /var/log/cron
- c. /var/log/boot.log
- d. /var/log/secure

## 5.3. Interpreting and Managing Syslog Events

### Objectives

- Find the syslog log files, interpret the syslog messages, and configure which syslog messages are recorded to which files or other targets.

### Log Events to the System

Many programs use the syslog protocol to log events to the system. Each log message is categorized by *facility* (the subsystem that produces the message) and *priority* (the message's severity).

The following table lists the standard syslog facilities:

Table 10. Overview of Syslog Facilities

Code	Facility	Facility description
0	kern	Kernel messages
1	user	User-level messages
2	mail	Mail system messages
3	daemon	System daemon messages
4	auth	Authentication and security messages
5	syslog	Internal syslog messages
6	lpr	Printer messages
7	news	Network news messages

<b>Code</b>	<b>Facility</b>	<b>Facility description</b>
8	uucp	UUCP protocol messages
9	cron	Clock daemon messages
10	authpriv	Nonsystem authorization messages
11	ftp	FTP protocol messages
16-23	local0 to local7	Custom local messages

The following table lists the standard syslog priorities in descending order:

**Table 11. Overview of Syslog Priorities**

<b>Code</b>	<b>Priority</b>	<b>Priority description</b>
0	emerg	System is unusable
1	alert	Action must be taken immediately
2	crit	Critical condition
3	err	Noncritical error condition
4	warning	Warning condition
5	notice	Normal but significant event
6	info	Informational event

Code	Priority	Priority description
7	debug	Debugging-level message

## The Rsyslog Service

The `rsyslog` service implements the `syslog` logging. The `rsyslog` service uses the facility and priority of log messages to determine how to handle them.

*Rules* specify the facility and priority in the `/etc/rsyslog.conf` file and in any file in the `/etc/rsyslog.d` directory with the `.conf` extension. Software packages can add rules by installing an appropriate file in the `/etc/rsyslog.d` directory.

Each `rsyslog` rule consists of a filter, followed by an action.

*Filters* select a subset of the `syslog` messages by using various methods. For example, the facility/priority-based filter method indicates the facility and priority of the `syslog` messages that the rule matches. An asterisk (\*) can act as a wildcard to match all possible values of a facility or a priority.

*Actions* indicate how to handle messages that match the specified filter. Many possible actions are available, and a rule can have more than one action. For example, an action can indicate which file to save the log message in.

The following rule in the `/etc/rsyslog.conf` file saves messages that are sent to the `authpriv` facility of any priority to the `/var/log/secure` file:

```
authpriv.* /var/log/secure
```

If you specify a priority for the selector of a rule, then all messages to that facility at the specified priority or higher are logged to the action that is associated with the rule.

The following rule saves messages that are sent to the `authpriv` facility at the warning, notice, info, or debug priorities to the `/var/log/secure` file:

```
authpriv.warning /var/log/secure
```

Sometimes, log messages match more than one rule in the `rsyslog.conf` file. In such cases, the message is stored in more than one log file. The `none` keyword in the priority field indicates that no

messages for the indicated facility are stored in the given file, to limit stored messages.

Instead of being logged to a file, the `syslog` messages can also be printed to the terminals of all logged-in users. The `rsyslog.conf` file has a setting to print all the `syslog` messages with the `emerg` priority to the terminals of all logged-in users.

## Sample Rules of the Rsyslog Service

```
##### RULES #####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                         /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none        /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                       /var/log/secure

# Log all the mail messages in one place.
mail.*                                           -/var/log/maillog

# Log cron stuff
cron.*                                           /var/log/cron

# Everybody gets emergency messages
*. emerg                                         :omusrmmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                    /var/log/spooler

# Save boot messages also to boot.log
local7.*                                         /var/log/boot.log
```

### Note

The `syslog` subsystem has many more features beyond the scope of this course. To explore further, refer to the `rsyslog.conf(5)` man page and the extensive HTML documentation at `/usr/share/doc/rsyslog/html/index.html` that the `rsyslog-doc` package provides.

## Log File Rotation

The logrotate command is triggered by a systemd timer unit and rotates log files to prevent them from taking up too much space in the /var/log directory. When a log file is rotated, it is renamed with an extension that indicates the rotation date.

For example, the /var/log/messages file was renamed to the /var/log/messages-20250620 file when it was rotated on 2025-06-20. After the previous log file is rotated, a new log file is created and the log file's associated service is notified.

After approximately four weeks of log file rotations, the earliest log file is discarded to reclaim disk space.

A scheduled job runs the logrotate command daily to check the rotation requirements of various log files. Most log files are rotated weekly.

The logrotate command can rotate log files more quickly, or more slowly, or when the log files reach a specific size.

## Analyze a Log Entry

Log messages start with the earliest message at the start of the log file, and the latest message at the end of the log file. The rsyslog service uses a standard format for recording entries in log files.

The following example illustrates the anatomy of a log message in the /var/log/secure log file.

```
Mar 20 20:11:48 localhost sshd[1433]: Failed password for student from  
172.25.0.10 port 59344 ssh2
```

- **Mar 20 20:11:48:** The time stamp of the log entry
- **localhost:** The host that sent the log message
- **sshd[1433]:** The program or process name and PID number that sent the log message
- **Failed password for:** The message that was sent

## Monitor Log Events

Monitoring log files for events is helpful to reproduce issues. The tail -f /path/to/file command outputs the last ten lines of the specified file and continues to output newly written lines in the file.

For example, you can monitor login attempts by using a terminal session on two servers.

One terminal is for the hostA machine and the other terminal is for the hostB machine.

From the hostA machine in the first terminal, run the `tail -f` command and watch the `/var/log/secure` log file:

```
root@hostA:~# tail -f /var/log/secure
...output omitted...
```

From the hostB machine in the second terminal, run the `ssh` command and connect to the hostA machine as the `root` user:

```
root@hostB:~# ssh root@hostA
root@hostA's password: password
...output omitted...
root@hostA:~#
```

The log messages for the `root` user login from the hostB machine are visible in the first terminal on the hostA machine:

```
...output omitted...
Mar 20 09:01:13 hostA sshd[2712]: Accepted password for root from
172.25.254.254 port 56801 ssh2
Mar 20 09:01:13 hostA sshd[2712]: pam_unix(sshd:session): session opened for
user root by (uid=0)
```

## Send Log Messages Manually

The `logger` command sends messages to the `rsyslog` service. By default, the `logger` command sends the message of the `user` type with the `notice` priority (`user.notice`) unless otherwise specified with the `-p` option. The `logger` command is helpful for testing changes to the `rsyslog` service configuration.

For example, log messages that match the `local7.*` facility or priority filter are saved in the `/var/log/boot.log` file, based on a rule set in the default `/etc/rsyslog.conf` file. Therefore, to send a message to the `rsyslog` service and to record the message in the `/var/log/boot.log` log file, execute the following `logger` command:

```
root@host:~# logger -p local7.notice "Log entry created on host"
```

## Note

The `logger` command can also send structured journal entries to the system journal. For more information, see the documentation for the `--journald` option in the `logger(1)` man page.

## References

`logger(1)`, `tail(1)`, `rsyslog.conf(5)`, and `logrotate(8)` man pages

`rsyslog` Manual

- The HTML documentation at `/usr/share/doc/rsyslog/html/index.html`, which the `rsyslog-doc` package provides.

For more information, refer to the *Troubleshooting Problems by Using Log Files* chapter in the *Red Hat Enterprise Linux 10 Risk Reduction and Recovery Operations* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/risk\\_reduction\\_and\\_recovery\\_operations/index#troubleshooting-problems-by-using-log-files](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/risk_reduction_and_recovery_operations/index#troubleshooting-problems-by-using-log-files)

## 5.4. Guided Exercise

### Interpret and Manage Syslog Events

Configure the syslog log file to redirect all the log messages of a specified priority to a log file.

#### Outcomes

- Configure the syslog log file to use the rsyslog service to write all log messages with the debug priority to a log file.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start logs-syslog
```

#### Instructions

- Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Configure the rsyslog service to log all messages with the debug or higher priority for any services to the /var/log/messages-debug log file by changing the /etc/rsyslog.d/debug.conf configuration file.
  - Create the /etc/rsyslog.d/debug.conf file with the necessary entries to redirect all log messages with the debug or higher priority to the /var/log/messages-debug log file.

This configuration line logs the syslog messages with any facility and with the debug or higher priority level:

- The wildcard (\*) in the facility field of the configuration line indicates any facility of log messages.

- The rsyslog service writes the matching messages to the /var/log/messages-debug log file.

```
* .debug /var/log/messages-debug
```

## 2.2. Restart the rsyslog service.

```
[root@servera ~]# systemctl restart rsyslog
```

3. Verify that all the log messages with the debug priority appear in the /var/log/messages-debug log file.

### 3.1. Generate a log message with the user type and the debug priority.

```
[root@servera ~]# logger -p user.debug "Debug Message Test"
```

### 3.2. View the last 10 log messages from the /var/log/messages-debug log file, and verify that you see the Debug Message Test message among the other log messages.

```
[root@servera ~]# tail /var/log/messages-debug
Feb 13 18:22:38 servera systemd[1]: Stopping System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25176]: [origin software="rsyslogd"...
Feb 13 18:22:38 servera systemd[1]: Stopped System Logging Service.
Feb 13 18:22:38 servera systemd[1]: Starting System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25410]: environment variable TZ is...
Feb 13 18:22:38 servera systemd[1]: Started System Logging Service.
Feb 13 18:22:38 servera rsyslogd[25410]: [origin software="rsyslogd"...
Feb 13 18:27:58 servera root[25416]: Debug Message Test
```

4. Return to the workstation system as the student user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish logs-syslog
```

## 5.5. Finding and Interpreting System Journal Log Entries

### Objectives

- Find and interpret log entries in the system journal to review the system status.

### Find Events on the System Journal

The `systemd-journald` service stores logging data in a structured, indexed binary file that is known as a *journal*. This data includes extra information about the log event.

For example, for the `syslog` events, this information includes the priority of the original message, and the facility, which is a value that the `syslog` service assigns to track the process that originated a message.

#### Important

In Red Hat Enterprise Linux, the memory-based `/run/log` directory holds the system journal by default. The contents of the `/run/log` directory are lost when the system is shut down. You can change the `journald` directory to a persistent location, which is discussed later in this lesson.

To retrieve log messages from the journal, use the `journalctl` command. You can use the `journalctl` command to view all messages in the journal, or to search for specific events based on options and criteria. If you run the command as the `root` user, then you have full access to the journal. Although regular users can also use the `journalctl` command, the system restricts them from seeing certain messages.

```
root@host:~# journalctl  
...output omitted...  
Jun 09 12:56:12 localhost kernel: ACPI: Reserving FACP table memory at...  
Jun 09 12:56:12 localhost kernel: ACPI: Reserving DSDT table memory at...  
Jun 09 12:56:12 localhost kernel: ACPI: Reserving FACS table memory at...  
Jun 09 12:56:12 localhost kernel: ACPI: Reserving APIC table memory at...  
Jun 09 12:56:12 localhost kernel: ACPI: Reserving WAET table memory at...  
Jun 09 12:56:12 localhost kernel: No NUMA configuration found  
lines 1-44 [q]
```

The `journalctl` command highlights important log messages; messages with the notice or warning priority are in bold text, whereas messages with the error priority or higher are in red text.

The key to successful use of the journal for troubleshooting and auditing is to limit journal searches to show only relevant output.

By default, the `journalctl` command `-n` option shows the last 10 log entries. You can adjust the number of log entries with an optional argument that specifies how many log entries to display.

For example, to review the last five log entries, you can run the following `journalctl` command:

```
root@host:~# journalctl -n 5  
Jun 09 21:03:58 host sshd-session[9202]: Connection closed...  
Jun 09 21:04:00 host systemd[1]: status-api.service: Scheduled...  
Jun 09 21:04:00 host systemd[1]: Starting status-api.service...  
Jun 09 21:04:10 host sshd-session[9204]: Connection closed by...  
Jun 09 21:04:23 host sshd-session[9205]: Connection closed by...
```

Similar to the `tail` command, the `journalctl` command `-f` option outputs the last ten lines of the system journal and continues to output new journal entries when the journal appends them. To exit the `journalctl` command `-f` option, use the `Ctrl + C` key combination.

```
root@host:~# journalctl -f
Jun 09 21:05:01 host bash[9243]:    % Total    % Received %...
Jun 09 21:05:01 host bash[9243]:...
Jun 09 21:05:01 host bash[9243]: [158B blob data]
Jun 09 21:05:01 host bash[9243]: {
Jun 09 21:05:01 host bash[9243]:   "msg": "Status received and...
Jun 09 21:05:01 host bash[9243]: }
Jun 09 21:05:01 host systemd[1]: status-api.service: Deactivated successfully.
Jun 09 21:05:11 host sshd-session[9247]: Connection closed by...
Jun 09 21:05:24 host sshd-session[9255]: Connection closed by...
Jun 09 21:05:36 host sshd-session[9258]: Connection closed by...
Jun 09 21:05:48 host sshd-session[9260]: Connection closed by...
```

Ctrl + C

To help troubleshoot problems, you can filter the output of the journal by the priority of the journal entries. The `journalctl` command `-p` option shows the journal entries with a specified priority level (by name or by number) or higher. The `journalctl` command processes the `debug`, `info`, `notice`, `warning`, `err`, `crit`, `alert`, and `emerg` priority levels, in ascending priority order.

As an example, run the following `journalctl` command to list journal entries with the `err` priority or higher:

```
root@host:~# journalctl -p err
...output omitted...
Jun 09 12:56:23 workstation systemd[1206]: Failed to start app-gnome...
Jun 09 12:56:23 workstation systemd[1206]: Failed to start app-gnome...
Jun 09 12:56:23 workstation systemd[1206]: Failed to start app-gnome...
Jun 09 12:56:26 workstation (d-subman)[1493]: org.gnome.SettingsDaemon...
Jun 09 12:56:26 workstation systemd[1206]: Failed to start org.gnome...
...output omitted...
```

You can show messages for a specified systemd unit by using the `journalctl` command `-u` option and the unit name.

```
root@host:~# journalctl -u sshd.service
...output omitted...
Jun 09 12:56:22 host systemd[1]: Started sshd.service - OpenSSH...
Jun 09 12:56:31 host sshd-session[2078]: Connection closed by...
Jun 09 12:57:21 host systemd[1]: Stopping sshd.service - OpenSSH...
Jun 09 12:57:21 host sshd[1163]: Received signal 15; terminating.
Jun 09 12:57:21 host systemd[1]: sshd.service: Deactivated successfully.
Jun 09 12:57:21 host systemd[1]: Stopped sshd.service - OpenSSH server...
Jun 09 12:57:21 host systemd[1]: Starting sshd.service - OpenSSH server...
...output omitted...
lines 1-44  q
```

When using the `journalctl` command to find specific events, you can limit the output to a specific time range by using the `--since` option or the `--until` option. Both options take a time argument in the "YYYY-MM-DD hh:mm:ss" format. The double quotation marks are required to preserve the space within each option's value.

The `journalctl` command uses the 00:00:00 format when the time argument is omitted, and uses the current day when the day argument is omitted. Both `journalctl` options support `yesterday`, `today`, and `tomorrow` as valid arguments in addition to the date and time.

As an example, run the following `journalctl` command to list all journal entries from today's records:

```
root@host:~# journalctl --since today
...output omitted...
Jun 09 12:56:12 localhost kernel: clocksource: kvm-clock: mask: 0xffffffff...
Jun 09 12:56:12 localhost kernel: tsc: Detected 2494.172 MHz processor
Jun 09 12:56:12 localhost kernel: e820: update [mem 0x00000000-0x00000fff]...
Jun 09 12:56:12 localhost kernel: e820: remove [mem 0x000a0000-0x000fffff]...
Jun 09 12:56:12 localhost kernel: last_pfn = 0x1c0000 max_arch_pfn =...
Jun 09 12:56:12 localhost kernel: MTRR map: 4 entries (3 fixed + 1...
Jun 09 12:56:12 localhost kernel: x86/PAT: Configuration [0-7]: WB WC...
Jun 09 12:56:12 localhost kernel: last_pfn = 0xbffdb max_arch_pfn =...
Jun 09 12:56:12 localhost kernel: found SMP MP-table at [mem 0x000f5bc0-...
...output omitted...
lines 1996-2043/2043 (END)  q
```

Run the following `journalctl` command to list all journals since 2025-06-01 20:30 and until 2025-06-04 10:00:

```
root@host:~# journalctl --since "2025-06-01 20:30" --until "2025-06-04 10:00"  
...output omitted...
```

You can also specify all entries since a relative time until the present.

For example, to specify all entries since the last hour, you can use the following command:

```
root@host:~# journalctl --since "-1 hour"  
...output omitted...
```

#### Note

You can use other, more sophisticated time specifications with the `--since` and `--until` options. For some examples, see the `systemd.time(7)` man page.

In addition to the default journal output, you can view all the fields for each journal record by specifying the `verbose` option. Verbose output is useful to help find more specific information about journal events. You can also specify any field name and value to filter the journal output.

```
root@host:~# journalctl -o verbose  
Mon 2025-06-09 12:56:12.962826 UTC [s=ef6b97242c5140218987a8b063034712...  
_SOURCE_BOOTTIME_TIMESTAMP=0  
_SOURCE_MONOTONIC_TIMESTAMP=0  
_TRANSPORT=kernel  
PRIORITY=5  
SYSLOG_FACILITY=0  
SYSLOG_IDENTIFIER=kernel  
MESSAGE=Linux version 6.12.0-55.12.1.el10_0.x86_64 (mockbuild@...  
_BOOT_ID=d7f2dbc620504aa08585be2d04c4c596  
_MACHINE_ID=d676b38fd6a747d989b1e2dd98c0ffe8  
_HOSTNAME=localhost  
_RUNTIME_SCOPE=initrd  
...output omitted...  
lines 1-44 q
```

You can combine multiple system journal fields to form a granular search query with the `journalctl` command.

The following list shows some fields of the system journal to search for relevant lines to a particular process or event:

\_COMM

The command name

\_EXE

The path to the executable file for the process

\_PID

The PID of the process

\_UID

The UID of the user that runs the process

\_SYSTEMD\_UNIT

The systemd unit that started the process

For example, the following `journalctl` command shows all related journal entries to the `sshd.service` service with the 2188 PID:

```
root@host:~# journalctl _SYSTEMD_UNIT=sshd.service _PID=2188
Jun 09 12:57:21 workstation (sshd)[2188]: sshd.service: Referenced but...
Jun 09 12:57:21 workstation sshd[2188]: Server listening on 0.0.0.0 port 22.
Jun 09 12:57:21 workstation sshd[2188]: Server listening on :: port 22.
```

#### Note

For a list of journal fields, refer to the `systemd.journal-fields(7)` man page.

#### References

`journalctl(1)`, `systemd.journal-fields(7)`, and `systemd.time(7)` man pages

For more information, refer to the *Troubleshooting Problems by Using Log Files* chapter of the *Red Hat Enterprise Linux 10 Risk Reduction and Recovery Operations* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/risk\\_reduction\\_and\\_recovery\\_operations/index#troubleshooting-problems-by-using-log-files](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/risk_reduction_and_recovery_operations/index#troubleshooting-problems-by-using-log-files)

## 5.6. Guided Exercise

### Find and Interpret System Journal Log Entries

Find log entries of the system journal by using different criteria to review the system status.

#### Outcomes

- Search the system journal for entries to record events based on different criteria.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start logs-systemd
```

#### Instructions

- Log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- Use the journalctl command \_PID=1 option to display only log events that originate from the systemd PID 1 process. To quit from the journalctl command, press **q**.

The following output might differ on your system:

```
[student@servera ~]$ journalctl _PID=1
Jun 05 18:46:37 localhost systemd[1]: Finished systemd-sysctl.service...
Jun 05 18:46:37 localhost systemd[1]: Finished systemd-tmpfiles-setup-dev...
Jun 05 18:46:37 localhost systemd[1]: Starting systemd-sysusers.service...
Jun 05 18:46:38 localhost systemd[1]: Finished systemd-vconsole-setup...
Jun 05 18:46:38 localhost systemd[1]: dracut-cmdline-ask.service - dracut...
Jun 05 18:46:38 localhost systemd[1]: Starting dracut-cmdline.service...
Jun 05 18:46:38 localhost systemd[1]: Finished systemd-sysusers.service...
lines 1-5  q
```

- Use the journalctl command \_UID=81 option to display all log events that originated from a system service with a UID of 81.

```
[student@servera ~]$ journalctl _UID=81
Jun 05 18:46:47 servera.lab.example.com dbus-broker-launch[907]: Ready
```

4. Use the journalctl command -p warning option to display log events with the warning or a higher priority.

```
[student@servera ~]$ journalctl -p warning
Jun 05 18:46:37 localhost kernel: acpi PNP0A03:00: fail to add MMCONFIG...
Jun 05 18:46:38 localhost kernel: device-mapper: core: CONFIG_IMA+...
Jun 05 18:46:38 localhost rpc.idmapd[450]: Setting log level to 0
Jun 05 18:46:38 localhost rpc.idmapd[450]: libnfsidmap: Unable to
determine...
...output omitted...
Jun 05 18:46:47 servera.lab.example.com (qbalance)[935]:
irqbalance.service:...
Jun 05 18:46:47 servera.lab.example.com chronyd[955]: commandkey directive...
Jun 05 18:46:47 servera.lab.example.com chronyd[955]: generatecommandkey...
Jun 05 18:46:47 servera.lab.example.com chronyd[955]: Could not open
keyfile...
lines 1-14 q
```

5. Display all recorded log events in the past 10 minutes from the current time.

```
[student@servera ~]$ journalctl --since "-10min"
Jun 05 19:02:41 servera sshd-session[2261]: Connection closed by
172.25.250.254 port 57114
Jun 05 19:02:53 servera NetworkManager[1029]: <info> [1749150173.4104]...
Jun 05 19:02:53 servera NetworkManager[1029]: <info> [1749150173.4118]...
...output omitted...
Jun 05 19:03:38 servera NetworkManager[1029]: <info> [1749150218.4063]...
Jun 05 19:03:38 servera NetworkManager[1029]: <warn> [1749150218.4075]...
Jun 05 19:03:38 servera NetworkManager[1029]: <info> [1749150218.4077]...
lines 1-14 q
```

6. Use the journalctl command --since and \_SYSTEMD\_UNIT="sshd.service" options to display all the recorded log events that originated from the sshd service since 09:00:00 this morning.

 **Note**

Online classrooms typically run on the UTC time zone. To obtain results that start at 9:00:00 AM in your local time zone, adjust your --since value by the amount of your offset from UTC. Alternatively, ignore your local time and use a value of 9:00 to locate journal entries that occurred since 9:00:00 in the time zone (probably UTC) that the servera machine used.

```
[student@servera ~]$ journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"
Jun 05 18:47:54 servera (sshd)[1409]: sshd.service: Referenced but unset...
Jun 05 18:47:54 servera sshd[1409]: Server listening on 0.0.0.0 port 22.
Jun 05 18:47:54 servera sshd[1409]: Server listening on :: port 22.
Jun 05 18:48:02 servera sshd-session[1463]: Connection closed by 172.25....
Jun 05 18:48:14 servera sshd-session[1470]: Connection closed by 172.25....
Jun 05 18:48:39 servera sshd-session[1492]: Connection closed by 172.25....
Jun 05 18:48:51 servera sshd-session[1499]: Connection closed by 172.25....
...output omitted...
lines 1-14  q
```

7. Return to the workstation machine as the student user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish logs-systemd
```

## 5.7. Configuring a Persistent System Journal

### Objectives

- Configure a system journal to persistently store the log events after a reboot and to rotate the log events automatically.

### System Journal Storage

By default, Red Hat Enterprise Linux stores the system journal files in the `/run/log/journal` directory, and the system clears the system journal after a reboot. You can configure `systemd-journald` service to keep the journal files persistently, so that you can review journal events across reboots of the system.

### Configure a Persistent System Journal

Configure the `systemd-journald` service as follows to preserve system journals persistently across a reboot:

1. Create the `/var/log/journal` directory.

```
root@host:~# mkdir /var/log/journal
```

2. Run the `journalctl --flush` command to flush the current journal to storage. If the `systemd-journald` service successfully flushes the current journal, then the service creates subdirectories in the `/var/log/journal` directory.

```
root@host:~# journalctl --flush
```

3. The subdirectory in the `/var/log/journal` directory has hexadecimal characters in its long name and contains files with the `.journal` extension.

```
root@host:~# ls /var/log/journal  
4ec03abd2f7b40118b1b357f479b3112
```

4. The `.journal` binary files store structured and indexed journal entries.

```
root@host:~# ls /var/log/journal/4ec03abd2f7b40118b1b357f479b3112
system.journal  user-1000.journal
```

### Important

This service works because the default setting of the Storage parameter in the journal configuration is set to Storage=auto. This parameter means that if the `/var/log/journal` directory exists, then persistent storage is automatically enabled, but if the storage does not get enabled, then volatile storage in the `/run/log/journal` directory is used instead.

The journal configuration files are discussed in more detail later in this section.

## Review Journal Entries Relative to Boot Time

Although the system journals persist after a reboot, the `journalctl` command output includes entries from the current system boot as well as from the previous system boots. To limit the output to a specific system boot, use the `journalctl` command `-b` option.

The following `journalctl` command retrieves the entries from the first system boot only:

```
root@host:~# journalctl -b 1
...output omitted...
```

The following `journalctl` command retrieves the entries from the second system boot only. The argument is meaningful only if the system was rebooted at least twice:

```
root@host:~# journalctl -b 2
...output omitted...
```

You can list the system boot events that the `journalctl` command recognizes, by using the `--list-boots` option.

```
root@host:~# journalctl --list-boots
-6 27de... Wed 2025-06-04 20:04:32 EDT-Wed 2025-06-04 21:09:36 EDT
-5 6a18... Tue 2025-06-26 08:32:22 EDT-Thu 2025-06-26 16:02:33 EDT
-4 e2d7... Thu 2025-07-04 16:02:46 EDT-Fri 2025-07-04 20:59:29 EDT
-3 45c3... Sat 2025-07-05 11:19:47 EDT-Sat 2025-07-05 11:53:32 EDT
-2 dfae... Sat 2025-07-05 13:11:13 EDT-Sat 2025-07-05 13:27:26 EDT
-1 e754... Sat 2025-07-05 13:58:08 EDT-Sat 2025-07-05 14:10:53 EDT
  0 ee2c... Mon 2025-07-07 09:56:45 EDT-Mon 2025-07-07 12:57:21 EDT
```

The following `journalctl` command retrieves the entries from the current system boot only:

```
root@host:~# journalctl -b
...output omitted...
```

#### Note

When debugging a system crash with a persistent journal, you must limit the journal query to the reboot before the crash happened. You can use the `journalctl` command `-b` option with a negative number to indicate how many earlier system boots to include in the output.

For example, the `journalctl` command `-b -1` option limits the output to only the previous boot.

## System Journal Rotation

The advantage of persistent system journals is that the historical data is available immediately at boot. However, even with a persistent journal, the system does not keep all data forever.

The journal has a built-in log rotation mechanism that triggers monthly. In addition, the system does not allow the journals to get larger than 10% of the file system that they are on, or to leave less than 15% of the file system free. You can modify these values for both the runtime and persistent journals in the `/etc/systemd/journald.conf` configuration file. The `systemd-journald` process logs the current limits on the size of the journal when it starts.

The following command output shows the journal entries that reflect the current size limits:

```
[user@host ~]$ journalctl | grep -E 'Runtime Journal|System Journal'
Jun 10 22:22:57 localhost systemd-journald[263]: Runtime Journal
(/run/log/journal/93c43a50cada46e39819e0da68f790a2) is 4.2M, max 34.1M, 29.8M
free.
Jun 10 22:23:01 host systemd-journald[740]: Runtime Journal
(/run/log/journal/da54d37671ac4429bcdfacfe7bebb5a1) is 4.2M, max 34.1M, 29.8M
free.
Jun 10 22:30:50 host systemd-journald[740]: System Journal
(/var/log/journal/da54d37671ac4429bcdfacfe7bebb5a1) is 8M, max 997.3M, 989.3M
free.
...output omitted...
```

### Note

In the previous grep command, the vertical bar (|) symbol in the expression acts as an or operator. That is, the grep command matches any line with either the `Runtime Journal` string or the `System Journal` string from the `journalctl` command output. This command fetches the current size limits on the volatile (Runtime) journal store and on the persistent (System) journal store.

## Locating Journal Configuration Files

System administrators normally configure journal settings by editing the `/etc/systemd/journald.conf` configuration file, or by adding configuration files with a `.conf` suffix to the `/etc/systemd/journald.conf.d` directory.

However, a RHEL system might not have those files or that directory. In that case, lower-priority locations that are documented in the `journal.conf(5)` man page are checked for settings, and if none of those files or directories exist, then compiled-in default settings are used.

### Important

Starting from RHEL 10, the `/etc/systemd/journald.conf` file is not present at installation time. The `systemd-journald` service reads its default settings from the `/usr/lib/systemd/journald.conf` configuration file.

To change the journal's behavior, do not edit the `/usr/lib/systemd/journald.conf` configuration file directly. Instead, copy the `/usr/lib/systemd/journald.conf` file to the `/etc/systemd` directory, and then uncomment and change the appropriate settings in the resulting `/etc/systemd/journald.conf` file. Configuration settings in the `/etc/systemd/journald.conf` file take precedence over the default settings in the `/usr/lib/systemd/journald.conf` file.

## Configure Automatic Journal Rotation

To configure automatic journal rotation for persistent or runtime-only journals, set the following parameters in the `/etc/systemd/journald.conf` file.

Prefix the settings with `System` for persistent journals in the `/var/log/journal` directory.

Prefix the settings with `Runtime` for volatile journals in the `/run/log/journal` directory.

### SystemMaxUse and RuntimeMaxUse

The maximum amount of file system space that the journal can use. This value is 10% of the total file system space by default, and is capped at 4 GB.

### SystemMaxFileSize and RuntimeMaxFileSize

The maximum file size of a journal before it rotates. This value is one eighth of the `SystemMaxUse/RuntimeMaxUse` value by default, and is capped at 128 M, and usually supports keeping seven rotated journal files as history. If journal compact mode is enabled (the default), this value is capped at 4 GB.

### SystemKeepFree and RuntimeKeepFree

The minimum free file system space that must remain before archived journals are dropped. At least 15% of the total file system space is always kept free by default. This value is capped at 4 GB.

### Important

Only files with the `.journal` or `.journal~` suffixes are considered journal files.

After editing the `/etc/systemd/journald.conf` configuration file, restart the `systemd-journald` service to apply the configuration changes.

```
root@host:~# systemctl restart systemd-journald
```

## References

`journald.conf(5)` and `systemd-journald.service(8)` man pages

For more information, refer to the *Troubleshooting Problems by Using Log Files* chapter of the *Red Hat Enterprise Linux 10 Risk Reduction and Recovery Operations* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/risk\\_reduction\\_and\\_recovery\\_operations/index#troubleshooting-problems-by-using-log-files](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/risk_reduction_and_recovery_operations/index#troubleshooting-problems-by-using-log-files)

## 5.8. Guided Exercise

### Configure a Persistent System Journal

Configure a system journal to make the log events persist after a reboot.

#### Outcomes

- Configure a system journal to preserve the log events after a reboot.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start logs-preserve
```

#### Instructions

- Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Confirm that the /var/log/journal directory does not exist.

Because the /var/log/journal directory does not exist, the `systemd-journald` service does not preserve the log events after a reboot.

```
[root@servera ~]# ls /var/log/journal
ls: cannot access '/var/log/journal': No such file or directory
```

- Configure the `systemd-journald` service to preserve journals after a reboot.

- Create the /var/log/journal directory.

```
[root@servera ~]# mkdir /var/log/journal
```

**3.2.** Flush the current journal to storage.

```
[root@servera ~]# journalctl --flush
```

- 4.** Verify that the `systemd-journald` service preserves its journals so that they persist after a reboot.

**4.1.** Restart the `servera` machine.

The SSH connection terminates as soon as you restart the `servera` machine.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
student@workstation:~$
```

- 4.2.** Wait for the `servera` machine to reboot, and then log in to the `servera` machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 4.3.** Verify that a subdirectory with a long hexadecimal name exists in the `/var/log/journal` directory. You can find the journal files in that directory. The subdirectory name on your system might be different.

```
[student@servera ~]$ ls /var/log/journal
[sudo] password for student: student
63b272eae8d5443ca7aaa5593479b25f
```

- 4.4.** Check the contents of the hexadecimal directory.

```
[student@servera ~]$ ls /var/log/journal/63b272eae8d5443ca7aaa5593479b25f
system.journal  user-1000.journal
```

- 5.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish logs-preserve
```

## 5.9. Maintaining Synchronized Time

### Objectives

- Maintain accurate time synchronization by using the Network Time Protocol (NTP), and configure the system time zone to ensure correct timestamps for logged events.

### Administer Local Clocks and Time Zones

System time synchronization is critical for log file analysis across multiple systems. To work correctly, some services might also require correct time synchronization. Machines use NTP to provide and obtain correct time information over the internet. A machine might get accurate time information from public NTP services, such as the NTP Pool Project. Another option is to configure a system to synchronize directly with a high-quality hardware clock, so that it can provide accurate time information to local clients.

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

```
user@host:~$ timedatectl
          Local time: Wed 2022-03-16 05:53:05 EDT
          Universal time: Wed 2022-03-16 09:53:05 UTC
                  RTC time: Wed 2022-03-16 09:53:05
                    Time zone: America/New_York (EDT, -0400)
      System clock synchronized: yes
        NTP service: active
       RTC in local TZ: no
```

### Configuring the System Time Zone

You can list a database of time zones with the `timedatectl` command `list-timezones` option:

```
user@host:~$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
...output omitted...
```

The Internet Assigned Numbers Authority (IANA) provides a public time zone database, and the `timedatectl` command bases the time zone names on that database. IANA names time zones based on the continent or ocean, and then typically the largest city within the time zone region. For example, most of the US Mountain time zone is `America/Denver`.

Some localities inside the time zone have different daylight saving time rules. For example, in the US, much of the state of Arizona is in the US Mountain time zone; however it does not observe daylight saving time. Therefore, much of the state of Arizona uses the `America/Phoenix` time zone, which correctly accounts for the time difference.

Use the `tzselect` command to identify the correct time zone name. This command interactively prompts the user with questions about the system's location, and outputs the name of the correct time zone. It does not change the system's time zone setting.

The root user can change the system setting to update the current time zone with the `timedatectl` command `set-timezone` option.

For example, the following `timedatectl` command updates the current time zone to `America/Phoenix`.

```
root@host:~# timedatectl set-timezone America/Phoenix
root@host:~# timedatectl
    Local time: Wed 2025-04-02 03:05:55 MST
    Universal time: Wed 2025-04-02 10:05:55 UTC
        RTC time: Wed 2025-04-02 10:05:55
        Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
    NTP service: active
    RTC in local TZ: no
```

### Note

You can set a server's time zone to Coordinated Universal Time (UTC). The `tzselect` command does not include the name of the UTC time zone. Use the `timedatectl` command `set-timezone` option to set the system's current time zone to UTC.

## Manually Configuring System Time

Use the `timedatectl` command `set-time` option to change the system's current time. You might specify the time in the "YYYY-MM-DD hh:mm:ss" format, where you can omit either the date or the time.

For example, the following `timedatectl` command changes the local system time to `09:00:00`.

```
root@host:~# timedatectl set-time 9:00:00
root@host:~# timedatectl
    Local time: Wed 2025-04-02 09:00:27 MST
    Universal time: Wed 2025-04-02 16:00:27 UTC
        RTC time: Wed 2025-04-02 16:00:27
        Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
    NTP service: active
    RTC in local TZ: no
```

### Important

The previous example might fail with the `Failed to set time: Automatic time synchronization is enabled` error message. In that case, first disable the automatic time synchronization before manually setting the date or time, which is explained after this note.

The `timedatectl` command `set-ntp` option enables or disables NTP synchronization for automatic time changes. The `set-ntp` option requires either a `true` or a `false` argument to turn NTP synchronization on or off.

For example, the following `timedatectl` command turns off NTP synchronization.

```
root@host:~# timedatectl set-ntp false
```

#### Note

In Red Hat Enterprise Linux, the `timedatectl` command `set-ntp` option adjusts whether the `chronyd` NTP service is enabled. Other Linux distributions might use this setting to adjust a different NTP service or a Simple Network Time Protocol (SNTP) service.

Enabling or disabling NTP with other utilities in RHEL, such as in the graphical GNOME Settings application (which the `gnome-control-center` command provides), also updates this setting.

## Configure and Monitor the `Chronyd` Service

The `chronyd` service synchronizes the inaccurate local Real-Time Clock (RTC) with the configured NTP servers. If no network connectivity is available, then the `chronyd` service calculates the RTC clock drift, and records it in the `/var/lib/chrony/drift` file (which the `driftfile` setting specifies in the `/etc/chrony.conf` configuration file).

NTP servers can also be NTP clients. An NTP server can get its time information from high-performance reference clock hardware (which is typically based on an atomic clock or a GPS signal), or from an NTP server that might have access to such hardware.

The *stratum* of the NTP time source indicates its proximity to a time source. The stratum determines the number of hops that the machine is away from a high-performance reference clock. The reference clock is a "stratum 0" time source. An NTP server that is directly attached to the reference clock is a "stratum 1" time source. A machine that synchronizes time from a stratum 1 NTP server is a "stratum 2" time source, and so on.

## Selecting NTP Time Sources

Many organizations do not run their own stratum 1 NTP servers. High-performance reference clocks can be expensive, and many public time sources are available on the internet. It is often simpler to synchronize time from public NTP sources.

 **Note**

This section does not cover how to directly connect an NTP server to high-performance reference clock hardware, only to other NTP servers that might have access to such a clock. For more information about hardware reference clock configuration, read the documentation for the `refclock` directive on the `chrony.conf(5)` man page.

If you are not running a stratum 1 server, then must configure your NTP server to get its time from another NTP time source. Although you can get the time from a single source, it is better to configure three or more sources if one of the sources is malfunctioning, and to better correct for variable network lag.

The following key directives add higher-stratum servers to your `/etc/chrony.conf` configuration file:

- The `server` directive adds a single NTP server (you can use this directive more than once).
- The `pool` directive specifies a DNS name that resolves to multiple NTP servers, and that list can change over time.

By default, the `chronyd` service uses a pool of servers from the NTP Pool Project to synchronize time and requires no additional configuration. The `pool 2.rhel.pool.ntp.org iburst` line in the `/etc/chrony.conf` configuration file configures this service.

 **Note**

The `2.rhel.pool.ntp.org` name resolves to the IPv4 or IPv6 addresses of four public NTP servers. These addresses are selected at random from a larger pool of public NTP servers.

If your machines run on an isolated network so that the pool servers cannot be reached, then you must configure your own NTP servers.

The `server` directive in the `/etc/chronyd.conf` file specifies a single NTP server. You can define multiple servers in the `chrony` configuration file, one per line.

The first argument of the `server` line is the IP address or DNS name of an NTP server. Following the server IP address or name, you can list a series of options for the server. Red Hat recommends adding the `iburst` option to this directive, which configures `chronyd` to make 4 to 8 quick measurements so

that it can synchronize the clock more quickly. For more information about the chronyd configuration file options, read the `chrony.conf(5)` man page.

As an example, if the `server ntp.example.com iburst` line is in the `/etc/chrony.conf` configuration file, the chronyd service uses the `ntp.example.com` server as an NTP time source.

```
# Use public servers from the pool.ntp.org project.
pool 2.rhel.pool.ntp.org iburst
server ntp.example.com iburst
...output omitted...
```

Restart the service after pointing the chronyd service to the new time source.

```
root@host:~# systemctl restart chronyd
```

## Inspecting NTP Synchronization Status

The `chronyc` command acts as a client to the `chronyd` service. After setting up NTP synchronization, verify that the local system is seamlessly using the new NTP servers to synchronize the system clock, by using the `chronyc` command `sources` option. For more verbose output with additional explanations about the output, use the `chronyc` command `sources -v` option.

```
root@host:~# chronyc sources -v

    .-- Source mode '^' = server, '=' = peer, '#' = local clock.
    / .- Source state '*' = current best, '+' = combined, '-' = not combined,
    | /           'x' = may be in error, '~' = too variable, '?' = unusable.
    ||           .- xxxx [ yyyy ] +/- zzzz
    ||           xxxx = adjusted offset,
    ||           yyyy = measured offset,
    ||           zzzz = estimated error.
    ||
    MS Name/IP address      Stratum Poll  Reach LastRx Last sample
=====
^* ntp.example.com        2      6    377     20    -28us[ +361ns] +/-   14ms
^- 192.0.2.10              2     10    377    936   -2329us[ -2329us] +/- 120ms
^+ 2001:0db8:5:1890:f888:3e>  2     10    377   1009   -777us[ -673us] +/-   27ms
^+ clock3.example.net     2     10    377    149   -1220us[ -1220us] +/-   36ms
^- 2001:0db8:88:f031:3333:a> 2     10    377    767   -1300us[ -1300us] +/- 118ms
```

The asterisk character (\*) in the Source state (MS) field indicates that the `chronyd` service uses the `ntp.example.com` server as its best current time source that is selected for synchronization.

Sources with the plus character (+) in that field are also contributing to synchronization.

Sources with the minus character (-) in that field are usable for synchronization but are not currently contributing to it (probably better sources are present).

If an x or a tilde (~) is in that field, then the chronyd service determined that something seems wrong with the time that this remote NTP server provided.

If a question mark (?) is in that field, then the server cannot currently be used. It takes a few queries before a new server becomes usable, which is why the `iburst` option is helpful to quickly start time synchronization. If a server stays in the ? state for a long time, then it might be down or unreachable.

#### Note

A value in the Reach column of less than 377 means that fewer than eight of the most recent eight polls were successful, which can help diagnose servers in the ? state. Refer to the section of the `chronyc(1)` man page about the `sources` subcommand for details.

In the Name/IP address column, if an IPv6 address is long, it is truncated, and the end of the address is marked with a greater-than (>) character. To display the entire address, run the `chronyc` command `-n sources` option, which turns off DNS name resolution and displays untruncated IP addresses. This command can make the output of the command more difficult to read.

#### Important

You must specify the `-n` option between the `chronyc` and `sources` commands to work as expected.

#### References

[timedatectl\(1\)](#), [tzselect\(8\)](#), [chronyd\(8\)](#), [chrony.conf\(5\)](#), and [chronyc\(1\)](#) man pages

[Are the rhel.pool.ntp.org NTP Servers Supported by Red Hat?](#)

[NTP Pool Project](#)

[Time Zone Database](#)

## 5.10. Guided Exercise

### Maintain Synchronized Time

---

Configure the system to synchronize the time zone by using the NTP.

#### Outcomes

- Change the time zone on a server.
- Configure the server to synchronize its time with an NTP time source.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start logs-maintain
```

#### Instructions

1. Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
```

2. For this exercise, pretend that the servera machine is relocated to Haiti and that you need to update the time zone.

- 2.1. Select the appropriate 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
3) Antarctica
4) Asia
5) Atlantic Ocean
6) Australia
7) Europe
8) Indian Ocean
9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the timezone using the Posix TZ format.
#? 2
Please select a country whose clocks agree with yours.
1) Anguilla          19) Dominican Republic 37) Peru
2) Antigua & Barbuda 20) Ecuador           38) Puerto Rico
3) Argentina         21) El Salvador        39) St Barthelemy
4) Aruba             22) French Guiana    40) St Kitts & Nevis
5) Bahamas           23) Greenland         41) St Lucia
6) Barbados          24) Grenada           42) St Maarten (Dutch)
7) Belize            25) Guadeloupe       43) St Martin (French)
8) Bolivia           26) Guatemala        44) St Pierre & Miquelon
9) Brazil             27) Guyana            45) St Vincent
10) Canada           28) Haiti              46) Suriname
11) Caribbean NL     29) Honduras          47) Trinidad & Tobago
12) Cayman Islands   30) Jamaica           48) Turks & Caicos Is
13) Chile             31) Martinique       49) United States
14) Colombia          32) Mexico            50) Uruguay
15) Costa Rica        33)Montserrat      51) Venezuela
16) Cuba              34) Nicaragua        52) Virgin Islands (UK)
17) Curaçao          35) Panama           53) Virgin Islands (US)
18) Dominica          36) Paraguay
#? 28
```

The following information has been given:

Haiti

Therefore TZ='America/Port-au-Prince' will be used.  
Selected time is now: Fri Jun 6 13:49:36 EDT 2025.  
Universal Time is now: Fri Jun 6 17:49:36 UTC 2025.  
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
```

## 2.2. Update the America/Port-au-Prince time zone.

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

## 2.3. Verify that you correctly set the time zone to America/Port-au-Prince.

```
[root@servera ~]# timedatectl  
    Local time: Fri 2025-06-06 13:50:22 EDT  
    Universal time: Fri 2025-06-06 17:50:22 UTC  
        RTC time: Fri 2025-06-06 17:50:22  
      Time zone: America/Port-au-Prince (EDT, -0400)  
System clock synchronized: yes  
      NTP service: active  
    RTC in local TZ: no
```

## 3. Configure the chronyd service to synchronize the system time with the classroom.example.com server as the NTP time source.

### 3.1. Edit the /etc/chrony.conf configuration file to specify the classroom.example.com server as the NTP time source. Include the iburst option to speed up initial time synchronization:

```
...output omitted...  
server classroom.example.com iburst  
...output omitted...
```

## 3.2. Use the timedatectl command to enable time synchronization.

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

## 4. Verify that the NTP configuration synchronizes with the classroom.example.com time source in the classroom environment.

### 4.1. Verify that time synchronization is enabled.

 Note

If the output shows that the clock is not synchronized, then wait for a few seconds and rerun the `timedatectl` command. It takes a few seconds to successfully synchronize the time settings with the time source.

```
[root@servera ~]# timedatectl
          Local time: Fri 2025-06-06 13:51:13 EDT
          Universal time: Fri 2025-06-06 17:51:13 UTC
                  RTC time: Fri 2025-06-06 17:51:13
                 Time zone: America/Port-au-Prince (EDT, -0400)
System clock synchronized: yes
          NTP service: active
    RTC in local TZ: no
```

- 4.2.** Verify that the servera machine currently synchronizes its time settings with the classroom.example.com time source.

The output shows an asterisk character (\*) in the source state field (MS) for the classroom.example.com NTP time source. The asterisk indicates that the local system time successfully synchronizes with the NTP time source.

```
[root@servera ~]# chronyc sources -v
      .-- Source mode '^' = server, '=' = peer, '#' = local clock.
      / .- Source state '+*' = current best, '' = combined, '-' = not combined,
      | /           'x' = may be in error, '~' = too variable, '?' = unusable.
      ||           .- xxxx [ yyyy ] +/- zzzz
      ||           |   xxxx = adjusted offset,
      ||           |   yyyy = measured offset,
      ||           |   zzzz = estimated error.
      ||           |   |
      ||           |   |
      MS Name/IP address      Stratum Poll Reach LastRx Last sample
      =====
      ** classroom.example.com      3     9    377    469    +107us[ +130us] +/- 13ms
```

- 5.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish logs-maintain
```

## 5.11. Lab

### Analyze and Store Logs

---

Configure the system to update the time zone.

Configure the system journal to persistently store the event logs after a reboot.

Configure the syslog service to redirect all the log events of a specific priority to a file.

### Outcomes

- Update the time zone on an existing server.
- Configure system journals to persist to storage and to rotate more often.
- Configure a new log file to store all messages for authentication failures.

### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start logs-review
```

### Instructions

1. Pretend that the `serverb` machine is relocated to Jamaica and that you must update to the America/Jamaica time zone. Elevate the privileges of the student user to run the `timedatectl` command to update the time zone. Use `student` as the password.  
Verify that you correctly set the appropriate time zone.
2. View the recorded log events in the previous 30 minutes.
3. As the root user, configure persistent journals with automatic rotation. Verify that the journal is persistent after rebooting.
4. Create the `/etc/rsyslog.d/auth-errors.conf` file. Configure the `rsyslog` service to write messages to the `/var/log/auth-errors` file. Use the `authpriv` facility with the `alert` priority in the configuration file.

Enter the Logging test authpriv.alert log message in the system log by using the authpriv facility with the alert priority. Use student as the password.

5. Return to the workstation system as the student user.

## Evaluation

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade logs-review
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish logs-review
```

## Solution

---

Configure the system to update the time zone.

Configure the system journal to persistently store the event logs after a reboot.

Configure the syslog service to redirect all the log events of a specific priority to a file.

## Outcomes

- Update the time zone on an existing server.
- Configure system journals to persist to storage and to rotate more often.
- Configure a new log file to store all messages for authentication failures.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start logs-review
```

## Instructions

1. Pretend that the serverb machine is relocated to Jamaica and that you must update to the America/Jamaica time zone. Elevate the privileges of the student user to run the timedatectl command to update the time zone. Use student as the password.

Verify that you correctly set the appropriate time zone.

- 1.1. Log in to the serverb machine as the student user.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. Select the appropriate time zone for Jamaica.

```
[student@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
3) Antarctica
4) Asia
5) Atlantic Ocean
6) Australia
7) Europe
8) Indian Ocean
9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the timezone using the Posix TZ format.
#? 2
Please select a country whose clocks agree with yours.
1) Anguilla          19) Dominican Republic   37) Peru
2) Antigua & Barbuda 20) Ecuador           38) Puerto Rico
3) Argentina         21) El Salvador        39) St Barthelemy
4) Aruba             22) French Guiana      40) St Kitts & Nevis
5) Bahamas           23) Greenland          41) St Lucia
6) Barbados          24) Grenada            42) St Maarten (Dutch)
7) Belize             25) Guadeloupe        43) St Martin (French)
8) Bolivia            26) Guatemala          44) St Pierre & Miquelon
9) Brazil             27) Guyana             45) St Vincent
10) Canada            28) Haiti               46) Suriname
11) Caribbean NL     29) Honduras           47) Trinidad & Tobago
12) Cayman Islands    30) Jamaica            48) Turks & Caicos Is
13) Chile              31) Martinique        49) United States
14) Colombia          32) Mexico              50) Uruguay
15) Costa Rica         33)Montserrat        51) Venezuela
16) Cuba                34) Nicaragua          52) Virgin Islands (UK)
17) Curaçao           35) Panama             53) Virgin Islands (US)
18) Dominica          36) Paraguay
#? 30
```

The following information has been given:

Jamaica

Therefore TZ='America/Jamaica' will be used.

Selected time is now: Mon Jun 9 11:50:06 EST 2025.

Universal Time is now: Mon Jun 9 16:50:06 UTC 2025.

Is the above information OK?

- 1) Yes
  - 2) No
- #? 1

You can make this change permanent for yourself by appending the line

```
TZ='America/Jamaica'; 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/Jamaica
```

- 1.3.** Elevate the student user privileges to update the time zone to America/Jamaica. Use student as the password.

```
[student@serverb ~]$ sudo timedatectl set-timezone America/Jamaica  
[sudo] password for student: student  
[student@serverb ~]$
```

- 1.4.** Verify that you successfully set the time zone to America/Jamaica.

```
[student@serverb ~]$ timedatectl  
          Local time: Mon 2025-06-09 11:51:02 EST  
          Universal time: Mon 2025-06-09 16:51:02 UTC  
             RTC time: Mon 2025-06-09 16:51:02  
       Time zone: America/Jamaica (EST, -0500)  
System clock synchronized: yes  
      NTP service: active  
    RTC in local TZ: no
```

- 2.** View the recorded log events in the previous 30 minutes.

```
[student@serverb ~]$ journalctl --since "-30min"  
...output omitted...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4336] ...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4372] ...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4372] ...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4372] ...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4384] ...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4386] ...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4387] ...  
Jun 09 11:25:23 serverb NetworkManager[1034]: <info> [1749486323.4389] ...  
lines 1-44/44 q
```

- 3.** As the root user, configure persistent journals with automatic rotation. Verify that the journal is persistent after rebooting.

- 3.1.** Switch to the root user. Use student as the password.

```
[student@serverb ~]$ sudo -i  
[sudo] password for student: student  
[root@serverb ~]#
```

**3.2.** Create the /var/log/journal directory.

```
[root@serverb ~]# mkdir /var/log/journal
```

**3.3.** Flush the current journal to storage.

```
[root@serverb ~]# journalctl --flush
```

**3.4.** Restart the systemd-journald service to apply the configuration changes.

```
[root@serverb ~]# systemctl restart systemd-journald
```

**3.5.** Restart the serverb machine.

The SSH connection terminates as soon as you restart the serverb machine.

```
[root@serverb ~]# systemctl reboot  
...output omitted...  
Connection to serverb closed.  
student@workstation:~$
```

**3.6.** Wait for the serverb machine to reboot, and then log in to the serverb machine as the student user.

```
student@workstation:~$ ssh student@serverb  
...output omitted...  
[student@serverb ~]$
```

**3.7.** Verify that the journal files are persistently stored to local storage. The journal subdirectory name might be different from what is shown here.

```
[student@serverb ~]$ ls -l /var/log/journal  
total 4  
... root systemd-journal 53 Jun 10 22:30 da54d37671ac4429bcdfacfe7bebb5a1
```

**3.8.** Check the contents of the hexadecimal directory.

```
[student@serverb ~]$ ls -l /var/log/journal/da54d37671ac4429bcdfacfe7bebb5a1
total 16388
... 1 root systemd-journal 8388608 Jun 10 22:35 system.journal
... 1 root root          8388608 Jun 10 22:35 user-1000.journal
```

- 4.** Create the `/etc/rsyslog.d/auth-errors.conf` file. Configure the `rsyslog` service to write messages to the `/var/log/auth-errors` file. Use the `authpriv` facility with the `alert` priority in the configuration file.

Enter the `Logging test authpriv.alert` log message in the system log by using the `authpriv` facility with the `alert` priority. Use `student` as the password.

- 4.1.** Switch to the `root` user. Use `student` as the password.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 4.2.** Create the `/etc/rsyslog.d/auth-errors.conf` file by using the `sudo` command. Use `student` as the password.

Specify the new `/var/log/auth-errors` file as the destination for authentication and security messages. The file must appear as follows:

```
authpriv.alert /var/log/auth-errors
```

- 4.3.** Restart the `rsyslog` service to apply the configuration file changes.

```
[root@serverb ~]# systemctl restart rsyslog
```

- 4.4.** Use the `logger -p` command to write the `Logging test authpriv.alert` message to the `/var/log/auth-errors` file. Use the `authpriv` facility with the `alert` priority.

```
[root@serverb ~]# logger -p authpriv.alert "Logging test authpriv.alert"
```

- 4.5.** Verify that the `/var/log/auth-errors` file contains a log entry with the message that you previously specified.

```
[root@serverb ~]# tail /var/log/auth-errors
Jun  9 11:53:41 serverb student[4058]: Logging test authpriv.alert
```

5. Return to the workstation system as the student user.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
student@workstation:~$
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade logs-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish logs-review
```

## 5.12. Summary

---

In this lesson, you learned how to use various logging services to help with monitoring, auditing, and troubleshooting. Additionally, you learned to synchronize a server's clock to keep an accurate and consistent date and time with other systems. This synchronization is important for accurately recording the timeline of logged events.

The `systemd-journald` and `rsyslog` services collect, process, and store log messages that the system and applications generate. Most log files are stored in the `/var/log` directory; the `rsyslog` service can also write log messages to other locations, including databases, user accounts, and remote servers.

By default, the `systemd` journals are temporarily stored and do not persist across a reboot by default, although if you create a `/var/log/journal` directory and refresh the service, then they can be stored persistently. The system periodically rotates the `systemd-journald` system journal and the `rsyslog` log files to prevent them from consuming all storage on their file systems.

The `chronyd` service helps to synchronize a server's clock with different time sources. You can adjust the server's time zone based on its location or on any other location as appropriate.

## Chapter 6.

# Managing Security with SELinux

---

## Goal

Protect systems and manage security by using SELinux.

## Sections

- Operating SELinux (and Guided Exercise)
- Controlling SELinux File Contexts (and Guided Exercise)
- Tuning the SELinux Policy by Adjusting Booleans (and Guided Exercise)
- Investigating and Resolving SELinux Issues (and Guided Exercise)

## Lab

- Manage Security with SELinux

## 6.1. Operating SELinux

### Objectives

- Describe how SELinux works, and control the default and the current SELinux mode.

### SELinux Architecture

Security Enhanced Linux (SELinux) is a critical security feature of Linux. Access to files, ports, and other resources is controlled at a granular level. Processes are permitted to access only the resources that their SELinux policy or Boolean settings specify.

File permissions control file access for a specific user or group. However, file permissions do not prevent an authorized user with file access from using a file for an unintended purpose.

For example, with write access to a file, other editors or programs can still open and modify a structured data file that is designed for only a specific program to write to. These operations could result in file corruption or a data security issue. File permissions do not stop such undesired access, because they do not control *how* a file is used but only *who* is allowed to read, write, or run a file.

SELinux consists of application-specific policies that the application's developers define to declare which actions and accesses are allowed for each binary executable, configuration file, and data file that the application uses. This policy is known as a *targeted policy*, because one policy defines an application's activities. Policies declare the predefined labels that are configured on individual programs, files, and network ports.

### SELinux Usage

SELinux enforces a set of access rules that explicitly define allowed actions between processes and resources. Any action that is not defined in an access rule is not allowed. Because only defined actions are allowed, applications with a poor security design are still protected from malicious use. Applications or services with a targeted policy run in a *confined* domain, whereas an application without a policy runs *unconfined* and without any SELinux protection. Individual targeted policies can be disabled to assist with application and security policy development and debugging.

SELinux has the following operational modes:

- **Enforcing:** SELinux enforces the security policies on an entire system. Enforcing mode is the default and recommended mode in Red Hat Enterprise Linux.

- **Permissive:** SELinux loads the policies and is active, but instead of enforcing access control rules, it logs access violations. This mode is helpful for testing and troubleshooting applications and rules.
- **Disabled:** SELinux is turned off and SELinux violations are not implemented or logged. This mode also avoids labeling any files or directories with security contexts, which makes it difficult to enable SELinux in the future. Disabling SELinux is strongly discouraged.

#### Important

RHEL no longer supports setting the SELINUX=disabled option in the /etc/selinux/config file.

## SELinux Concepts

The primary goal of SELinux is to protect user data from improper use by compromised applications or system services. Most Linux administrators are familiar with the standard user, group, and world file permission security model, which is known as *Discretionary Access Control (DAC)* because administrators set file permissions as they need. SELinux provides an additional layer of object-based security, which is defined in granular rules. This layer of security is known as *Mandatory Access Control (MAC)*. MAC policies apply to all users and cannot be bypassed for specific users by discretionary configuration settings.

For example, a web server's open firewall port allows remote anonymous access to a web client. However, a malicious user who accesses that port might try to compromise a system through an existing vulnerability. If a vulnerability compromises the permissions for the apache user and group, then a malicious user might directly access the /var/www/html document root content, or the system's /tmp and /var/tmp directories, or other accessible files and directories.

SELinux policies are security rules that define how specific processes access relevant files, directories, and ports. Every resource entity, such as a file, process, directory, or port, has an *SELinux context* label. The context label matches a defined SELinux policy rule to allow a process to access the labeled resource. By default, an SELinux policy does not allow any access unless an explicit rule grants access. When no allow rule is defined, all access is disallowed.

SELinux labels have user, role, type, and security level fields. Targeted policy, which is enabled in RHEL by default, defines rules by using the type context. Type context names typically end with \_t.

SELinux User	/	Role	/	Type	/	Level	/	File
unconfined_u:object_r:httppd_sys_content_t:s0	/var/www/html/file2							

Figure 3. SELinux file context

## Policy Access Rule Concepts

For example, a web server process is labeled with the `httpd_t` type context. Web server files and directories in the `/var/www/html/` directory and other locations are labeled with the `httpd_sys_content_t` type context. Temporary files in the `/tmp` and `/var/tmp` directories have the `tmp_t` type contexts as a label. The web server's ports have the `http_port_t` type context as a label.

An Apache web server process runs with the `httpd_t` type context. A policy rule permits the Apache server to access files and directories that are labeled with the `httpd_sys_content_t` type context. By default, files in the `/var/www/html` directory have the `httpd_sys_content_t` type context. A web server policy has by default no allow rules for using files that are labeled `tmp_t`, such as in the `/tmp` and `/var/tmp` directories, thus disallowing access. With SELinux enabled, a malicious user who uses a compromised Apache process would still not have access to the `/tmp` directory files.

A MariaDB server process runs with the `mysqld_t` type context. By default, files in the `/data/mysql` directory have the `mysqld_db_t` type context. A MariaDB server can access the `mysqld_db_t` labeled files, but has no rules to allow access to files for other services, such as `httpd_sys_content_t` labeled files.

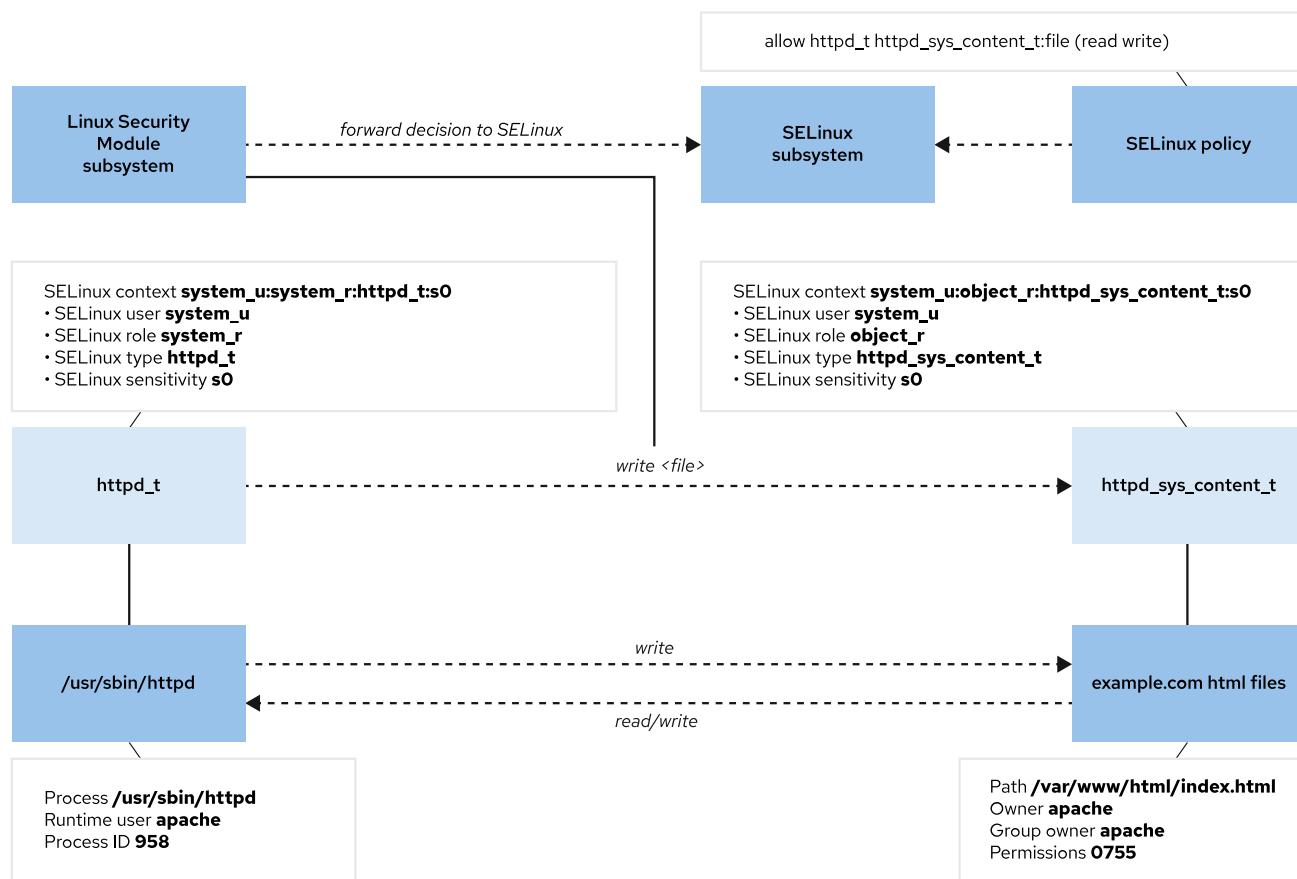


Figure 4. SELinux decision-making flow

Many commands that list resources use the `-Z` option to manage SELinux contexts. For example, the `ps`, `ls`, `cp`, and `mkdir` commands all use the `-Z` option.

In the following example, use the `ps` command `-Z` option to view the SELinux contexts of the processes:

```
root@host:~# ps axZ
  LABEL           PID TTY STAT TIME COMMAND
system_u:system_r:init_t:s0      1 ?    Ss   0:03 /usr/lib/systemd/systemd...
system_u:system_r:kernel_t:s0     2 ?    S    0:00 [kthreadd]
system_u:system_r:kernel_t:s0     3 ?    S    0:00 [pool_workqueue_release]
system_u:system_r:kernel_t:s0     4 ?    I<   0:00 [kworker/R-rcu_gp]
system_u:system_r:kernel_t:s0     5 ?    I<   0:00 [kworker/R-sync_wq]
...output omitted...
```

In the following example, use the `ps` command `-Z` option to view the SELinux context of the `httpd` service:

```
root@host:~# ps -ZC httpd
LABEL PID TTY TIME CMD
system_u:system_r:httpd_t:s0 6780 ?
system_u:system_r:httpd_t:s0 6781 ?
system_u:system_r:httpd_t:s0 6782 ?
system_u:system_r:httpd_t:s0 6783 ?
system_u:system_r:httpd_t:s0 6789 ?
```

In the following example, use the `ls` command `-Z` option to view the SELinux context of the `/var/www` directory:

```
root@host:~# ls -Z /var/www
system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
system_u:object_r:httpd_sys_content_t:s0 html
```

## Change the SELinux Mode

Use the `getenforce` command to view the current SELinux mode:

```
root@host:~# getenforce
Enforcing
```

Use the `setenforce` command to change the SELinux mode:

```
root@host:~# setenforce
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]
```

Use the `setenforce` command to change to the permissive mode:

```
root@host:~# setenforce 0
```

Use the `getenforce` command to view the current SELinux mode:

```
root@host:~# getenforce
Permissive
```

Alternatively, set the SELinux mode at boot time with a kernel parameter. Pass the `enforcing=0` kernel parameter to boot the system into the permissive mode, or pass the `enforcing=1` kernel parameter to boot into the enforcing mode. You can disable SELinux by passing the `selinux=0` kernel parameter, or pass `selinux=1` to enable SELinux.

Red Hat recommends rebooting the server when you change the SELinux mode from the permissive mode to the enforcing mode. A reboot ensures that the services that are started in the permissive mode are confined in the next boot.

## Set the Default SELinux Mode

To configure SELinux persistently, use the `/etc/selinux/config` file. In the following default example, the configuration sets SELinux to the enforcing mode. The comments list other valid values, such as the permissive and disabled modes.

```
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#       enforcing - SELinux security policy is enforced.  
#       permissive - SELinux prints warnings instead of enforcing.  
#       disabled - No SELinux policy is loaded.  
...output omitted...  
  
SELINUX=enforcing  
# SELINUXTYPE= can take one of these three values:  
#       targeted - Targeted processes are protected,  
#       minimum - Modification of targeted policy. Only selected processes are  
protected.  
#       mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

The system reads this file at boot time and starts SELinux accordingly. The `selinux=0|1` and `enforcing=0|1` kernel arguments override this configuration.

### References

`getenforce(8)`, `setenforce(8)`, and `selinux_config(5)` man pages

For more information about SELinux for Red Hat Enterprise Linux 10, refer to the *Getting Started with SELinux* chapter in the *Using SELinux* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_selinux/index#getting-started-with-selinux](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_selinux/index#getting-started-with-selinux)

For more information about SELinux states and modes for Red Hat Enterprise Linux 10, refer to the *Changing SELinux States and Modes* chapter in the *Using SELinux* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_selinux/index#changing-selinux-states-and-modes](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_selinux/index#changing-selinux-states-and-modes)



## 6.2. Guided Exercise

### Operate SELinux

Control the current SELinux mode and adjust the default SELinux mode that is used at boot time.

#### Outcomes

- View and set the current SELinux mode.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start selinux-opsmode
```

#### Instructions

1. Log in to the `servera` machine as the `student` user and then switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Change the default SELinux mode to the permissive mode.

- 2.1. Use the `getenforce` command to verify the current SELinux mode on the `servera` machine.

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

- 2.2. Edit the `/etc/selinux/config` configuration file to change the `SELINUX` parameter from `enforcing` to `permissive` mode.

```
...output omitted...
SELINUX=permissive
...output omitted...
```

- 2.3.** Use the **setenforce** command to change the SELinux parameter to the permissive mode.

```
[root@servera ~]# setenforce 0
```

- 2.4.** Use the **getenforce** command to verify the current SELinux mode.

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

- 3.** Change the default SELinux mode back to the enforcing mode in the configuration file.

- 3.1.** Edit the **/etc/selinux/config** configuration file to change the SELinux parameter from permissive to enforcing mode.

```
...output omitted...
SELINUX=enforcing
...output omitted...
```

- 3.2.** Use the **setenforce** command to set the current SELinux mode to the enforcing mode.

```
[root@servera ~]# setenforce 1
```

- 3.3.** Use the **getenforce** command to confirm that SELinux is set to the enforcing mode.

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

- 4.** Reboot the **servera** machine to implement the persistent configuration.

```
[root@servera ~]# systemctl reboot
...output omitted...
student@workstation:~$
```

- 5.** Verify the SELinux mode on the **servera** machine.

- 5.1.** Log in to the servera machine as the student user and then switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 5.2.** Use the getenforce command to verify the current SELinux mode.

```
[student@servera ~]$ getenforce
Enforcing
```

- 6.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish selinux-opsmode
```

## 6.3. Controlling SELinux File Contexts

### Objectives

- Control SELinux file contexts in the system policy to change the contexts on files based on the policy settings, and manually set nonstandard contexts.

### Initial SELinux Context

All resources, such as processes, files, and ports, are labeled with an SELinux *context*. SELinux maintains a file-based database of file labeling policies in the /etc/selinux/targeted-contexts/files directory. New files obtain a default label when their file name matches an existing labeling policy.

When a new file's name does not match an existing labeling policy, the file inherits the same label as the parent directory. With labeling inheritance, all files are always labeled when created, regardless of whether an explicit policy exists for a file.

When files are created in default locations that have an existing labeling policy, or when a policy exists for a custom location, new files are then labeled with a correct SELinux context. However, if a file is created in an unexpected location without an existing labeling policy, then the inherited label might not be correct for the new file's intended purpose.

Furthermore, copying a file to a new location can cause that file's SELinux context to change, where the new context is determined by the new location's labeling policy, or from parent directory inheritance if no policy exists. A file's SELinux context can be preserved during copying to retain the context label that was determined for the file's original location. For example, the `cp -p` command preserves all file attributes during copying where possible, and the `cp --preserve=context` command preserves only SELinux contexts.

The `ls -Z` command displays the SELinux context of a file or directory.

### Note

Copying a file always creates a file inode, and that inode's attributes, including the SELinux context, must be initially set, as previously discussed.

However, moving a file does not typically create an inode if the move occurs within the same file system, but instead moves the existing inode's file name to a new location. Because the existing inode's attributes do not need to be initialized, a file that is moved with the `mv` command preserves its SELinux context unless you set a new context on the file with the `-Z` option.

After you copy or move a file, verify that it has the appropriate SELinux context and set it correctly if necessary.

The following example demonstrates how this process works.

Create two files in the `/tmp` directory.

```
root@host:~# touch /tmp/file1 /tmp/file2
```

Both files receive the `user_tmp_t` context type due to context inheritance from the parent directory.

```
root@host:~# ls -Z /tmp/file*
unconfined_u:object_r:user_tmp_t:s0 /tmp/file1
unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
```

The `/var/www/html` directory has the `httpd_sys_content_t` context.

```
root@host:~# ls -Zd /var/www/html/
system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

Move one file from the `/tmp` directory to the `/var/www/html` directory.

```
root@host:~# mv /tmp/file1 /var/www/html/
```

Copy the other file to the same directory.

```
root@host:~# cp /tmp/file2 /var/www/html/
```

The moved file retained its original label and the copied file inherited the destination directory label. The SELinux user role is `unconfined_u`, the SELinux role is `object_r`, and the (lowest possible) sensitivity level is `s0`. Advanced SELinux configurations and features use these values.

```
root@host:~# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

## Change the SELinux Context

You can manage the SELinux context on files by using the `semanage fcontext`, `restorecon`, and `chcon` commands.

The recommended method to change the context for a file is to create a file context policy by using the `semanage fcontext` command, and then to apply the specified context in the policy to the file by using the `restorecon` command. This method ensures that you can relabel the file to its correct context with the `restorecon` command whenever necessary. The advantage of this method is that you do not need to remember what the context is supposed to be, and you can correct the context on a set of files.

The `chcon` command changes the SELinux context directly on files, without referencing the system's SELinux policy. The `chcon` command is useful for testing and debugging; however, changing contexts manually with this method is temporary. Although file contexts that you can change manually survive a reboot, they might be replaced if you run the `restorecon` command to relabel the contents of the file system.

### Important

When an SELinux system *relabel* occurs, all files on a system are labeled with their policy defaults. When you use the `restorecon` command on a file, any context that you change manually on the file is replaced if it does not match the rules in the SELinux policy.

The following example creates a directory with the `default_t` SELinux context, which it inherited from the `/` parent directory.

Create the `/virtual` directory

```
root@host:~# mkdir /virtual
```

View the file context of the /virtual directory.

```
root@host:~# ls -Zd /virtual
unconfined_u:object_r:default_t:s0 /virtual
```

The chcon command sets the file context of the /virtual directory to the httpd\_sys\_content\_t type.

```
root@host:~# chcon -t httpd_sys_content_t /virtual
```

View the updated file context of the /virtual directory.

```
root@host:~# ls -Zd /virtual
unconfined_u:object_r:httpd_sys_content_t:s0 /virtual
```

The restorecon command resets the context to the default value of default\_t. Note the Relabeled message.

```
root@host:~# restorecon -v /virtual
Relabeled /virtual from unconfined_u:object_r:httpd_sys_content_t:s0 to
unconfined_u:object_r:default_t:s0
```

View the updated file context of the /virtual directory.

```
root@host:~# ls -Zd /virtual
unconfined_u:object_r:default_t:s0 /virtual
```

## Define SELinux Default File Context Policies

The semanage fcontext command displays and modifies the policies that determine the default file contexts. You can list all the file context policy rules by running the semanage fcontext -l command. These rules use extended regular expression syntax to specify the path and file names.

When viewing policies, the most common extended regular expression is `(/.*)?`, which is usually appended to a directory name. This notation is humorously named *the pirate*, because it looks like a face with an eye patch and a hooked hand next to it.

This syntax is described as "a set of characters that begin with a slash and followed by any number of characters, where the set can either exist or not exist". The expression matches the directory itself, even when empty, and also matches almost any file name that is created within that directory.

For example, the following rule specifies that the /var/www/cgi-bin directory, and any files in it or in its subdirectories (and in their subdirectories, and so on), have the system\_u:object\_r:httpd\_sys\_script\_exec\_t:s0 SELinux context, unless a more specific rule overrides this one.

```
/var/www/cgi-bin(/.*)? all files system_u:object_r:httpd_sys_script_exec_t:s0
```

### Note

The all files field option from the previous example is the default file type that the semanage command uses when you do not specify one.

This option applies to all file types that you can use with the semanage command; they are the same as the standard file types as in the *Controlling Access to Files* (RH0020L) lesson.

You can get more information from the `semanage-fcontext(8)` man page.

## Basic File Context Operations

The following table is a reference for the `semanage fcontext` command options to add, remove, or list SELinux file context policies:

**Table 12. The `semanage fcontext` Command**

Option	Description
<code>-a, --add</code>	Add a record of the specified object type.
<code>-d, --delete</code>	Delete a record of the specified object type.
<code>-l, --list</code>	List records of the specified object type.

To manage SELinux contexts, install the `policycoreutils` and `policycoreutils-python-utils` packages, which contain the `restorecon` and `semanage` commands.

To reset all files in a directory to the default policy context, first use the `semanage fcontext -l` command to locate and verify that the correct policy exists with the intended file context. Then, use the `restorecon` command on the wildcarded directory name to reset all the files recursively.

In the following example, view the file contexts before and after using the `semanage` and `restorecon` commands.

First, verify the SELinux context for the files:

```
root@host:~# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

Then, use the `semanage fcontext -l` command to list the default SELinux file contexts:

```
root@host:~# semanage fcontext -l
...output omitted...
/var/www(/.*)?      all files      system_u:object_r:httpd_sys_content_t:s0
...output omitted...
```

The `semanage` command output indicates that all the files and subdirectories in the `/var/www` directory have the `httpd_sys_content_t` context by default. Running `restorecon` command on the wildcarded directory restores the default context on all files and subdirectories.

```
root@host:~# restorecon -Rv /var/www/
Relabeled /var/www/html/file1 from unconfined_u:object_r:user_tmp_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

View the file context of the `/var/www/html/file*` files.

```
root@host:~# ls -Z /var/www/html/file*
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The following example uses the `semanage` command to add a context policy for a new directory.

First, create the `/virtual` directory.

```
root@host:~# mkdir /virtual
```

Create the /virtual/index.html file.

```
root@host:~# touch /virtual/index.html
```

View the SELinux context for the /virtual directory.

```
root@host:~# ls -Zd /virtual/
unconfined_u:object_r:default_t:s0 /virtual
```

View the SELinux context for the /virtual/index.html file.

```
root@host:~# ls -Z /virtual/
unconfined_u:object_r:default_t:s0 index.html
```

Use the semanage fcontext command to add an SELinux file context policy for the directory.

```
root@host:~# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.*)?'
```

Use the restorecon command on the wildcarded directory to set the default context on the directory and all files within it.

```
root@host:~# restorecon -RFvv /virtual
Relabeled /virtual from unconfined_u:object_r:default_t:s0 to
system_u:object_r:httpd_sys_content_t:s0
Relabeled /virtual/index.html from unconfined_u:object_r:default_t:s0 to
system_u:object_r:httpd_sys_content_t:s0
```

View the SELinux context for the /virtual directory.

```
root@host:~# ls -Zd /virtual/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/
```

View the SELinux context for the /virtual/index.html file.

```
root@host:~# ls -Z /virtual/
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 index.html
```

Use the `semanage fcontext` command with the `-C` option to view any local customizations to the default policy.

```
root@host:~# semanage fcontext -l -C
SELinux fcontext      type          Context
/virtual(/.*)?        all files    system_u:object_r:httpd_sys_content_t:s0
```

## References

`chcon(1)`, `restorecon(8)`, `semanage(8)`, and `semanage-fcontext(8)` man pages

## 6.4. Guided Exercise

### Control SELinux File Contexts

View and set SELinux file contexts on files based on the policy settings.

#### Outcomes

- Configure the Apache HTTP server to publish web content from a nonstandard document root.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start selinux-filecontexts
```

#### Instructions

- Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Configure Apache HTTP server to use a document directory in a nonstandard location.

- 2.1.** Create the /custom directory.

```
[root@servera ~]# mkdir /custom
```

- 2.2.** Create the /custom/index.html file that contains the This is SERVERA. text.

```
[root@servera ~]# echo 'This is SERVERA.' > /custom/index.html
```

- 2.3.** Configure Apache to use the new directory location. Edit the Apache /etc/httpd/conf/httpd.conf configuration file, and replace the two occurrences of the /var/www/html directory with the /custom directory.

```
...output omitted...
DocumentRoot "/custom"
...output omitted...
<Directory "/custom">
...output omitted...
```

- 3.** Start and enable the Apache web service, and then confirm that the service is running.

**3.1.** Start and enable the Apache web service by using the `systemctl` command.

```
[root@servera ~]# systemctl enable --now httpd
Created symlink '/etc/systemd/system/multi-user.target.wants/httpd.service' →
'/usr/lib/systemd/system/httpd.service'.
```

**3.2.** Verify that the service is running.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset:
disabled)
  Active: active (running) since Fri 2025-06-06 13:42:54 UTC; 55s ago
  Invocation: 68210ceb6dd649d3b87f633c1a980775
    Docs: man:httpd.service(8)
   Main PID: 2104 (httpd)
...output omitted...
Jun 06 13:42:54 servera systemd[1]: Starting httpd.service - The Apache HTTP
Server...
Jun 06 13:42:54 servera (httpd)[2104]: httpd.service: Referenced but unset ...
Jun 06 13:42:54 servera httpd[2104]: Server configured, listening on: port 80
Jun 06 13:42:54 servera systemd[1]: Started httpd.service - The Apache HTTP
Server.
```

- 4.** Open a new terminal on the workstation machine, and use the `curl` command to try to view the `http://servera/index.html` web page. You get an error message that you do not have permission to access the file.

```
student@workstation:~$ curl http://servera/index.html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

5. On the servera machine, inspect the root cause for the issue and set the httpd\_sys\_content\_t SELinux context for the /custom/index.html file.

- 5.1. Compare the SELinux contexts of the /custom/ and /var/www/html directories.

```
[root@servera ~]# ls -ldZ /custom /var/www/html
drwxr-xr-x. 2 root root unconfined_u:object_r:default_t:s0    24 Jul  3 19:46
/custome
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0  6 Jan 27 00:00
/var/www/html
```

- 5.2. Define an SELinux file context rule that sets the httpd\_sys\_content\_t context for the /custom directory and all the files in it.

```
[root@servera ~]# semanage fcontext -a -t httpd_sys_content_t '/custom(/.*)?'
```

- 5.3. Apply the SELinux file contexts for the /custom directory.

```
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

6. Try to view http://servera/index.html again on the workstation machine with the curl command. You should see the This is SERVERA. message.

```
student@workstation:~$ curl http://servera/index.html
This is SERVERA.
```

7. Return to the workstation machine as the student user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish selinux-filecontexts
```

## 6.5. Tuning the SELinux Policy by Adjusting Booleans

### Objectives

- Tune the system by changing the behavior of the SELinux system policy by activating and deactivating SELinux Booleans.

### SELinux Booleans

An application or service developer writes an SELinux targeted policy to define the allowed behavior of the targeted application. A developer can include optional application behavior in the SELinux policy that can be enabled when the behavior is allowed on a specific system. SELinux Booleans enable or disable the SELinux policy's optional behavior. With Booleans, you can selectively tune the behavior of an application.

These optional behaviors are application-specific, and must be discovered and selected for each targeted application. Service-specific Booleans are documented in that service's SELinux man page.

For example, the web server httpd service has its `httpd(8)` man page, and an `httpd_selinux(8)` man page to document its SELinux policy, including the supported process types, file contexts, and the available Boolean-enabled behaviors. The SELinux man pages are provided by the `selinux-policy-doc` package.

Use the `getsebool` command to list available Booleans for the targeted policies on this system, and the current Boolean status. Use the `setsebool` command to enable or disable the running state of these behaviors. The `setsebool -P` command makes the setting persistent by writing to the policy file. Only privileged users can set SELinux Booleans.

In the following example, use the `getsebool` command to list all the available Booleans along with the status:

```
root@host:~# getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
...output omitted...
```

### Example Httpd Policy Boolean

The httpd service policy includes the `httpd_enable_homedirs` Boolean, which enables the sharing of home directories with the httpd service. Typically, a user's local home directory is accessible to the user only when logged in to the local system. Alternatively, home directories are shared and accessed by using a remote file sharing protocol, such as NFS. In both scenarios, home directories are not shared by using the httpd service, by default, and are not available to the user through a browser.

In the following example, list the status of the `httpd_enable_homedirs` Boolean:

```
root@host:~# getsebool httpd_enable_homedirs  
httpd_enable_homedirs --> off
```

You can enable sharing and enable users to access their home directories with a browser. When enabled, the httpd service shares home directories that are labeled with the `user_home_dir_t` file context. Users can then access and manage their home directory files from a browser.

## Manage the Policy Boolean

Setting SELinux Booleans with the `setsebool` command without the `-P` option is temporary, and settings return to the persistent values after rebooting. View additional information with the `semanage boolean -l` command, which lists the Booleans from the policy files, including whether a Boolean is persistent, the default and current values, and a short description.

In the following example, list the `httpd_enable_homedirs` Boolean:

```
root@host:~# semanage boolean -l | grep httpd_enable_homedirs  
httpd_enable_homedirs          (off    ,   off)  Allow httpd to enable homedirs
```

In the following example, enable the `httpd_enable_homedirs` Boolean:

```
root@host:~# setsebool httpd_enable_homedirs on
```

In the following example, list the `httpd_enable_homedirs` Boolean:

```
root@host:~# semanage boolean -l | grep httpd_enable_homedirs  
httpd_enable_homedirs          (on    ,   off)  Allow httpd to enable homedirs
```

In the following example, verify that the `httpd_enable_homedirs` Boolean is enabled:

```
root@host:~# getsebool httpd_enable_homedirs  
httpd_enable_homedirs --> on
```

The previous example temporarily set the current value for the `httpd_enable_homedirs` Boolean to `on`, until the system reboots.

To change the default setting, use the `setsebool -P` command to make the setting persistent:

```
root@host:~# setsebool -P httpd_enable_homedirs on
```

View the Boolean's information from the policy file:

```
root@host:~# semanage boolean -l | grep httpd_enable_homedirs  
httpd_enable_homedirs          (on    ,   on)  Allow httpd to enable homedirs
```

To list only persistent Booleans with customized settings compared to the original policy, use the `semanage boolean -l -C` command.

```
root@host:~# semanage boolean -l -C  
SELinux boolean           State  Default Description  
  
httpd_enable_homedirs      (on    ,   on)  Allow httpd to enable homedirs
```

## References

`booleans(8)`, `getsebool(8)`, `setsebool(8)`, `semanage(8)`, and `semanage-boolean(8)` man pages

## 6.6. Guided Exercise

### Tune the SELinux Policy by Adjusting Booleans

Change the behavior of the SELinux system policy by activating and deactivating SELinux Booleans.

#### Outcomes

- Configure the Apache web service to publish web content from the user's home directory.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start selinux-booleans
```

#### Instructions

- Log in to the `servera` machine as the `student` user and then switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Edit the `/etc/httpd/conf.d/userdir.conf` configuration file to enable the Apache feature so that users can publish web content from their home directory. Comment out the line in the `IfModule` section that sets the `UserDir` variable to the `disabled` value, and uncomment the line that sets the `UserDir` variable to the `public_html` value.

```
...output omitted...
<IfModule mod_userdir.c>
...output omitted...
# UserDir disabled

...output omitted...
UserDir public_html

...output omitted...
</IfModule>
```

**3.** Start and enable the Apache web service.

```
[root@servera ~]# systemctl enable --now httpd
Created symlink '/etc/systemd/system/multi-user.target.wants/httpd.service' →
'/usr/lib/systemd/system/httpd.service'.
```

**4.** Return as the student user on the servera machine.

Create the `~/public_html/index.html` file with the This is student content on SERVERA. content.

**4.1.** Return as the student user on the servera machine.

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

**4.2.** Create the `~/public_html` directory.

```
[student@servera ~]$ mkdir ~/public_html
```

**4.3.** Create the `~/public_html/index.html` file with the following content:

```
[student@servera ~]$ echo 'This is student content on SERVERA.' > \
~/public_html/index.html
```

**4.4.** For the Apache web service to serve the contents of the `/home/student/public_html` directory, it must be allowed to share files and subdirectories in the `/home/student` directory. When you created the `/home/student/public_html` directory, it was

automatically configured to allow anyone with home directory permission to access its contents.

Change the /home/student directory permissions to allow the Apache web service to access the public\_html subdirectory.

```
[student@servera ~]$ chmod 711 /home/student
```

**4.5.** Verify the /home/student directory permissions.

```
[student@servera ~]$ ls -ld /home/student
drwx--x--x. 16 student student 4096 Nov  3 09:28 /home/student
```

- 5.** On the workstation machine, open the Firefox web browser and enter the `http://servera/~student/index.html` address.

An error message states that you do not have permission to access the file.

- 6.** Log in as the root user on the servera machine. Use student as the password.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 7.** Use the `getsebool` command to determine whether any Booleans restrict access to home directories for the `httpd` service.

```
[root@servera ~]# getsebool -a | grep home
...output omitted...
httpd_enable_homedirs --> off
...output omitted...
```

- 8.** Use the `setsebool` command to enable persistent access to the home directory for the `httpd` service.

```
[root@servera ~]# setsebool -P httpd_enable_homedirs on
```

- 9.** On the workstation machine, open the Firefox web browser and enter the `http://servera/~student/index.html` address.

Verify that you can now see the This is student content on SERVERA. message. You might need to close and reopen your web browser to see the message.

**10.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish selinux-booleans
```

## 6.7. Investigating and Resolving SELinux Issues

### Objectives

- Investigate issues that potentially relate to SELinux by using SELinux log analysis tools and interpreting their output.

### Troubleshoot SELinux Issues

When applications unexpectedly fail to work due to SELinux access denials, methods and tools are available to resolve these issues. It is helpful to start by understanding some fundamental concepts and behaviors when SELinux is enabled.

- SELinux consists of targeted policies that explicitly define allowable actions.
- A policy entry defines a labeled process and a labeled resource that interact.
- The policy states the process type, and the file or port context, by using labels.
- The policy entry defines one process type, one resource label, and the explicit action to allow.
- An action can be a system call, a kernel function, or another specific programming routine.
- If no entry is created for a specific process-resource-action relationship, then the action is denied.
- When an action is denied, the attempt is logged with context information.

Red Hat Enterprise Linux provides a stable targeted SELinux policy for almost every service in the distribution. Therefore, it is unusual to have SELinux access problems with common RHEL services when they are configured correctly. SELinux access problems occur when services are implemented incorrectly, or when new applications have incomplete policies. Consider these troubleshooting concepts before making broad SELinux configuration changes.

- Most access denials indicate that SELinux is working correctly by blocking improper actions.
- Evaluating denied actions requires some familiarity with normal, expected service actions.
- The most common SELinux issue is an incorrect context on new, copied, or moved files.
- File contexts can be fixed when an existing policy references their location.
- Optional Boolean policy features are documented in the `_selinux` man pages.
- Implementing Boolean features generally requires setting additional non-SELinux configuration.
- SELinux policies do not replace or circumvent file permissions or access control list restrictions.

When a common application or service fails, and the service is known to have a working SELinux policy, first see the service's `_selinux` man page to verify the correct context type label. View the affected process and file attributes to verify that the correct labels are set.

## Monitor SELinux Violations

The SELinux troubleshooting service, from the `setroubleshoot-server` package, provides tools to diagnose SELinux issues. When SELinux denies an action, an Access Vector Cache (AVC) message is logged to the `/var/log/audit/audit.log` security log file. The SELinux troubleshooting service monitors for AVC events and sends an event summary to the `/var/log/messages` file.

The AVC summary includes an event unique identifier (UUID). Use the `sealert -l UUID` command to view comprehensive report details for the specific event. Use the `sealert -a /var/log/audit/audit.log` command to view all existing events.

Consider the following example sequence of commands on a standard Apache web server.

Create the `/root/mypage` file.

```
root@host:~# touch /root/mypage
```

Move the `/root/mypage` file to the `/var/www/html` directory.

```
root@host:~# mv /root/mypage /var/www/html
```

Start the `httpd` service.

```
root@host:~# systemctl start httpd
```

Try to access the Apache server at the `http://localhost/mypage` URL.

```
root@host:~# curl http://localhost/mypage
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

The web server does not display the content, and returns a permission denied error. An AVC event is logged to the /var/log/audit/audit.log and /var/log/messages files. Note the suggested sealert command and UUID in the /var/log/messages event message.

View the log messages from the /var/log/audit/audit.log file.

```
root@host:~# tail /var/log/audit/audit.log
...output omitted...
type=AVC msg=audit(1749508506.271:159): avc: denied { getattr } for pid=1984
comm="httpd" path="/var/www/html/mypage" dev="sda3" ino=8805547
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
...output omitted...
```

View the log messages from the /var/log/messages file.

```
root@host:~# tail /var/log/messages
...output omitted...
Jun  9 22:35:06 servera setroubleshoot[2175]: SELinux is preventing
/usr/sbin/httpd from getattr access on the file /var/www/html/mypage. For
complete SELinux messages run: sealert -l a0bca4b2-46a9-4252-b0cd-6de0dfcdd454
...output omitted...
```

The sealert command output describes the event, and includes the affected process, the accessed file, and the attempted and denied action. The output includes advice for correcting the file's label, if appropriate. Additional advice describes how to generate a new policy to allow the denied action. Use the given advice only when it is appropriate for your scenario.

 **Important**

The `sealert` command output includes a confidence rating, which indicates the level of confidence that the given advice mitigates the denial. However, that advice might not be appropriate for your scenario.

For example, if the AVC denial is caused by a file that is placed in the wrong location, then the `sealert` command might advise either to adjust the file's context label or to create a policy for this location and action. Although technically accurate, it is not the correct solution for your scenario. If the root cause is a wrong location or file name, then moving or renaming the file and then restoring a correct file context is the correct solution instead.

```
root@host:~# sealert -l a0bca4b2-46a9-4252-b0cd-6de0dfcdd454
SELinux is preventing /usr/sbin/httpd from getattr access on the file
/var/www/html/mypage.
```

```
***** Plugin restorecon (99.5 confidence) suggests *****
```

If you want to fix the label.

```
/var/www/html/mypage default label should be httpd_sys_content_t.
```

Then you can run restorecon. The access attempt may have been stopped due to insufficient permissions to access a parent directory in which case try to change the following command accordingly.

Do

```
# /sbin/restorecon -v /var/www/html/mypage
```

```
***** Plugin catchall (1.49 confidence) suggests *****
```

If you believe that httpd should be allowed getattr access on the mypage file by default.

Then you should report this as a bug.

You can generate a local policy module to allow this access.

Do

allow this access for now by executing:

```
# ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp
```

Additional Information:

Source Context	system_u:system_r:httpd_t:s0
Target Context	unconfined_u:object_r:admin_home_t:s0
Target Objects	/var/www/html/mypage [ file ]
Source	httpd
Source Path	/usr/sbin/httpd
...output omitted...	

Raw Audit Messages

```
type=AVC msg=audit(1749508506.271:159): avc: denied { getattr } for pid=1984
comm="httpd" path="/var/www/html/mypage" dev="sda3" ino=8805547
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0

type=SYSCALL msg=audit(1749508506.271:159): arch=x86_64 syscall=newfstatat
success=no exit=EACCES a0=fffffff9c a1=7fd554011a70 a2=7fd562de19f0 a3=100
items=0 ppid=1982 pid=1984 auid=4294967295 uid=48 gid=48 euid=48 suid=48
fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm=httpd
```

```
For use by Wong Chia Cheong jason.wong76.jason.wong@trainoate.com Copyright © 2025 Red Hat, Inc.  
exe=/usr/sbin/httpd subj=system_u:system_r:httpd_t:s0 key=(null)  
Hash: httpd,httpd_t,admin_home_t,file,getattr
```

In this example, the accessed file is in the correct location, but does not have the correct SELinux file context. The Raw Audit Messages section displays information from the /var/log/audit/audit.log event entry. Use the restorecon /var/www/html/mypage command to set the correct context label. To correct multiple files recursively, use the restorecon -R command on the parent directory.

Use the ausearch command to search for AVC events in the /var/log/audit/audit.log log file. Use the -m option to specify the AVC message type and the -ts option to provide a time hint, such as recent.

```
root@host:~# ausearch -m AVC -ts recent  
----  
time->Mon Jun  9 22:35:06 2025  
type=PROCTITLE msg=audit(1749508506.271:159):  
proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44  
type=SYSCALL msg=audit(1749508506.271:159): arch=c000003e syscall=262  
success=no exit=-13 a0=fffffff9c a1=7fd554011a70 a2=7fd562de19f0 a3=100 items=0  
ppid=1982 pid=1984 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48  
egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd"  
exe="/usr/sbin/httpd" subj=system_u:system_r:httpd_t:s0 key=(null)  
type=AVC msg=audit(1749508506.271:159): avc: denied { getattr } for pid=1984  
comm="httpd" path="/var/www/html/mypage" dev="sda3" ino=8805547  
scontext=system_u:system_r:httpd_t:s0  
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
```

## Troubleshoot SELinux Issues with the Web Console

The RHEL web console includes tools for troubleshooting SELinux issues. Select **SELinux** from the left menu. The SELinux policy window displays the current enforcing state. The **SELinux access control errors** section lists current SELinux issues.

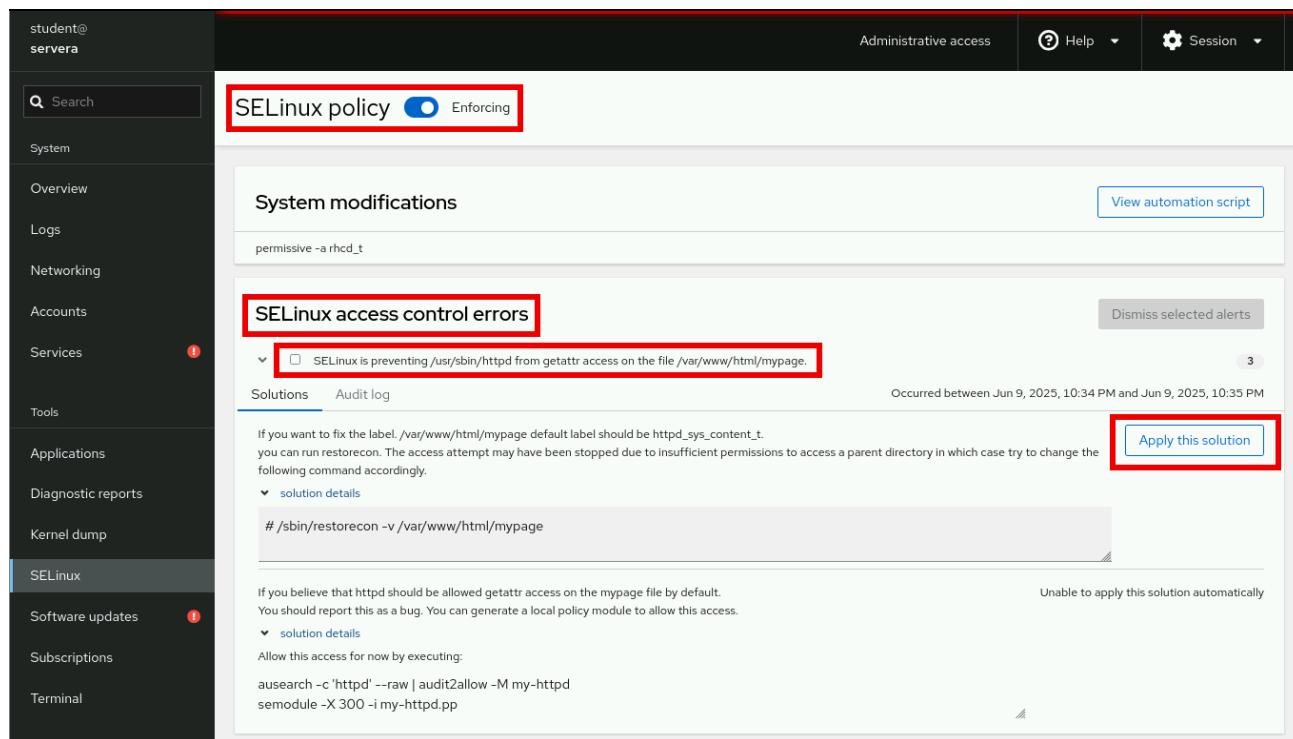


Figure 5. SELinux policy and errors in the web console

Click the **>** character to display event details. Click **Solution details** to display all event details and advice. You can click **Apply this solution**.

After correcting the issue, the **SELinux access control errors** section should remove that event from view. If the **No SELinux alerts** message appears, then you have corrected all current SELinux issues.

### References

[sealert\(8\) man page](#)

## 6.8. Guided Exercise

### Investigate and Resolve SELinux Issues

Troubleshoot SELinux issues by using SELinux log analysis tools and interpreting their output.

#### Outcomes

- Gain experience with SELinux troubleshooting tools.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start selinux-issues
```

#### Instructions

- From a web browser on the workstation machine, open the `http://servera/index.html` web page.  
An error message states that you do not have permission to access the file.
- Log in to the `servera` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Use the `less` command to view the contents of the `/var/log/messages` file. You use the `/` character and search for the `sealert` text. Press the `n` key until you reach the last occurrence, because previous exercises might also have generated SELinux messages. Copy the suggested `sealert` command so that you can use it in the next step. Use the `q` key to quit the `less` command.

```
[root@servera ~]# less /var/log/messages
...output omitted...
Jun  6 18:11:23 servera setroubleshoot[14796]: SELinux is preventing
/usr/sbin/httpd from getattr access on the file /custom/index.html. For
complete SELinux messages run: sealert -l 5a96d815-a187-4024-be5d-
36daf8e0d774
...output omitted...
```

4. Run the suggested sealert command. Note the source context, the target objects, the policy, and the enforcing mode. Find the correct SELinux context label for the file that the httpd service tries to serve.
  - 4.1. Run the sealert command. The output explains that the /custom/index.html file has an incorrect context label.

```
[root@servera ~]# sealert -l 5a96d815-a187-4024-be5d-36daf8e0d774
SELinux is preventing /usr/sbin/httpd from getattr access on the file
/custom/index.html.

***** Plugin catchall_labels (83.8 confidence) suggests *****

If you want to allow httpd to have getattr access on the index.html file
Then you need to change the label on /custom/index.html
Do
# semanage fcontext -a -t FILE_TYPE '/custom/index.html'
where FILE_TYPE is one of the following: ... abrt_dump_oops_exec_t, abrt_etc_t,
abrt_exec_t, abrt_handle_event_exec_t, abrt_helper_exec_t,
abrt_retrace_coredump_exec_t, abrt_retrace_spool_t, abrt_retrace_worker_exec_t,
abrt_tmp_t, abrt_upload_watch_tmp_t, abrt_var_cache_t, abrt_var_log_t,
abrt_var_run_t, accountsd_exec_t, acct_data_t, acct_exec_t, admin_crontab_tmp_t,
admin_passwd_exec_t, afs_logfile_t, aide_exec_t, aide_log_t, alsa_exec_t, ...
Then execute:
restorecon -v '/custom/index.html'

Additional Information:
Source Context           system_u:system_r:httpd_t:s0
Target Context          unconfined_u:object_r:default_t:s0
Target Objects          /custom/index.html [ file ]
Source                 httpd
Source Path             /usr/sbin/httpd
Port                   <Unknown>
Host                   servera
Source RPM Packages    httpd-core-2.4.63-1.el10.x86_64
Target RPM Packages    selinux-policy-targeted-40.13.26-1.el10.noarch
SELinux Policy RPM     selinux-policy-targeted-40.13.26-1.el10.noarch
Local Policy RPM       True
Selinux Enabled         targeted
Policy Type            Enforcing
Enforcing Mode         servera
Host Name              Linux servera 6.12.0-55.12.1.el10_0.x86_64 #1 SMP
Platform               PREEMPT_DYNAMIC Tue May 13 02:51:44 EDT 2025
                        x86_64

Alert Count             6
First Seen              2025-06-06 13:49:29 UTC
Last Seen               2025-06-06 18:11:19 UTC
Local ID                5a96d815-a187-4024-be5d-36daf8e0d774

Raw Audit Messages
type=AVC msg=audit(1749233479.381:1477): avc: denied { getattr } for pid=2114
comm="httpd" path="/custom/index.html" dev="sda3" ino=17265301
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

...output omitted...

- 4.2.** Verify the SELinux context for the /var/www/html directory, from where the httpd service serves the content by default. The httpd\_sys\_content\_t SELinux context is appropriate for the /custom/index.html file.

```
[root@servera ~]# ls -ldZ /var/www/html
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0 6 Jan 27 00:00
/var/www/html
```

- 5.** The Raw Audit Messages section of the sealert command contains information from the /var/log/audit/audit.log file. Use the ausearch command to search that file. The -m option searches based on the message type. The -ts option sets the starting date and time for the search. The following entry identifies the relevant process and file that cause the alert. The process is the httpd Apache web server, the file is /custom/index.html, and the context is system\_r:httpd\_t.

```
[root@servera ~]# ausearch -m AVC -ts today
...output omitted...
---
time->Fri Jun  6 18:11:19 2025
type=PROCTITLE msg=audit(1749233479.381:1477):
proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1749233479.381:1477): arch=c000003e syscall=262
success=no exit=-13 a0=fffffff9c a1=7f4aec013a98 a2=7f4afa3e09f0 a3=100
items=0 ppid=2104 pid=2114 auid=4294967295 uid=48 gid=48 euid=48 suid=48
fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd"
exe="/usr/sbin/httpd" subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1749233479.381:1477): avc:  denied { setattr } for
pid=2114 comm="httpd" path="/custom/index.html" dev="sda3" ino=17265301
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

- 6.** Resolve the issue by applying the httpd\_sys\_content\_t context.

- 6.1.** Reconfigure the SELinux policy to allow the httpd service to access the contents in the /custom/ directory.

```
[root@servera ~]# semanage fcontext -a -t httpd_sys_content_t '/custom(/.*)?'
```

- 6.2.** Use restorecon to apply the right SELinux file contexts for the /custom directory, by following the modified policy.

```
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

- 7.** On the workstation machine, use a web browser to view the `http://servera/index.html` URL.

You must see the `This is SERVERA.` message.

- 8.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish selinux-issues
```

## 6.9. Lab

# Manage Security with SELinux

Set the SELinux mode for the system.

Update the SELinux file contexts on files based on policy settings.

Adjust the SELinux system policy by activating and deactivating the SELinux Booleans.

## Outcomes

- Identify issues in system log files.
- Adjust the SELinux configuration.

## Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start selinux-review
```

## Instructions

1. Log in to the serverb machine as the student user and switch to the root user. Use student as the password.
2. From a web browser on the workstation machine, view the `http://serverb/lab.html` web page. You see the error message: You do not have permission to access this resource.
3. Research and identify the SELinux issue that prevents the Apache service from serving web content.
4. Display the SELinux context of the new HTTP document directory and the original HTTP document directory. Resolve the SELinux issue that prevents the Apache server from serving web content.
5. Verify that the Apache server can now serve web content on the workstation machine.
6. Return to the workstation machine as the student user.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade selinux-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish selinux-review
```

## Solution

---

Set the SELinux mode for the system.

Update the SELinux file contexts on files based on policy settings.

Adjust the SELinux system policy by activating and deactivating the SELinux Booleans.

## Outcomes

- Identify issues in system log files.
- Adjust the SELinux configuration.

## Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start selinux-review
```

## Instructions

1. Log in to the serverb machine as the student user and switch to the root user. Use `student` as the password.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

2. From a web browser on the workstation machine, view the `http://serverb/lab.html` web page. You see the error message: You do not have permission to access this resource.
3. Research and identify the SELinux issue that prevents the Apache service from serving web content.
  - 3.1. View the contents of the `/var/log/messages` file. Use the `/` key and search for the `sealert` string. Use the `q` key to quit the less command.

```
[root@serverb ~]# less /var/log/messages
...output omitted...
Jun 12 23:20:07 serverb setroubleshoot[2089]: failed to retrieve rpm info for
path '/lab-content/lab.html':
Jun 12 23:30:32 serverb setroubleshoot[2282]: SELinux is preventing
/usr/sbin/httpd from getattr access on the file /lab-content/lab.html. For
complete SELinux messages run: sealert -l 3ba31d72-a61a-4162-8064-a961581b9b6a
Jun 12 23:30:32 serverb setroubleshoot[2282]: SELinux is preventing
/usr/sbin/httpd from getattr access on the file /lab-content/lab.html.
...output omitted...
```

- 3.2.** Run the suggested `sealert` command. Note the source context, the target objects, the policy, and the enforcing mode.

```
[root@serverb ~]# sealert -l 3ba31d72-a61a-4162-8064-a961581b9b6a
SELinux is preventing /usr/sbin/httpd from getattr access on the file /lab-
content/lab.html.

***** Plugin catchall_labels (83.8 confidence) suggests *****

If you want to allow httpd to have getattr access on the lab.html file
Then you need to change the label on /lab-content/lab.html
Do
# semanage fcontext -a -t FILE_TYPE '/lab-content/lab.html'
where FILE_TYPE is one of the following:
...output omitted...

Additional Information:
Source Context          system_u:system_r:httpd_t:s0
Target Context          unconfined_u:object_r:default_t:s0
Target Objects          /lab-content/lab.html [ file ]
Source                 httpd
Source Path             /usr/sbin/httpd
...output omitted...
Local ID                3ba31d72-a61a-4162-8064-a961581b9b6a

Raw Audit Messages
type=AVC msg=audit(1749771030.133:236): avc:  denied  { getattr } for  pid=1911
comm="httpd" path="/lab-content/lab.html" dev="sda3" ino=8640016
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0

type=SYSCALL msg=audit(1749771030.133:236): arch=x86_64 syscall=newfstatat
success=no exit=EACCES a0=fffffff9c a1=7fa59c004928 a2=7fa5aafd49f0 a3=100
items=0 ppid=1908 pid=1911 auid=4294967295 uid=48 gid=48 euid=48 suid=48
fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm=httpd
exe=/usr/sbin/httpd subj=system_u:system_r:httpd_t:s0 key=(null)

Hash: httpd,httpd_t,default_t,file,getattr
```

- 3.3.** The Raw Audit Messages section of the sealert command contains information from the /var/log/audit/audit.log file. Search that file. The -m option searches on the message type. The ts option searches based on time. The following entry identifies the relevant process, file, and contexts that caused the alert. The process is the httpd Apache web server, the file is /lab-content/lab.html, and the context is system\_r:httpd\_t.

```
[root@serverb ~]# ausearch -m AVC -ts recent
...output omitted...
-----
time->Thu Jun 12 23:30:30 2025
type=PROCTITLE msg=audit(1749771030.133:236):
proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1749771030.133:236): arch=c000003e syscall=262 success=no
exit=-13 a0=fffffff9c a1=7fa59c004928 a2=7fa5aaf49f0 a3=100 items=0 ppid=1908
pid=1911 auid=4294967295 uid=48 gid=48 euid=48 suid=48 egid=48 sgid=48
fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1749771030.133:236): avc: denied { getattr } for pid=1911
comm="httpd" path="/lab-content/lab.html" dev="sda3" ino=8640016
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

- 4.** Display the SELinux context of the new HTTP document directory and the original HTTP document directory. Resolve the SELinux issue that prevents the Apache server from serving web content.

- 4.1.** Compare the SELinux context for the /lab-content and /var/www/html directories.

```
[root@serverb ~]# ls -dZ /lab-content /var/www/html
unconfined_u:object_r:default_t:s0 /lab-content
system_u:object_r:httpd_sys_content_t:s0 /var/www/html
```

- 4.2.** Create a file context rule that sets the default type to httpd\_sys\_content\_ for the /lab-content directory and all the files in it.

```
[root@serverb ~]# semanage fcontext -a -t httpd_sys_content_t '/lab-
content(/.*)?'
```

- 4.3.** Correct the SELinux context for the files in the /lab-content directory.

```
[root@serverb ~]# restorecon -Rv /lab-content/
Relabeled /lab-content from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /lab-content/lab.html from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

- 5.** Verify that the Apache server can now serve web content on the workstation machine.

- 5.1.** Use a web browser on the workstation machine to access the `http://serverb/lab.html` URL. If the following content is displayed, then the issue is resolved.

This is the html file for the SELinux final lab on SERVERB.

**6.** Return to the workstation machine as the student user.

```
[root@serverb ~]# exit  
logout  
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
student@workstation:~$
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade selinux-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish selinux-review
```

## 6.10. Summary

---

In this lesson, you learned about SELinux and how it can enhance the security of a Linux system.

SELinux has three modes: enforcing, permissive, and disabled, and you can switch between them to affect the behavior of SELinux. The SELinux policies define the rules for Mandatory Access Control on a system.

SELinux contexts enforce security policies. The `chcon` command changes the context of files and directories regardless of the SELinux policy. The `semanage fcontext` command manages the policy to modify SELinux file contexts, and the `restorecon` command applies these contexts to files and directories.

SELinux Booleans enable or disable the SELinux policy's optional settings, to selectively tune the behavior of an application. The `getsebool` command views the SELinux Booleans and their current state, and the `setsebool` command enables or disables them, either temporarily or persistently.

The `/var/log/audit/audit.log` file stores SELinux events in raw format, and the `/var/log/messages` file stores generic messages from the `setroubleshootd` daemon. Those log files, along with the `sealert` command, can analyze SELinux events and give suggestions for resolving issues.

## **Chapter 7.**

### **Archiving Files**

---

#### **Goal**

Create compressed archives of files so that they can be backed up and transferred to other systems.

#### **Sections**

- Managing Compressed Tar Archives (and Guided Exercise)

## 7.1. Managing Compressed Tar Archives

### Objectives

- Describe compressing tar archives to list the files and directories and extract them.

### Create Archives from the Command Line

An *archive* is a single regular file or device file that contains multiple files. The device file could be a tape drive, flash drive, or other removable media. When using a regular file, archiving is analogous to the `zip` utility and similar variations that are popular on most operating systems.

Archive files are used to create manageable personal backups, or to simplify transferring a set of files across a network when other methods, such as the `rsync` utility, are unavailable or might be more complex. Archive files can be created with or without using compression to reduce the archive file size.

#### Note

The original, ubiquitous `zip` compression and file packaging utility uses the PKZIP (Phil Katz's ZIP for MS-DOS systems) algorithm, and is supported on RHEL with the `zip` and `unzip` commands. Many other compression algorithms have been developed since `zip` was introduced, and each one has its advantages in speed or compression ratio. For creating compressed archives for general use, any `tar`-supported compression algorithm is acceptable.

On Linux, the `tar` utility is the common command to create, manage, and extract archives. Use the `tar` command to gather multiple files into a single archive file. A *tar archive* is a structured sequence of file metadata and data with an index so you can extract individual files.

Files can be compressed during creation by using one of the supported compression algorithms. The `tar` command can list the contents of an archive without extracting, and can extract original files directly from both compressed and uncompressed archives.

### Options of the Tar Utility

One of the following `tar` command actions is required to perform a `tar` operation:

- `-c` or `--create` : Create an archive file.
- `-t` or `--list` : List the contents of an archive.
- `-x` or `--extract` : Extract an archive.

The following `tar` command general options are often included:

- `-v` or `--verbose`: Show the files that are being archived or extracted during the `tar` operation.
- `-f` or `--file`: Follow this option with the archive file name to create or open.
- `-p` or `--preserve-permissions`: Preserve the original file permissions when extracting.
- `--xattrs`: Enable extended attribute support, and store extended file attributes.
- `--selinux`: Enable SELinux context support, and store SELinux file contexts.

The following `tar` command compression options are used to select an algorithm:

- `-a` or `--auto-compress`: Use the archive's suffix to determine the algorithm to use.
- `-z` or `--gzip`: Use the gzip compression algorithm, which results in a `.tar.gz` suffix.
- `-j` or `--bzip2`: Use the bzip2 compression algorithm, which results in a `.tar.bz2` suffix.
- `-J` or `--xz`: Use the xz compression algorithm, which results in a `.tar.xz` suffix.

### Note

The `tar` command still supports the legacy option style that does not use a dash (-) character. You might find this syntax in legacy scripts or documentation, and the behavior is essentially the same. For command consistency, Red Hat recommends using the short- or long-option styles instead.

## Create an Archive

To create an archive with the `tar` command, use the `-c` and `-f` options with the archive file name as the first argument, followed by a list of files and directories to include in the archive.

The `tar` command recognizes absolute and relative file name syntax. By default, `tar` removes the leading forward slash (/) character from absolute file names, so that files are stored internally with relative path names. This technique is safer, because extracting absolute path names always overwrites existing files. With files that are archived with relative path names, files can be extracted to a new directory without overwriting existing files.

The following command creates the `mybackup.tar` archive to contain the `myapp1.log`, `myapp2.log`, and `myapp3.log` files from the user's home directory. If a file with the same name as the requested archive exists in the target directory, then the `tar` command overwrites the file.

```
user@host:~$ tar -cf mybackup.tar myapp1.log myapp2.log myapp3.log
```

A user must have read permission on the target files that are being archived. For example, creating an archive in the /etc directory requires root privileges, because only privileged users can read all /etc files. An unprivileged user can create an archive of the /etc directory, but the archive excludes files that the user cannot read, and excludes directories for which the user lacks the read and execute permissions.

In this example, the root user creates the /root/etc-backup.tar archive of the /etc directory.

```
root@host:~# tar -cf /root/etc-backup.tar /etc
tar: Removing leading `/' from member names
```

### Important

Extended file attributes, such as access control lists (ACL) and SELinux file contexts, are not preserved by default in an archive. Use the --acls option to include POSIX ACLs; use the --selinux option to include SELinux file contexts; or use the --xattrs option to include other extended attributes.

Although extended attributes are outside the scope of this lesson, you can find more information in the references section.

## List Archive Contents

Use the tar command -t option to list the file names from within the archive that is specified with the -f option. The files are listed with relative name syntax, because the leading forward slash was removed during archive creation.

```
root@host:~# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

## Extract Archive Contents

Extract a tar archive to an empty directory to avoid overwriting existing files. When the root user extracts an archive, the extracted files preserve the original user and group ownership. If a regular user

extracts files, then the user becomes the owner of the extracted files.

Create the /root/etcbackup directory to extract the tar archive.

```
root@host:~# mkdir /root/etcbackup
```

List the file names in the /root/etc.tar archive.

```
root@host:~# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

Extract the archive to the /root/etcbackup directory:

```
root@host:~/etcbackup# tar -xf /root/etc.tar
```

When you extract files from an archive, the current umask is used to modify each extracted file's permissions. Instead, use the tar command -p option to preserve the original archived permissions for extracted files. The -p option is enabled by default for a superuser.

```
user@host:~/scripts$ tar -xpf /home/user/myscripts.tar
```

## Create a Compressed Archive

The tar command supports these compression methods, and others:

- The **gzip** compression is the earlier, fastest method, and is widely available across platforms.
- The **bzip2** compression creates smaller archives but is less widely available than gzip.
- The **xz** compression is later, and offers the best compression ratio of the available methods.

The effectiveness of any compression algorithm depends on the type of data that is compressed. Previously compressed data files, such as picture formats or RPM files, typically do not significantly compress further, regardless of the compression algorithm that is used.

Create the /root/etcbackup.tar.gz archive with gzip compression from the contents of the /etc directory:

```
root@host:~# tar -czf /root/etcbackup.tar.gz /etc
tar: Removing leading `/' from member names
```

Create the /root/logbackup.tar.bz2 archive with bzip2 compression from the contents of the /var/log directory:

```
root@host:~# tar -cjf /root/logbackup.tar.bz2 /var/log
tar: Removing leading `/' from member names
```

Create the /root/sshconfig.tar.xz archive with xz compression from the contents of the /etc/ssh directory:

```
root@host:~# tar -cJf /root/sshconfig.tar.xz /etc/ssh
tar: Removing leading `/' from member names
```

After creating a compressed archive, you can still verify its table of contents with the `tar` command - `-tf` options. It is not necessary to specify the compression option when listing a compressed archive file, because the compression type is read from the archive's header.

List the archived content in the /root/etcbackup.tar.gz file, which uses the gzip compression:

```
root@host:~# tar -tf /root/etcbackup.tar.gz
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

## Extract Compressed Archive Contents

The `tar` command can automatically determine which compression was used, so it is not necessary to specify the compression option. If you include an incorrect compression type, then the `tar` command reports that the specified compression type does not match the file's type. Listing a compressed `tar` archive works in the same way as listing an uncompressed `tar` archive.

In the following example, the `tar` command uses the `-z` option, which indicates gzip compression, but the file name extension is `.xz`, which indicates xz compression:

```
root@host:~# tar -xzf /root/etcbackup.tar.xz  
  
gzip: stdin: not in gzip format  
tar: Child returned status 1  
tar: Error is not recoverable: exiting now
```

The gzip, bzip2, and xz algorithms are also implemented as stand-alone commands for compressing individual files without creating an archive. With these commands, you cannot create a single compressed file of multiple files, such as a directory. As previously discussed, to create a compressed archive of multiple files, use the `tar` command with your preferred compression option. To uncompress a single compressed file or a compressed archive file without extracting its contents, use the gunzip, bunzip2, or unxz stand-alone commands.

The gzip and xz commands provide the `-l` option to view the uncompressed size of a compressed single or archive file. Use this option to verify that enough space is available before uncompressing or extracting a file.

Use the gzip command to view the uncompressed size of the `file.tar.gz` archive.

```
user@host:~$ gzip -l file.tar.gz  
      compressed      uncompressed   ratio   uncompressed_name  
           221603125          303841280  27.1%   file.tar
```

Use the xz command to view the uncompressed size of the `file.tar.gz` archive.

```
user@host:~$ xz -l file.tar.xz  
  Strms   Blocks   Compressed   Uncompressed   Ratio   Check   Filename  
    1       1     195.7 MiB     289.8 MiB  0.675  CRC64   file.xz
```

## References

`tar(1)`, `gzip(1)`, `gunzip(1)`, `bzip2(1)`, `bunzip2(1)`, `xz(1)`, `unxz(1)`, `xattr(7)` and `acl(5)` man pages

## 7.2. Guided Exercise

### Manage Compressed Tar Archives

Create a compressed tar archive, list the files and directories that are stored in it, and extract them.

#### Outcomes

- Archive a directory tree and extract the archive content to another location.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start archive-manage
```

#### Instructions

- From the workstation machine, log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Create archives of the /etc directory, both without compression, and with the gzip, bzip2, and xz compression algorithms. Compare the sizes of the resulting archive files.

- Create a tar archive of the /etc directory. Save the archive file as the /tmp/etc.tar archive.

```
[root@servera ~]# tar -cf /tmp/etc.tar /etc
...output omitted...
```

- Create a tar archive of the /etc directory that is compressed by the gzip compression algorithm. Save the archive file as the /tmp/etc.tar.gz archive.

```
[root@servera ~]# tar -czf /tmp/etc.tar.gz /etc  
...output omitted...
```

- 2.3.** Create a tar archive of the /etc directory that is compressed by the bzip2 compression algorithm. Save the archive file as the /tmp/etc.tar.bz2 archive.

```
[root@servera ~]# tar -cjf /tmp/etc.tar.bz2 /etc  
...output omitted...
```

- 2.4.** Create a tar archive of the /etc directory that is compressed by the xz compression algorithm. Save the archive file as the /tmp/etc.tar.xz archive.

```
[root@servera ~]# tar -cJf /tmp/etc.tar.xz /etc  
...output omitted...
```

- 2.5.** Use the ls -lh command to compare the sizes of the resulting archive files.

```
[root@servera ~]# ls -lh /tmp/etc.tar*  
-rw-r--r--. 1 root root 22M Jun 4 11:13 /tmp/etc.tar  
-rw-r--r--. 1 root root 4.7M Jun 4 11:54 /tmp/etc.tar.bz2  
-rw-r--r--. 1 root root 5.5M Jun 4 11:30 /tmp/etc.tar.gz  
-rw-r--r--. 1 root root 4.1M Jun 4 11:56 /tmp/etc.tar.xz
```

- 3.** Verify that the etc.tar.gz archive contains the files from the /etc directory.

```
[root@servera ~]# tar -tzf /tmp/etc.tar.gz  
etc/  
etc/xdg/  
etc/xdg/autostart/  
etc/xdg/systemd/  
etc/xdg/systemd/user  
...output omitted...
```

- 4.** Create the /backuptest directory. Verify that the etc.tar.gz backup file is a valid archive by decompressing the file to the /backuptest directory.

- 4.1.** Create the /backuptest directory and change to that directory.

```
[root@servera ~]# mkdir /backuptest  
[root@servera ~]# cd /backuptest  
[root@servera backuptest]#
```

- 4.2.** Extract the /tmp/etc.tar.gz archive to the /backuptest directory.

```
[root@servera backuptest]# tar -xzf /tmp/etc.tar.gz
```

- 4.3.** List the contents of the /backuptest directory.

```
[root@servera backuptest]# ls -l  
total 12  
drwxr-xr-x. 112 root root 8192 Jun 4 12:30 etc
```

- 4.4.** Verify that the directory contains the /etc directory backup files.

```
[root@servera backuptest]# ls -l etc  
total 1228  
-rw-r--r--. 1 root root 5923 Nov 26 2024 DIR_COLORS  
-rw-r--r--. 1 root root 6005 Nov 26 2024 DIR_COLORS.lightbgcolor  
-rw-r--r--. 1 root root 94 Oct 29 2024 GREP_COLORS  
drwxr-xr-x. 7 root root 134 Apr 23 14:35 NetworkManager  
drwxr-xr-x. 2 root root 48 Apr 23 14:35 PackageKit  
drwxr-xr-x. 5 root root 51 Apr 23 14:35 X11  
-rw-r--r--. 1 root root 12 Feb 13 00:00 adjtime  
-rw-r--r--. 1 root root 1529 Nov 29 2023 aliases  
...output omitted...
```

- 5.** Return to the workstation machine as the student user.

```
[root@servera backuptest]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish archive-manage
```

## 7.3. Summary

---

In this lesson, you learned about tools to archive and compress files and directories.

The `tar` utility is a versatile command to create, list, and extract archive files. It can handle both single files and entire directories, and can preserve metadata and special attributes. The `tar` utility can also compress archives by using the `gzip`, `bzip2`, and `xz` compression algorithms.

The `gzip`, `bzip2`, and `xz` compression tools can also be used as stand-alone commands. Each of these tools has its own compression algorithm, with varying levels of compression and speed that depend on the data that is compressed.

## **Chapter 8.**

### **Transferring Files**

---

#### **Goal**

Securely transfer files from one system to another.

#### **Sections**

- Transferring Files Between Systems (and Guided Exercise)
- Synchronizing Content Between Systems (and Guided Exercise)

#### **Lab**

- Transfer Files

## 8.1. Transferring Files Between Systems

### Objectives

- Use SSH utilities to securely transfer files to and from remote systems.

### Transfer Remote Files with the Secure File Transfer Program

The OpenSSH suite securely runs shell commands on remote systems. Use the *Secure File Transfer Program* (SFTP) to interactively upload files to or download files from an SSH server. This program is part of the OpenSSH suite. A session with the `sftp` command uses the secure authentication mechanism and encrypted data transfer to and from the SSH server.

Specify a remote location for the source or destination of the files to copy. For the format of the remote location, use `user@host:path`. The `user@` part of the argument is optional; if this part is missing, then the `sftp` command uses the current local user to log in as the remote user.

When you run the `sftp` command by providing the remote user and host, the terminal opens an interactive session.

```
user@host:~$ sftp remoteuser@remotehost
remoteuser@remotehost's password: password
Connected to remotehost.
sftp>
```

The interactive `sftp` session accepts various commands that work the same way in the remote file system as in the local file system, such as the `ls`, `cd`, `mkdir`, `rmdir`, and `pwd` commands.

- The `put` command uploads a file to the remote system.
- The `get` command downloads a file from the remote system.
- The `exit` command exits the `sftp` session.

List the available `sftp` commands by using the `help` command in the `sftp` session:

```
sftp> help
Available commands:
bye                                Quit sftp
cd path                             Change remote directory to 'path'
chgrp [-h] grp path                 Change group of file 'path' to 'grp'
chmod [-h] mode path                Change permissions of file 'path' to 'mode'
chown [-h] own path                Change owner of file 'path' to 'own'
...output omitted...
```

In an sftp session, you might run some commands on your local host by using the `l` character before the command.

For example, the `pwd` command prints the current working directory on the remote host.

```
sftp> pwd
Remote working directory: /home/remoteuser
```

To print the current working directory on your local host, use the `lpwd` command.

```
sftp> lpwd
Local working directory: /home/user
```

In the following example, upload the `/etc/hosts` file from the local system to the newly created `/home/remoteuser/hostbackup` directory on the `remotehost` system.

```
sftp> mkdir hostbackup
sftp> cd hostbackup
```

Change to the `hostbackup` directory on the `remotehost` system.

```
sftp> cd hostbackup
```

The sftp session expects that the `put` command is followed by a local file in the connecting user's home directory, in this case the `/home/remoteuser` directory:

```
sftp> put /etc/hosts
Uploading /etc/hosts to /home/remoteuser/hostbackup/hosts
/etc/hosts                                         100%   227      0.2KB/s  00:00
```

To copy a whole directory tree recursively, use the `sftp` command `-r` option.

The following example recursively copies the /home/user/directory local directory to the remotehost machine.

```
sftp> put -r directory
Uploading directory/ to /home/remoteuser/directory
Entering directory/
file1                      100%   0      0.0KB/s  00:00
file2                      100%   0      0.0KB/s  00:00
```

To download the /etc/dnf/dnf.conf file from the remote host to the current directory on the local system, use the get command.

```
sftp> get /etc/dnf/dnf.conf
Fetching /etc/dnf/dnf.conf to dnf.conf
/etc/dnf.conf                100%  813      0.8KB/s  00:00
```

To get a remote file with the sftp command on a single command line, without opening an interactive session, use the following syntax. You cannot use single command-line syntax to put files on a remote host.

```
user@host:~$ sftp remoteuser@remotehost:/home/remoteuser/remotefile
Connected to remotehost.
Fetching /home/remoteuser/remotefile to remotefile
remotefile                           100%    7
15.7KB/s  00:00
```

## Transfer Files with the Secure Copy Command

The *Secure Copy Protocol* (SCP) uses the `scp` command, which is also part of the OpenSSH suite, and copies files from a remote system to the local system, or from the local system to a remote system.

Starting in RHEL 10, the `scp` command uses SFTP to transfer files.

You can specify a remote location for the source or destination of the files that you are copying. The `scp` command uses the `user@host` remote username with hostname/IP address to identify the target system and user. If you do not specify a user, then the command attempts to log in as a remote user by using the current local user.

The `scp` command can also transfer files with the legacy SCP. This protocol contains a code injection vulnerability, therefore Red Hat recommends that you do not use the `scp` command with the legacy SCP. To revert to the legacy `scp` protocol, you can include the `-O` flag.

## Important

Starting in RHEL 10, the use of the scp protocol can be disabled on the system. If the file /etc/ssh/disable\_scp exists, then any attempt to use the scp protocol might fail, and the -O option cannot be used, but the sftp protocol still works.

The following example demonstrates how to copy the local /etc/dnf/dnf.conf and /etc/hosts files to the remote host's home directory:

```
user@host:~$ scp /etc/dnf/dnf.conf /etc/hosts  
remoteuser@remotehost:/home/remoteuser  
remoteuser@remotehost's password: password  
dnf.conf                                100%   813      0.8KB/s  00:00  
hosts                                    100%   227      0.2KB/s  00:00
```

You can also copy a file in the other direction, from a remote system to the local file system.

In this example, the file /etc/hostname on the remote host is copied to the local /home/user directory. Here, the scp command authenticates to the remote host as the remoteuser user.

```
[user@host ~]$ scp remoteuser@remotehost:/etc/hostname /home/user  
remoteuser@remotehost's password: password  
hostname
```

## References

sftp(1) man pages, scp(1) man pages

For more information about OpenSSH SCP deprecation, in RHEL 9 and later versions, refer to [OpenSSH SCP Deprecation in, RHEL 9: What You Need to Know](#)

For more information about issues with SCP, refer to [CVE-2020-15778](#)

## 8.2. Guided Exercise

### Transfer Files Between Systems

Securely transfer files to and from remote systems by using SSH utilities.

#### Outcomes

- Copy files from a remote host to a directory on the local machine.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start rcopy-transfer
```

#### Instructions

1. Log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Create /home/student/serverbackup1 and /home/student/serverbackup2 directories.

```
[student@servera ~]$ mkdir ~/serverbackup{1,2}
```

3. Use the sftp command to interactively copy the /etc/ssh directory from the serverb machine to the /home/student/serverbackup1 directory on the servera machine.

- 3.1. Use the sftp command to open a session to the serverb machine as the root user. Only the root user can read all the contents in the /etc/ssh directory. Use redhat as the password.

```
[student@servera ~]$ sftp root@serverb  
root@serverb's password: redhat  
Connected to serverb.  
sftp>
```

- 3.2.** Change the local current directory to the /home/student/serverbackup1 directory.

```
sftp> lcd /home/student/serverbackup1/
```

- 3.3.** Recursively copy the /etc/ssh directory from the serverb machine to the /home/student/serverbackup1 directory on the servera machine.

```
sftp> get -r /etc/ssh  
Fetching /etc/ssh/ to ssh  
  Retrieving /etc/ssh  
    Retrieving /etc/ssh/sshd_config.d  
      50-redhat.conf          100%   581   330.3KB/s  00:00  
      01-training.conf        100%    36   23.9KB/s  00:00  
    Retrieving /etc/ssh/sshd_config.d  
      40-redhat-crypto-policies.conf 100%   412   1.2MB/s  00:00  
      50-redhat.conf          100%   307   829.8KB/s  00:00  
      50-cloud-init.conf       100%    27   87.0KB/s  00:00  
      moduli                  100%  529KB  40.3MB/s  00:00  
      ssh_config               100%  1916   1.6MB/s  00:00  
      sshd_config              100%  3712   8.4MB/s  00:00  
      ssh_host_rsa_key         100%  2602   6.5MB/s  00:00  
      ssh_host_rsa_key.pub     100%   566   1.6MB/s  00:00  
      ssh_host_ecdsa_key       100%   505   1.2MB/s  00:00  
      ssh_host_ecdsa_key.pub   100%   174   429.9KB/s  00:00  
      ssh_host_ed25519_key     100%   399   1.2MB/s  00:00  
      ssh_host_ed25519_key.pub 100%    94   269.0KB/s  00:00
```

- 3.4.** Exit from the sftp session.

```
sftp> exit
```

- 3.5.** Verify that the /etc/ssh directory from the serverb machine is copied to the /home/student/serverbackup1 directory on the servera machine.

```
[student@servera ~]$ ls -lR ~/serverbackup1
/home/student/serverbackup1:
total 4
drwxr-xr-x. 4 student student 4096 Jul 14 08:10 ssh

/home/student/serverbackup1/ssh:
total 564
-rw-r--r--. 1 student student 541716 Jul 14 08:10 moduli
-rw-r--r--. 1 student student 1916 Jul 14 08:10 ssh_config
drwxr-xr-x. 2 student student 52 Jul 14 08:10 ssh_config.d
-rw-----. 1 student student 505 Jul 14 08:10 ssh_host_ecdsa_key
-rw-r--r--. 1 student student 174 Jul 14 08:10 ssh_host_ecdsa_key.pub
-rw-----. 1 student student 399 Jul 14 08:10 ssh_host_ed25519_key
-rw-r--r--. 1 student student 94 Jul 14 08:10 ssh_host_ed25519_key.pub
-rw-----. 1 student student 2602 Jul 14 08:10 ssh_host_rsa_key
-rw-r--r--. 1 student student 566 Jul 14 08:10 ssh_host_rsa_key.pub
-rw-----. 1 student student 3712 Jul 14 08:10 sshd_config
drwx-----. 2 student student 92 Jul 14 08:10 sshd_config.d

/home/student/serverbackup1/ssh/ssh_config.d:
total 8
-rw-r--r--. 1 student student 36 Jul 14 08:10 01-training.conf
-rw-r--r--. 1 student student 581 Jul 14 08:10 50-redhat.conf

/home/student/serverbackup1/ssh/sshd_config.d:
total 12
-rw-----. 1 student student 412 Jul 14 08:10 40-redhat-crypto-policies.conf
-rw-----. 1 student student 27 Jul 14 08:10 50-cloud-init.conf
-rw-----. 1 student student 307 Jul 14 08:10 50-redhat.conf
```

4. Similarly use the scp command to noninteractively copy the /etc/ssh directory from the serverb machine to the /home/student/serverbackup2 directory on the servera machine.
  - 4.1. Use the scp command to recursively copy the /etc/ssh directory from the serverb machine to the /home/student/serverbackup2 directory on the servera machine. Use redhat as the password.

```
[student@servera ~]$ scp -r root@serverb:/etc/ssh ~/serverbackup2
root@serverb's password: redhat
50-redhat.conf                           100%   581      1.7MB/s  00:00
01-training.conf                          100%    36      37.1KB/s  00:00
40-redhat-crypto-policies.conf           100%   412      1.1MB/s  00:00
50-redhat.conf                           100%   307     888.3KB/s  00:00
50-cloud-init.conf                       100%    27      63.3KB/s  00:00
moduli                                    100%  529KB  117.2MB/s  00:00
ssh_config                                100%  1916      4.1MB/s  00:00
sshd_config                               100%  3712      7.2MB/s  00:00
ssh_host_rsa_key                          100%  2602      6.1MB/s  00:00
ssh_host_rsa_key.pub                      100%   566      1.3MB/s  00:00
ssh_host_ecdsa_key                        100%   505      1.2MB/s  00:00
ssh_host_ecdsa_key.pub                   100%   174     429.7KB/s  00:00
ssh_host_ed25519_key                     100%   399     950.4KB/s  00:00
ssh_host_ed25519_key.pub                 100%    94     211.8KB/s  00:00
```

- 4.2.** Verify that the /etc/ssh directory from the serverb machine is copied to the /home/student/serverbackup2 directory on the servera machine.

```
[student@servera ~]$ ls -lR ~/serverbackup2
/home/student/serverbackup2:
total 4
drwxr-xr-x. 4 student student 4096 Jul 14 08:13 ssh

/home/student/serverbackup2/ssh:
total 564
-rw-r--r--. 1 student student 541716 Jul 14 08:13 moduli
-rw-r--r--. 1 student student 1916 Jul 14 08:13 ssh_config
drwxr-xr-x. 2 student student 52 Jul 14 08:13 ssh_config.d
-rw-----. 1 student student 505 Jul 14 08:13 ssh_host_ecdsa_key
-rw-r--r--. 1 student student 174 Jul 14 08:13 ssh_host_ecdsa_key.pub
-rw-----. 1 student student 399 Jul 14 08:13 ssh_host_ed25519_key
-rw-r--r--. 1 student student 94 Jul 14 08:13 ssh_host_ed25519_key.pub
-rw-----. 1 student student 2602 Jul 14 08:13 ssh_host_rsa_key
-rw-r--r--. 1 student student 566 Jul 14 08:13 ssh_host_rsa_key.pub
-rw-----. 1 student student 3712 Jul 14 08:13 sshd_config
drwx-----. 2 student student 92 Jul 14 08:13 sshd_config.d

/home/student/serverbackup2/ssh/ssh_config.d:
total 8
-rw-r--r--. 1 student student 36 Jul 14 08:13 01-training.conf
-rw-r--r--. 1 student student 581 Jul 14 08:13 50-redhat.conf

/home/student/serverbackup2/ssh/sshd_config.d:
total 12
-rw-----. 1 student student 412 Jul 14 08:13 40-redhat-crypto-policies.conf
-rw-----. 1 student student 27 Jul 14 08:13 50-cloud-init.conf
-rw-----. 1 student student 307 Jul 14 08:13 50-redhat.conf
```

## 5. Return to the workstation machine as the student user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish rcopy-transfer
```

## 8.3. Synchronizing Content Between Systems

### Objectives

- Efficiently and securely synchronize the content between a local file or directory and a remote copy.

### Synchronize Remote Files and Directories

The `rsync` command is another way to copy files from one system to another system securely. The tool uses an algorithm that minimizes copied data by synchronizing only the changed portions of files. If two files or directories are similar between two servers, then the `rsync` command copies only the differences between the file systems.

An advantage of the `rsync` command is that it copies files efficiently and securely between a local system and a remote system. Whereas an initial directory synchronization takes similar time to copying it, subsequent synchronizations copy only the differences over the network, which substantially accelerates updates.

Use the `rsync` command `-n` option for a dry run, to simulate what happens when the command is executed. The dry run shows the changes that the `rsync` command would perform when executing the command. Perform a dry run before the actual `rsync` command operation to ensure that no critical files are overwritten or deleted.

When synchronizing with the `rsync` command, two standard options are the `-v` and `-a` options.

The `rsync` command `-v` or `--verbose` option provides a more detailed output. This option is helpful for troubleshooting and viewing live progress.

— The `rsync` command `-a` or `--archive` option enables archive mode. This option enables recursive copying and turns on other options to preserve most characteristics of the files.

Archive mode is the same as specifying the following options:

Table 13. Options in Archive Mode

Option	Description
<code>-r, --recursive</code>	This option synchronizes the whole directory tree recursively.

Option	Description
-l, --links	This option synchronizes symbolic links.
-p, --perms	This option preserves permissions.
-t, --times	This option preserves time stamps.
-g, --group	This option preserves group ownership.
-o, --owner	This option preserves the owner of the files.
-D, --devices	This option preserves device files.

Archive mode does not preserve hard links, because it might add significant time to the synchronization. Use the `rsync` command `-H` option to preserve hard links.

#### Note

To include extended attributes when synchronizing files, add these options to the `rsync` command:

- `-A` to preserve Access Control Lists (ACLs)
- `-X` to preserve SELinux file contexts

You can use the `rsync` command to synchronize the contents of a local file or directory with a file or directory on a remote machine, with either machine as the source. You can also synchronize the contents of two local files or directories on the same machine.

Like the `sftp` command, the `rsync` command specifies remote locations in the `user@host: path` format. The remote location can be either the source or the destination system, provided that either one machine is a remote system and the other is a local system, or both machines are local systems.

To preserve file ownership on the destination system, you must be the `root` user. If the destination is remote, then authenticate as the `root` user. If the destination is local, then you must run the `rsync` command as the `root` user.

The following example synchronizes the local `/var/log` directory from the host machine to the `/tmp` directory on the `hosta` system:

```
root@host:~# rsync -av /var/log hosta:/tmp
root@hosta's password: password
receiving incremental file list
log/
log/README
log/boot.log
...output omitted...

sent 9,783 bytes  received 290,576 bytes  85,816.86 bytes/sec
total size is 11,585,690  speedup is 38.57
```

The following example synchronizes the `/var/log` remote directory from the `hosta` machine to the `/tmp` directory on the host machine:

```
root@host:~# rsync -av hosta:/var/log /tmp
root@hosta's password: password
receiving incremental file list
log/boot.log
log/dnf.librepo.log
log/dnf.log
...output omitted...

sent 9,783 bytes  received 290,576 bytes  85,816.86 bytes/sec
total size is 11,585,690  speedup is 38.57
```

The following example synchronizes the contents of the `/var/log` directory to the `/tmp` directory on the same machine and uses the `sudo` command to escalate to the `root` user:

```
user@host:~$ sudo rsync -av /var/log /tmp
[sudo] password for user: password
receiving incremental file list
log/
log/README
log/boot.log
...output omitted...
log/tuned/tuned.log

sent 11,592,423 bytes received 779 bytes 23,186,404.00 bytes/sec
total size is 11,586,755 speedup is 1.00
user@host:~$ ls /tmp
log  ssh-RLjDdarkKiW1
```

### Important

Correctly specifying a trailing slash with a source directory is important. A trailing slash with a source directory synchronizes the contents of the directory without including the directory itself. The contents are synchronized directly into the destination directory.

Without the trailing slash, the source directory itself might synchronize to the destination directory. The source directory's contents are in the new subdirectory in the destination.

Bash tab-completion automatically adds a trailing slash to directory names.

In this example, the contents of the `/var/log/` directory are synchronized to the `/tmp` directory instead of to the `log` directory that is created in the `/tmp` directory.

```
root@host:~# rsync -av /var/log/ /tmp
sending incremental file list
./
README
boot.log
...output omitted...
tuned/tuned.log

sent 11,592,389 bytes  received 778 bytes  23,186,334.00 bytes/sec
total size is 11,586,755  speedup is 1.00
root@host:~# ls /tmp
anaconda                      dnf.rpm.log-20190318  private
audit                          dnf.rpm.log-20190324  qemu-ga
boot.log                       dnf.rpm.log-20190331  README
...output omitted...
```

## References

[rsync\(1\) man page](#)

## 8.4. Guided Exercise

### Synchronize Content Between Systems

Synchronize the content of a local directory with a copy on a remote server.

#### Outcomes

- Synchronize the contents of a local directory with a copy on a remote server.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start rcopy-sync
```

#### Instructions

1. On the `serverb` machine, create the `/home/student/serverlogs` directory.

- 1.1. Log in to the `serverb` machine as the `student` user.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. On the `serverb` machine, create the `/home/student/serverlogs` directory to store the synchronized log files from the `servera` machine.

```
[student@serverb ~]$ mkdir ~/serverlogs
```

2. Open a new terminal window and log in to the `servera` machine as the `student` user, and then switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

3. On the servera machine, use the rsync command to synchronize the /var/log directory on the servera machine with the /home/student/serverlogs directory on the serverb machine.

- 3.1. On the servera machine, use the rsync command to synchronize the /var/log directory on the servera machine with the /home/student/serverlogs directory on the serverb machine. Only the root user can read all the /var/log directory contents on the servera machine. Transfer all the files in the initial synchronization. Use student as the password to log in to the serverb machine as the student user.

```
[root@servera ~]# rsync -av /var/log student@serverb:/home/student/serverlogs
...output omitted...
student@serverb's password: student
sending incremental file list
log/
log/boot.log
log/boot.log-20250708
log/btmp
log/btmp-20250708
log/cloud-init-output.log
...output omitted...

sent 3,251,516 bytes received 676 bytes 260,175.36 bytes/sec
total size is 3,248,149 speedup is 1.00
```

4. On the servera machine, execute the logger "Log files synchronized" command to add an entry in the /var/log/messages log file to reflect when the last synchronization occurred.

```
[root@servera ~]# logger "Log files synchronized"
```

5. On the servera machine, use the rsync command to securely synchronize the /var/log directory on the servera machine with the /home/student/serverlogs directory on the serverb machine. This time, only the changed log files are transferred.

```
[root@servera ~]# rsync -av /var/log student@serverb:/home/student/serverlogs
student@serverb's password: student
sending incremental file list
log/messages
log/audit/audit.log

sent 10,435 bytes received 7,801 bytes 3,315.64 bytes/sec
total size is 3,251,420 speedup is 178.30
```

6. On the serverb machine, verify the contents of the /home/student/serverlogs/log/messages file by using the tail command.

```
[student@serverb ~]$ tail -n 5 ~/serverlogs/log/messages
Jul  8 16:04:43 servera systemd[1]: Starting cockpit-wsinstance-
https@e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855.socket
- Socket for Cockpit Web Service https instance
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855...
Jul  8 16:04:43 servera systemd[1]: Listening on cockpit-wsinstance-
https@e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855.socket
- Socket for Cockpit Web Service https instance
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855.
Jul  8 16:04:43 servera systemd[1]: Started cockpit-wsinstance-
https@e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855.service
- Cockpit Web Service https instance
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855.
Jul  8 16:04:43 servera systemd[1]: cockpit-wsinstance-https-factory@41-1291-
61789.service: Deactivated successfully.
Jul  8 16:05:30 servera root[4259]: Log files synchronized
```

7. Close other terminal windows and return to the workstation machine as the student user.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish rcopy-sync
```

## 8.5. Lab

### Transfer Files

Securely synchronize a local file or directory with a remote copy.

#### Outcomes

- Synchronize a directory from a remote host to a local host.
- Securely copy a file to a remote host.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start rcopy-review
```

#### Instructions

1. On the serverb machine, synchronize the /etc directory tree from the servera machine to the /configsync directory by using the rsync command in archive mode.
2. Securely copy the /root/securedoc file from the serverb machine to the /home/student directory on the workstation machine by using the sftp command. Use student as the password.
3. Return to the workstation machine as the student user.
4. View the contents of the securedoc file on the workstation machine.

#### Evaluation

— As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade rcopy-review
```

#### Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish rcopy-review
```

## Solution

---

Securely synchronize a local file or directory with a remote copy.

## Outcomes

- Synchronize a directory from a remote host to a local host.
- Securely copy a file to a remote host.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start rcopy-review
```

## Instructions

1. On the serverb machine, synchronize the /etc directory tree from the servera machine to the /configsync directory by using the rsync command in archive mode.

- 1.1. Log in to the serverb machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 1.2. Create the /configsync directory to store the synchronized files from the servera machine.

```
[root@serverb ~]# mkdir /configsync
```

- 1.3. Synchronize the /etc directory tree from the servera machine to the /configsync directory on the serverb machine.

```
[root@serverb ~]# rsync -av root@servera:/etc /configsync
...output omitted...
receiving incremental file list
etc/
etc/.pwd.lock
etc/.rht_authorized_keys
etc/.updated
...output omitted...
sent 16,649 bytes received 22,010,221 bytes 14,684,580.00 bytes/sec
total size is 21,925,844 speedup is 1.00
```

**1.4.** View the contents of the /configsync directory.

```
[root@serverb ~]# ls -l /configsync
total 12
drwxr-xr-x. 111 root root 8192 Jul 11 06:29 etc
```

**2.** Securely copy the /root/securedoc file from the serverb machine to the /home/student directory on the workstation machine by using the sftp command. Use student as the password.

```
[root@serverb ~]# sftp student@workstation
...output omitted...
student@workstation's password: student
Connected to workstation.
sftp> put securedoc
Uploading securedoc to /home/student/securedoc
securedoc                                100%   30   68.9KB/s  00:00
sftp> exit
```

**3.** Return to the workstation machine as the student user.

```
[root@serverb ~]# exit
[student@serverb ~]$ exit
student@workstation:~$
```

**4.** View the contents of the securedoc file on the workstation machine.

```
student@workstation:~$ cat securedoc
Secure copy file from serverb
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade rcopy-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish rcopy-review
```

## 8.6. Summary

---

In this lesson, you learned how to securely transfer and synchronously copy files and directories between remote systems.

Besides providing a secure remote shell, OpenSSH also provides the sftp and scp commands to transfer files securely to and from a remote system. Starting in Red Hat Enterprise Linux 10, the use of SCP can be disabled on the system.

The rsync command securely synchronizes files between a local directory and a remote directory.

## Chapter 9.

# Tuning System Performance

---

## Goal

Improve system performance by setting a tuning profile and by adjusting the scheduling priority of specific processes.

## Sections

- Setting a Tuning Profile (and Guided Exercise)
- Influencing Process Scheduling (and Guided Exercise)

## Lab

- Tune System Performance

## 9.1. Setting a Tuning Profile

### Objectives

- Optimize system performance by selecting the tuning daemon's most appropriate profile.

### Tuning Systems

System administrators seek ways to get the best performance out of their hardware, by fine-tuning system settings to meet different demands. TuneD is a powerful tool for Red Hat Enterprise Linux that simplifies this process. It automatically optimizes system performance and power efficiency by applying predefined *tuning profiles* that are tailored to specific workload requirements, to ensure that your system is always running optimally, whether it is a high-performance server or a power-conscious laptop.

TuneD runs continuously in the background as a daemon. It can apply tuning adjustments statically or dynamically. In Red Hat Enterprise Linux, static tuning is the default mode of operation for TuneD.

### The Default Static Tuning Mode

Static tuning configures kernel parameters and other system settings as defined in tuning profiles. When the TuneD service starts, or if you switch to a different profile, these settings are applied immediately. Because TuneD is enabled to run on boot by default, these settings are applied every time that your system starts. These settings remain constant, and provide a stable baseline for overall performance without further adjustments, even when system activity changes.

### Configuring Dynamic Tuning

Although static tuning offers a robust and predictable foundation, TuneD also provides dynamic tuning. This feature enables the tuned daemon to constantly monitor system activity and to adjust settings when the runtime behavior changes. If you enable this feature, then your system's tuning continuously adapts to the current workload, and builds on the initial settings from your chosen profile.

To accomplish this monitoring and changes, the tuned daemon uses specialized modules named monitor plug-ins and tuning plug-ins, respectively.

*Monitor plug-ins* analyze the system and gather real-time data such as CPU load, disk I/O, and network traffic.

*Tuning plug-ins* then use this information for dynamic tuning. For example, the net plug-in can dynamically change the interface speed based on interface usage.

Dynamic tuning is disabled by default in Red Hat Enterprise Linux to ensure predictable performance. You can enable dynamic tuning by changing the `dynamic_tuning` variable to 1 in the `/etc/tuned/tuned-main.conf` configuration file. If you enable dynamic tuning, then the `tuned` daemon periodically monitors the system and adjusts settings based on its runtime behavior.

You can control how often these updates happen by setting the `update_interval` variable (in seconds) in the `/etc/tuned/tuned-main.conf` configuration file.

```
...output omitted...
# Dynamically tune devices, if disabled only static tuning will be used.
dynamic_tuning = 1
...output omitted...
# Update interval for dynamic tunings (in seconds).
# It must be multiply of the sleep_interval.
update_interval = 10
...output omitted...
```

## Installing and Enabling the Tuning Service

In Red Hat Enterprise Linux, the `TuneD` tool is typically preinstalled.

Use the following command to install the `tuned` package:

```
root@host:~$ dnf install tuned
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

Use the following command to enable the `tuned` service:

```
root@host:~$ systemctl enable --now tuned
Created symlink /etc/systemd/system/multi-user.target.wants/tuned.service →
/usr/lib/systemd/system/tuned.service.
```

The `TuneD` application provides profiles in the following categories:

- Power-saving profiles
- Performance-boosting profiles

The performance-boosting profiles might focus on the following aspects:

- Low latency for storage and network

- High throughput for storage and network
- Virtual machine performance
- Virtualization host performance

The following table lists the tuning profiles that are distributed with Red Hat Enterprise Linux 10:

**Table 14. Tuning Profiles That Are Distributed with Red Hat Enterprise Linux 10**

Tuned Profile	Purpose
accelerator-performance	Increased performance-based tuning.
aws	Optimize for Amazon Web Services (AWS) EC2 instances.
balanced	General nonspecialized tuned profile.
balanced-battery	Balanced profile for power saving.
desktop	Optimize for the desktop use-case.
hpc-compute	Optimize for High-Performance Computing (HPC) compute workloads.
intel-sst	Configure for Intel Speed Select Base Frequency.
latency-performance	Optimize for deterministic performance at the cost of increased power consumption.
network-latency	Optimize for deterministic performance at the cost of increased power consumption, focused on low-latency network performance.

Tuned Profile	Purpose
network-throughput	Optimize for streaming network throughput, generally necessary only on earlier CPUs or 40 G+ networks.
optimize-serial-console	Optimize for serial console use.
powersave	Optimize for low power consumption.
throughput-performance	Broadly applicable tuning that provides excellent performance across various common server workloads.
virtual-guest	Optimize for running inside a virtual guest.
virtual-host	Optimize for running Kernel-based Virtual Machine (KVM) guests.

The TuneD application stores its preinstalled profiles, which are referred to as system or factory profiles, in the /usr/lib/tuned/profiles directory. These profiles are not intended for direct editing.

Each profile is in its own directory, which contains the main tuned.conf configuration file and, optionally, other related files.

The following example lists the profiles in the /usr/lib/tuned/profiles directory:

```
root@host:~# ls /usr/lib/tuned/profiles/
accelerator-performance  hpc-compute          optimize-serial-console
aws                      intel-sst            powersave
balanced                 latency-performance   throughput-performance
balanced-battery          network-latency     virtual-guest
desktop                  network-throughput  virtual-host
```

The following example lists the contents of the /usr/lib/tuned/profiles/virtual-guest directory:

```
root@host:~# ls /usr/lib/tuned/profiles/virtual-guest/
tuned.conf
```

The tuned.conf configuration file appears as follows:

```
#  
# tuned configuration  
  
#[main]  
summary=Optimize for running inside a virtual guest  
include=throughput-performance  
  
[vm]  
# If a workload mostly uses anonymous memory and it hits this limit, the entire  
# working set is buffered for I/O, and any more write buffering would require  
# swapping, so it's time to throttle writes until I/O can catch up. Workloads  
# that mostly use file mappings may be able to use even higher values.  
#  
# The generator of dirty data starts writeback at this percentage (system  
default  
# is 20%)  
dirty_ratio = 30  
  
[sysctl]  
# Filesystem I/O is usually much more efficient than swapping, so try to keep  
# swapping low. It's usually safe to go even lower than this on systems with  
# server-grade storage.  
vm.swappiness = 30
```

The [main] section in the file can include a summary of the tuning profile. This section also supports the include directive, which is a feature that enables profile inheritance. The TuneD profile (the *child profile*) can inherit all settings from another profile (its *parent profile*). With this inheritance-based configuration model, when creating tuning profiles, you can use a preinstalled profile as a foundation. You can then add or modify specific parameters to suit your needs.

To create or customize tuning profiles, copy the relevant profile directory from the /usr/lib/tuned/profiles directory to the /etc/tuned/profiles directory, and make your modifications. If profile directories with the same name exist in both locations, then the one in the /etc/tuned/profiles directory takes precedence. Therefore, never directly modify files in the /usr/lib/tuned/profiles system directory.

The remaining sections in the `tuned.conf` file use tuning plug-ins to modify system parameters. For instance, in the previous example, the `[sysctl]` section modifies the `vm.swappiness` kernel parameter through the `sysctl` tuning plug-in. The `vm.swappiness` parameter dictates how aggressively the kernel moves processes from physical memory to swap space.

To check the value of this parameter on your system, use the `sysctl` command:

```
root@host:~# sysctl vm.swappiness
vm.swappiness = 30
```

This parameter accepts values from 0 to 100. A lower value, such as 30, would be set by activating the `virtual-guest` profile to reduce the kernel's tendency to use swap space.

## Managing Profiles from the Command Line

The `tuned-adm` command is your primary tool for managing the TuneD application and its profiles.

The command supports the following tuning activities:

- Query current active settings.
- List all available tuning profiles.
- Receive a recommended tuning profile for your system.
- Directly switch between tuning profiles.
- Deactivate all TuneD settings.

You can identify the currently active tuning profile with the `tuned-adm active` command.

```
root@host:~# tuned-adm active
Current active profile: throughput-performance
```

The `tuned-adm verify` command verifies whether the current system settings match the active profile.

```
root@host:~# tuned-adm verify
Verification succeeded, current system settings match the preset profile.
See TuneD log file ('/var/log/tuned/tuned.log') for details.
```

The `tuned-adm list` command lists all available tuning profiles, including built-in and custom-created profiles.

```
root@host:~# tuned-adm list
Available profiles:
- accelerator-performance      - Throughput performance based tuning with ...
- aws                         - Optimize for aws ec2 instances
- balanced                     - General non-specialized tuned profile
- balanced-battery             - Balanced profile biased towards power savings
...
- desktop                      - Optimize for the desktop use-case
...output omitted...
Current active profile: throughput-performance
```

For information about a specific tuning profile, use the `tuned-adm profile_info` command followed by the profile's name.

```
root@host:~# tuned-adm profile_info network-latency
Profile name:
network-latency

Profile summary:
Optimize for deterministic performance at the cost of increased power
consumption, focused on low latency network performance
...output omitted...
```

If no profile is specified, then the `tuned-adm profile_info` command displays information for the currently active tuning profile.

To switch to a different profile that better matches your system's current tuning requirements, use the `tuned-adm profile` command followed by the name of the profile. This command changes the active tuning profile persistently.

```
root@host:~# tuned-adm profile virtual-guest
```

The `tuned-adm recommend` command suggests a tuning profile based on various system characteristics. These characteristics include factors such as whether the system is a virtual machine, and other predefined categories that are selected during system installation.

The system also uses this mechanism to determine the default profile after its initial installation.

```
root@host:~# tuned-adm recommend  
virtual-guest
```

To revert the settings that the current profile applied, you have two options: either switch to a different profile or deactivate the tuned daemon. To turn off all TuneD activity, use the `tuned-adm off` command.

```
root@host:~# tuned-adm off
```

## Managing Profiles with the Web Console

To manage system performance profiles with the web console, you must log in and escalate your privileges. You can then execute commands that modify system performance profiles, which require administrative access due to their impact on core system parameters.

You can switch to the administrative access mode in the web console by clicking either the **Limited access** or the **Turn on administrative access** button. You are then prompted to enter your password.

As a privileged user, click the **Overview** menu option in the left navigation bar. The Performance profile field displays the current active profile.

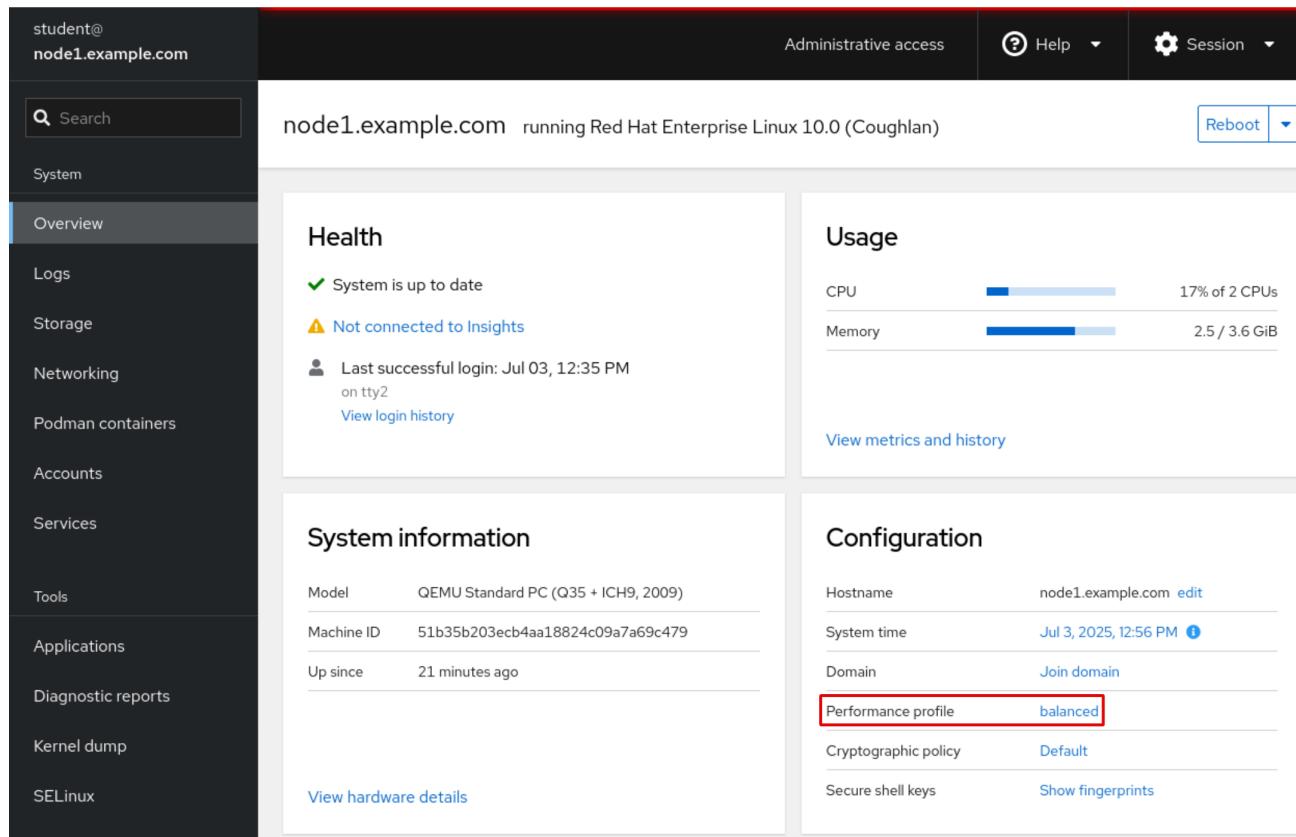


Figure 6. Active performance profile

To select a different profile, click the active profile link. In the **Change performance profile** user interface, scroll through the profile list and select the chosen profile by clicking it. Then, click the **Change profile** button to apply the change.

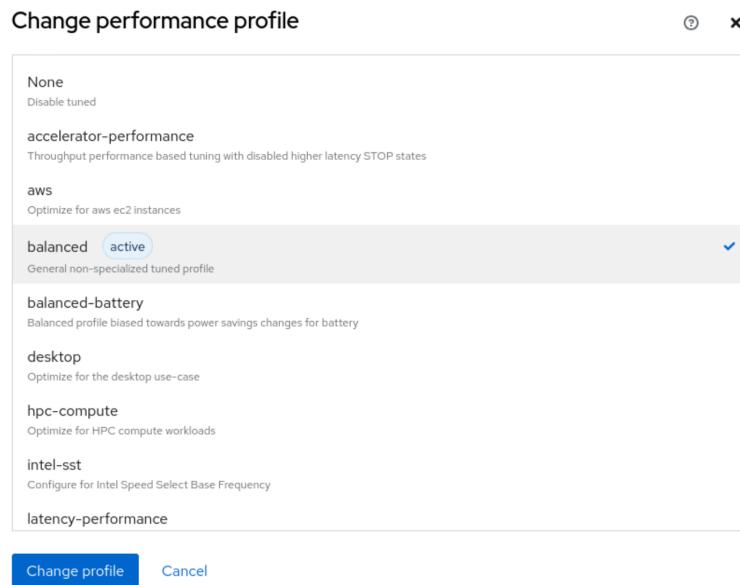


Figure 7. Select a preferred performance profile

To verify changes, return to the main **Overview** page, and confirm that it displays the chosen profile in the Performance profile field.

## Enabling the Web Console

Starting in Red Hat Enterprise Linux 10, the web console is preinstalled. You can install the web console by installing the `cockpit` package:

```
root@host:~# dnf install cockpit
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

Enable and start the `cockpit.socket` unit to access the web console:

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

If you are using a custom firewall configuration, then you must open the 9090 port or add the `cockpit` service to allow incoming connections to the web console.

```
root@host:~# firewall-cmd --add-service=cockpit --permanent  
success
```

Update the firewalld service with the changes:

```
root@host:~# firewall-cmd --reload  
success
```

To access the web console, open the `https://servername:9090` URL and replace `servername` with your server's hostname or IP address.

By default, the cockpit service uses a self-signed TLS certificate to secure connections. When you connect for the first time, your web browser might display a security warning due to a commercial Certificate Authority not trusting this certificate. You can safely ignore this warning.

Log in to the web console by using the user and password on the system.

## References

`tuned(8)`, `tuned.conf(5)`, `tuned-main.conf(5)`, `tuned-profiles(7)`, and `tuned-adm(1)` man pages

For more information, refer to the *Monitoring and Managing System Status and Performance* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/monitoring\\_and\\_managing\\_system\\_status\\_and\\_performance/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/monitoring_and_managing_system_status_and_performance/index)

## 9.2. Guided Exercise

### Set a Tuning Profile

Tune system performance by selecting an appropriate tuning profile.

#### Outcomes

- Persistently switch to a new tuning profile and validate the resulting system settings.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start tuning-profiles
```

#### Instructions

- Log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- Verify that the TuneD package is installed, and is enabled and running.

- Verify that the tuned package is installed.

```
[student@servera ~]$ dnf list tuned
...output omitted...
Installed Packages
tuned.noarch                               2.25.1-1.el10          @System
```

- Verify that the tuned service is enabled.

```
[student@servera ~]$ systemctl is-enabled tuned
enabled
```

- Verify that the tuned service is currently running.

```
[student@servera ~] systemctl is-active tuned  
active
```

3. List the available tuning profiles and identify the active profile.

```
[student@servera ~]$ tuned-adm list  
Available profiles:  
...output omitted...  
- virtual-guest - Optimize for running inside a virtual guest  
- virtual-host - Optimize for running KVM guests  
Current active profile: virtual-guest
```

4. Review the /usr/lib/tuned/profiles/virtual-guest/tuned.conf configuration file for the current active profile. The virtual-guest tuning profile is based on the throughput-performance profile, with different values for the vm.dirty\_ratio and vm.swappiness parameters. Verify that the virtual-guest tuning profile sets the right values for these parameters on your system.

- 4.1. Review the tuned.conf configuration file for the virtual-guest profile. Note the values for the vm.dirty\_ratio and vm.swappiness parameters.

```
[student@servera ~]$ cat /usr/lib/tuned/profiles/virtual-guest/tuned.conf  
...output omitted...  
  
[vm]  
...output omitted...  
dirty_ratio = 30  
  
[sysctl]  
...output omitted...  
vm.swappiness = 30
```

- 4.2. Verify that the current system settings match the active virtual-guest profile by using the tuned-adm command.

```
[student@servera ~]$ tuned-adm verify  
Verification succeeded, current system settings match the preset profile.  
See TuneD log file ('/var/log/tuned/tuned.log') for details.
```

- 4.3. Retrieve the current values that the virtual-guest tuning profile sets on your system for the vm.dirty\_ratio kernel parameter.

```
[student@servera ~]$ sysctl vm.dirty_ratio  
vm.dirty_ratio = 30
```

- 4.4.** Retrieve the current values that the `virtual-guest` tuning profile sets on your system for the `vm.swappiness` kernel parameter.

```
[student@servera ~]$ sysctl vm.swappiness  
vm.swappiness = 30
```

- 5.** Review the `tuned.conf` configuration file for the `latency-performance` tuning profile, and note the different values for the `vm.dirty_ratio` and `vm.swappiness` parameters. Then, persistently switch to the `latency-performance` profile and verify the new values for these two parameters.
- 5.1.** Review the `tuned.conf` configuration file in the `/usr/lib/tuned/profiles/latency-performance` directory. Take note of the values it sets for the `vm.dirty_ratio` and `vm.swappiness` parameters.

```
[student@servera ~]$ cat /usr/lib/tuned/profiles/latency-performance/tuned.conf  
...output omitted...  
[vm]  
...output omitted...  
dirty_ratio=10  
...output omitted...  
[sysctl]  
...output omitted...  
vm.swappiness=10  
...output omitted...
```

- 5.2.** Persistently change the active tuning profile to the `latency-performance` profile. Use `student` as the password.

```
[student@servera ~]$ sudo tuned-adm profile latency-performance
```

- 5.3.** Confirm that `latency-performance` is the active tuning profile.

```
[student@servera ~]$ tuned-adm active  
Current active profile: latency-performance
```

- 5.4.** Verify the value for the `vm.dirty_ratio` parameter.

```
[student@servera ~]$ sysctl vm.dirty_ratio  
vm.dirty_ratio = 10
```

- 5.5.** Verify the value for the `vm.swappiness` parameter.

```
[student@servera ~]$ sysctl vm.swappiness  
vm.swappiness = 10
```

- 6.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish tuning-profiles
```

## 9.3. Influencing Process Scheduling

### Objectives

- Describe command-line tools to prioritize or deprioritize the CPU scheduling priority for specific processes.

### Preemptive Multitasking

Modern computers, from your personal workstation to the largest supercomputers, are built with multi-core, multithreaded CPUs. With this design, CPUs can execute many instruction threads simultaneously, and significantly boosts their processing power. High-performance supercomputers take this capability to a higher level, and process millions of threads in parallel across hundreds or thousands of CPUs.

Despite these advances, CPU saturation remains a significant challenge across all system types. Whether a user overloads a personal desktop with many applications, or an enterprise server faces thousands of requests per second, systems often meet situations where the number of process threads exceeds the immediate capacity of their available CPUs.

To manage these demanding situations, Linux and other operating systems use a technique known as *time-slicing* or *multitasking*. This ability is managed by the process scheduler, which is the part of the kernel that grants CPU time to tasks.

On modern multi-core CPU systems, the scheduler can execute multiple threads simultaneously across different cores or hardware threads. However, the ability to execute threads in parallel was not always the case. Historically, on single-CPU systems, true simultaneous execution of multiple programs was impossible; the operating system instead created an illusion of concurrency. By rapidly cycling through all runnable processes, giving each a short timeslice, the scheduler ensured that all active applications appeared to be making progress. This fundamental concept of time-slicing remains relevant even in modern multi-core environments.

The kernel forcibly interrupting a running process and scheduling another process to run on the CPU is called *preemption*. Preemptive multitasking prevents any single CPU-intensive program from monopolizing the CPU and making the system unresponsive.

### Process Scheduling Policies

Rather than allocating CPU time randomly, the kernel's scheduler instead operates based on sophisticated scheduling policies to ensure that every process receives a share of the available CPU time.

For example, special scheduling policies might apply to real-time tasks that are time-critical and with high priorities.

Examples of real-time policies in the kernel source code include SCHED\_FIFO (first in - first out) and SCHED\_RR (round-robin scheduling). However, most applications or processes that you launch default to the SCHED\_NORMAL policy, which is the standard time-sharing scheduling policy. For POSIX-standard compliance reasons, SCHED\_OTHER is also a valid alias for the SCHED\_NORMAL policy.

Starting in Red Hat Enterprise Linux 10, the algorithm that implements the SCHED\_NORMAL policy is called *Earliest Eligible Virtual Deadline First* (EEVDF), which replaces Completely Fair Scheduler (CFS) from earlier versions of RHEL.

EEVDF builds on fairness concepts from CFS and also introduces the concept of deadlines. Each task is assigned an eligible virtual deadline based on how much CPU time it is owed and its priority. The scheduler then picks the task with the earliest eligible virtual deadline to run next. EEVDF offers better latency handling and responsiveness than CFS due to its integrated deadline calculations.

## Nice Values in Linux

The sophisticated deadline calculations in EEVDF dynamically manage system-wide fairness and responsiveness. However, users can still influence an individual process's relative priority or its share of CPU time through its nice value.

The nice value is a parameter of the default SCHED\_NORMAL policy that a user can set to enable a process to be "nice" (or "less nice") to other processes by adjusting its readiness to yield CPU time. This value ranges from -20 (the least "nice", to indicate highest priority) to +19 (the most "nice", to indicate the lowest priority). By default, most processes start with a nice value of 0.

A process with a lower nice value (for example, -10) demands a proportionally larger share of CPU time. Conversely, a process with a higher nice value (for example, +10) requests a smaller share.

Within the EEVDF scheduler, this nice value directly influences a task's weight, which in turn impacts the calculation of its eligible virtual deadline. A higher weight (from a lower nice value) results in an earlier deadline, which enables the process to run sooner and more often.

### Important

The nice values affect only non-real-time (SCHED\_NORMAL) processes and do not elevate a process's priority above any real-time tasks, for example SCHED\_FIFO or SCHED\_RR tasks. Real-time tasks belong to a higher-priority scheduling class and always have priority over normal processes.

On a system that is not experiencing CPU saturation, the effects of nice values might be less noticeable, because even lower-priority tasks can receive ample CPU time due to lack of contention for resources.

## Permission to Modify Nice Values

Privileged users can decrease a process's nice value, which makes it less "nice" and increases its priority. The EEVDF scheduler consequently schedules the process sooner and more often, and so it acquires a larger share of CPU time.

Conversely, unprivileged users can increase the nice value only for their own processes, which makes them "nicer" and thus lowers their priority. They might then be scheduled less often and receive a smaller share of CPU time. Unprivileged users cannot decrease their processes' nice values to raise their importance, nor can they adjust the nice value for another user's process.

## Viewing Process Priorities

The top and ps commands are often used to inspect the priority of running processes.

The top command provides a dynamic view of running processes and system resource usage by constantly updating its display.

In contrast, the ps command displays a static snapshot of current processes when the command is executed.

### Dynamically Inspecting Processes

The top command displays each process with key metrics, including its priority (PR) and nice (NI) value.

The PR column shows the top command's representation of the kernel's calculated scheduling priority for a process, and provides a unified view of both real-time and normal processes on a single scale.

```
top - 13:27:06 up 3 min, 3 users, load average: 0.45, 0.49, 0.22
Tasks: 250 total, 1 running, 249 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 4.3 sy, 0.0 ni, 95.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 5662.9 total, 4358.2 free, 1041.5 used, 511.4 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 4621.5 avail Mem

      PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
 4201 student    30  10  232212  2912  2400 R 100.0  0.1  8:08.03 sha1sum
 4203 student    39  19  232212  2992  2480 R 90.9   0.1  8:03.72 sha1sum
 4163 student    20  0  232212  2904  2392 R 81.8   0.1  8:37.92 sha1sum
 2189 student    20  0  232016  4944  2896 R 8.3    0.1  0:00.02 top
    1 root       20  0  49516  41284 10312 S 0.0   0.7  0:02.11 systemd
    2 root       20  0      0      0      0 S 0.0   0.0  0:00.00 kthreadd
    3 root       20  0      0      0      0 S 0.0   0.0  0:00.00 pool_wor
...
    4 root       0 -20      0      0      0 I 0.0   0.0  0:00.00
kworker/R ...
...output omitted...
```

For non-real-time (SCHED\_NORMAL) processes, the priority can be calculated by using the formula: PR = 20 + nice\_value.

As seen in the output of the previous command, the first process has a nice value of 10, which maps to a PR value of 30 ( $20 + 10 = 30$ ).

The following diagram shows how normal processes with nice values and real-time processes are mapped to a single PR column in the top command output.

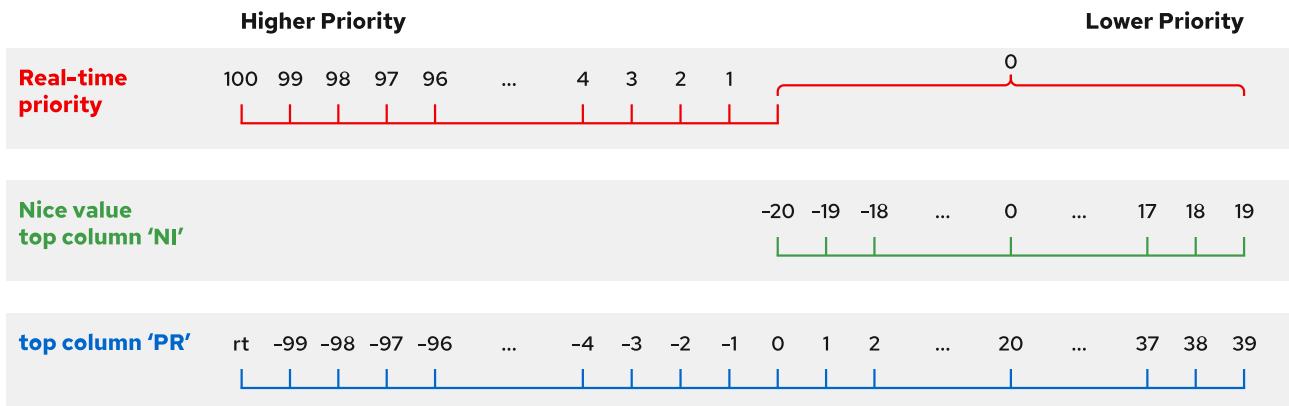


Figure 8. Priorities and nice values as reported by the top command

### Note

For real-time processes, the PR value in the top command is calculated differently. The formula is:  $PR = -1 - \text{real\_time\_priority}$ . The numerical range for real-time priorities is 1 (lowest RT priority) to 99 (highest RT priority). This calculation always results in negative numbers for real-time processes in the PR column. Additionally, the top command might sometimes display the rt string in the PR column, which corresponds to a value of -100 (the highest possible scheduling priority for real-time processes).

## Statically Inspecting Processes

The ps command provides a snapshot of the currently running processes on your system.

The following example demonstrates how to use the ps command to display specific columns for currently running sha1sum processes:

```
user@host:~$ ps -o pid,priority,nice,cls,pcpu,comm -C sha1sum
  PID PRI NI  CLS %CPU COMMAND
 2176  20   0   TS 99.2 sha1sum
 2183  30  10   TS 99.1 sha1sum
 2185 -11   -   RR 99.3 sha1sum
```

In the CLS (scheduling class) column, TS stands for *time sharing*, which indicates that the default SCHED\_NORMAL policy manages the process. Other common CLS values, such as FF (*first in, first out*) and RR (*round robin*) signify real-time processes. Because real-time processes do not use nice values, their NI column displays a dash (-).

The ps command also supports sorting processes by a selected field.

The following example sorts every process on the system by nice value in descending order:

```
user@host:~$ ps -eo pid,priority,nice,cls,pcpu,comm --sort=-nice | head
  PID PRI NI CLS %CPU COMMAND
    60  39  19  TS  0.0 khugepaged
  2185  39  19  TS 99.2 sha1sum
  2183  30  10  TS 99.4 sha1sum
    59  25   5  TS  0.0 ksmd
   895  21   1  TS  0.0 rtkit-daemon
    1  20   0  TS  0.2 systemd
    2  20   0  TS  0.0 kthreadd
    3  20   0  TS  0.0 pool_workqueue_release
    8  20   0  TS  0.0 kworker/0:0-events_freezable
```

## Starting a Process with a Specific Nice Value

By default, when a process is created, it inherits its parent's nice value. When a process starts from the command line, it inherits its nice value from the shell process. Typically, new processes run with the default nice value of 0.

The following example starts a CPU-intensive process from the shell as a background job, and then displays the nice value of the matching process name.

```
user@host:~$ sha1sum /dev/zero &
[1] 2794
user@host:~$ ps -o pid,nice,comm -C sha1sum
  PID NI COMMAND
 2794  0 sha1sum
```

All users can use the nice command to start commands with the default or a higher nice value.

The following example starts the same command with the default nice value of 10, and displays the nice value of matching processes with the same name:

```
user@host:~$ nice sha1sum /dev/zero &
[2] 2810
user@host:~$ ps -o pid,nice,comm -C sha1sum
  PID NI COMMAND
 2794  0 sha1sum
 2810  10 sha1sum
```

Use the nice command -n option to apply a user-defined nice value when starting a process.

The following example starts another background job with a nice value of 15 and displays the result:

```
user@host:~$ nice -n 15 sha1sum /dev/zero &
[3] 2814
user@host:~$ ps -o pid,nice,comm -C sha1sum
  PID  NI COMMAND
 2794   0 sha1sum
 2810  10 sha1sum
 2814  15 sha1sum
```

### Important

When running CPU-intensive processes, particularly for testing, ensure that you terminate them to prevent system slowdown or resource exhaustion:

```
user@host:~$ pkill sha1sum
[1]  Terminated          sha1sum /dev/zero
[2]- Terminated          nice sha1sum /dev/zero
[3]+ Terminated          nice -n 15 sha1sum /dev/zero
```

## Changing the Nice Value of a Running Process

You can change the nice value of an already running process by using the `renice` command.

The following example demonstrates the `renice` command by identifying a process via its process ID and increasing its nice value to 19.

```
user@host:~$ renice -n 19 3033
3033 (process ID) old priority 10, new priority 19
```

A subsequent attempt to lower the nice value back to 10 would fail, because unprivileged users cannot lower nice values:

```
user@host:~$ renice -n 10 3033
renice: failed to set priority for 3033 (process ID): Permission denied
```

As a workaround, the user can terminate and restart the process, to restore its standard priority.

You can alternatively use the `top` command to change the nice value for an existing process. From the `top` interactive interface, press the `R` key to access the `renice` command. Enter the process ID, and then enter the new nice value.

## References

`nice(1)`, `renice(1)`, `top(1)`, `ps(1)`, `sched(7)`, and `sched_setscheduler(2)` man pages

## 9.4. Guided Exercise

### Influence Process Scheduling

Use command-line tools to change the priority of the CPU scheduling for specific processes.

#### Outcomes

- Adjust scheduling priorities for processes.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start tuning-nice
```

##### Important

This exercise uses commands that perform an endless checksum on a device file and that intentionally use significant CPU resources.

#### Instructions

1. Log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Determine the number of available CPU cores, and then start two instances by using the sha1sum /dev/zero & command for each core.

- 2.1. Use the nproc command to determine the number of processors in the system.

```
[student@servera ~]$ nproc
2
```

- 2.2. Use the for loop in the shell to start multiple instances of the sha1sum /dev/zero & command. Start two instances for each virtual processor in the previous step.

The for loop creates four instances. The PID values in your output might vary on your system.

```
[student@servera ~]$ for i in {1..4}; do sha1sum /dev/zero & done
[1] 8531
[2] 8532
[3] 8533
[4] 8534
```

3. Verify that the sha1sum jobs are running in the background.

```
[student@servera ~]$ jobs
[1]  Running                  sha1sum /dev/zero &
[2]  Running                  sha1sum /dev/zero &
[3]- Running                 sha1sum /dev/zero &
[4]+ Running                 sha1sum /dev/zero &
```

4. Use the ps command to display the CPU usage for each sha1sum process.

```
[student@servera ~]$ ps u -C sha1sum
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
student     8531 49.5  0.1 232212  2980 pts/0      R    22:16  3:59 sha1sum
/dev/zero
student     8532 49.4  0.1 232212  2912 pts/0      R    22:16  3:58 sha1sum
/dev/zero
student     8533 49.7  0.1 232212  2908 pts/0      R    22:16  4:00 sha1sum
/dev/zero
student     8534 49.4  0.1 232212  2940 pts/0      R    22:16  3:59 sha1sum
/dev/zero
```

5. Terminate all the sha1sum processes, and then verify that no jobs are running.

- 5.1. Use the pkill command to terminate all running processes that match the sha1sum name.

```
[student@servera ~]$ pkill sha1sum
[1]  Terminated                sha1sum /dev/zero
[2]  Terminated                sha1sum /dev/zero
[3]- Terminated               sha1sum /dev/zero
[4]+ Terminated               sha1sum /dev/zero
```

- 5.2. Verify that no jobs are running.

```
[student@servera ~]$ jobs
```

- 6.** Start multiple instances of the sha1sum /dev/zero & command, and then start one additional instance of the sha1sum /dev/zero & command with a nice value of 12. Start at least as many instances as the number of system virtual processors.

Three regular instances are started, plus another with a higher nice level.

- 6.1.** Use the for loop to start three instances of the sha1sum /dev/zero & command.

```
student@servera ~]$ for i in {1..3}; do sha1sum /dev/zero & done  
[1] 9433  
[2] 9434  
[3] 9435
```

- 6.2.** Use the nice command to start the fourth instance with a nice value of 12. Note the PID of this process for use in a further step. The PID might vary on your system.

```
[student@servera ~]$ nice -n 12 sha1sum /dev/zero &  
[4] 9498
```

- 7.** Use the ps command to display the PID, CPU usage, nice value, and command name for each sha1sum process. The instance with the nice value of 12 displays a lower percentage for CPU usage than the other instances.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm -C sha1sum  
PID %CPU NI COMMAND  
9433 65.3 0 sha1sum  
9434 65.2 0 sha1sum  
9435 64.9 0 sha1sum  
9498 5.4 12 sha1sum
```

- 8.** Use the renice command for the process with the nice value of 12 to change this value from 12 to 5. Use student as the password.

```
[student@servera ~]$ sudo renice -n 5 9498
[sudo] password for student:
9498 (process ID) old priority 12, new priority 5
```

- 9.** Use the ps command to display the CPU usage and nice value of the sha1sum processes. By lowering the nice level, the priority of the process increases, and results in higher CPU usage.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm -C sha1sum
  PID %CPU NI COMMAND
 9433 63.4  0 sha1sum
 9434 63.7  0 sha1sum
 9435 63.1  0 sha1sum
 9498  9.7  5 sha1sum
```

- 10.** Use the pkill command to terminate all running processes that match the sha1sum process name.

```
[student@servera ~]$ pkill sha1sum
...output omitted...
```

- 11.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish tuning-nice
```

## 9.5. Lab

# Tune System Performance

Use a tuning profile to tune system performance and adjust the scheduling priority of specific processes.

### Outcomes

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

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

#### Important

This lab uses commands that perform an endless checksum on a device file and that intentionally use significant CPU resources.

### Instructions

1. Change the current tuning profile for the `serverb` machine to the `balanced` profile, which is a general tuned profile. Then, list the information for the `balanced` tuning profile.
2. Two processes on the `serverb` machine are consuming a high percentage of CPU time. Adjust the `nice` value of each of the two processes to 10 to allow more CPU time for other processes.
3. Return to the workstation machine as the student user.

### Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade tuning-review
```

### Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish tuning-review
```

## Solution

Use a tuning profile to tune system performance and adjust the scheduling priority of specific processes.

## Outcomes

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

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

### Important

This lab uses commands that perform an endless checksum on a device file and that intentionally use significant CPU resources.

## Instructions

1. Change the current tuning profile for the serverb machine to the balanced profile, which is a general tuned profile. Then, list the information for the balanced tuning profile.

- 1.1. Log in to the serverb machine as the student user.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. Verify that the tuned package is installed.

```
[student@serverb ~]$ dnf list tuned
...output omitted...
Installed Packages
tuned.noarch                               2.25.1-1.el10          @System
```

- 1.3. Verify that the tuned service is active.

```
[student@serverb ~]$ systemctl is-active tuned  
active
```

- 1.4.** List the available tuning profiles along with the current active profile.

```
[student@serverb ~]$ tuned-adm list  
Available profiles:  
- accelerator-performance      - Throughput performance based tuning with  
disabled higher latency STOP states  
- aws                         - Optimize for aws ec2 instances  
- balanced                     - General non-specialized tuned profile  
- balanced-battery             - Balanced profile biased towards power savings  
changes for battery  
- desktop                      - Optimize for the desktop use-case  
- hpc-compute                  - Optimize for HPC compute workloads  
- intel-sst                     - Configure for Intel Speed Select Base Frequency  
- latency-performance          - Optimize for deterministic performance at the  
cost of increased power consumption  
- network-latency               - Optimize for deterministic performance at the  
cost of increased power consumption, focused on low latency network performance  
- network-throughput            - Optimize for streaming network throughput,  
generally only necessary on older CPUs or 40G+ networks  
- optimize-serial-console       - Optimize for serial console use.  
- powersave                     - Optimize for low power consumption  
- throughput-performance        - Broadly applicable tuning that provides  
excellent performance across a variety of common server workloads  
- virtual-guest                 - Optimize for running inside a virtual guest  
- virtual-host                  - Optimize for running KVM guests  
Current active profile: virtual-guest
```

- 1.5.** Change the current active tuning profile to the balanced profile. Use student as the password.

```
[student@serverb ~]$ sudo tuned-adm profile balanced
```

- 1.6.** List summary information of the current active tuned profile. Verify that the active profile is the balanced profile.

```
[student@serverb ~]$ sudo tuned-adm profile_info  
Profile name:  
balanced  
...output omitted...
```

- 2.** Two processes on the serverb machine are consuming a high percentage of CPU time. Adjust the

nice value of each of the two processes to 10 to allow more CPU time for other processes.

- 2.1.** Determine the top two CPU consumers on the serverb machine. Use the ps command to sort its output by CPU usage, and arrange the list so the most CPU-intensive processes appear at the bottom. The CPU usage might vary on your machine.

```
[student@serverb ~]$ ps aux --sort=pcpu
USER     PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
...output omitted...
root     1079 98.5  0.1 225340  2300 ?        RN      06:25   4:29 sha1sum /dev/zero
root     1095 99.0  0.1 225340  2232 ?        R<     06:25   4:30 md5sum /dev/zero
```

- 2.2.** Identify the current nice value for each of the top two CPU consumers. Note the PID values of the two processes for use in further steps. The PID values might vary on your machine.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum;pgrep md5sum)
PID %CPU NI COMMAND
1079 98.8  2 sha1sum
1095 99.1  -2 md5sum
```

- 2.3.** Adjust the nice value for the two processes to 10.

```
[student@serverb ~]$ sudo renice -n 10 1079 1095
[sudo] password for student: student
1079 (process ID) old priority 2, new priority 10
1095 (process ID) old priority -2, new priority 10
```

- 2.4.** Verify that the updated nice value is 10 for both processes.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum;pgrep md5sum)
PID %CPU NI COMMAND
1079 98.9  10 sha1sum
1095 99.2  10 md5sum
```

- 3.** Return to the workstation machine as the student user.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade tuning-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish tuning-review
```

## 9.6. Summary

---

In Red Hat Enterprise Linux, TuneD is a powerful tool that simplifies performance and optimizes power efficiency through predefined tuning profiles.

Users can manage TuneD profiles by using the `tuned-adm` command-line tool or the graphical web console.

Users can influence the scheduling priorities of individual processes through the `nice` values. The `nice` value, which ranges from -20 (least "nice", highest priority) to +19 (most "nice", lowest priority), enables users to adjust a process's readiness to yield CPU time.

The `top` and `ps` commands inspect the priority of running processes and their `nice` values.

The `nice` command starts a process with a specific `nice` value.

The `renice` command changes the `nice` value of an already running process.

Only privileged users can decrease the `nice` value of a process to increase its priority.

## Chapter 10.

### Managing Basic Storage

---

#### Goal

Manage storage devices by creating partitions, file systems, and swap spaces from the command line.

#### Sections

- Creating and Managing File Systems on Standard Partitions (and Guided Exercise)
- Managing Swap Space (and Guided Exercise)

#### Lab

- Manage Basic Storage

## 10.1. Creating and Managing File Systems on Standard Partitions

### Objectives

- Manage storage by creating storage partitions, formatting them with file systems, and mounting the file systems for use.

### Disk Partitioning

*Disk partitioning* divides a hard drive into multiple logical storage partitions. You can use partitions to divide storage based on different requirements, and this division provides many benefits:

- Limit available space to applications or users.
- Separate operating system and program files from user files.
- Create a separate area for memory swapping.
- Limit disk space use to improve the performance of diagnostic tools and backup imaging.

On x86 systems, the two primary partitioning schemes are *Master Boot Record (MBR)* and *GUID Partition Table (GPT)*.

### Master Boot Record (MBR) Partition Scheme

The *Master Boot Record (MBR)* partitioning scheme is the standard on systems that run Basic Input/Output System (BIOS) firmware. This scheme supports a maximum of four primary partitions. On Linux systems, with extended and logical partitions, you can create up to 15 partitions. With a 32-bit partition size, disks that are partitioned with MBR can have a size of up to 2 TiB.



Figure 9. MBR partitioning of the /dev/sdb storage device

The 2 TiB disk and partition size limit is now a common and restrictive limitation. Consequently, the legacy MBR scheme is superseded by the GUID Partition Table (GPT) partitioning scheme.

### GUID Partition Table (GPT) Partition Scheme

For systems that run Unified Extensible Firmware Interface (UEFI) firmware, the *GUID Partition Scheme (GPT)* is the standard for disk partitioning, and addresses the limitations of the MBR scheme.

The GPT scheme provides a maximum of 128 partitions. The GPT scheme allocates 64 bits for logical block addresses to support partitions and disks of up to eight zebibytes (ZiB) or eight billion tebibytes (TiB).



Figure 10. GPT partitioning of the /dev/sdb storage device

The GPT scheme provides additional features and benefits compared to the MBR scheme. The GPT scheme uses a Globally Unique Identifier (GUID) to identify each disk and partition. This scheme makes the partition table redundant, with the primary partition at the head of the disk, and a backup secondary partition at the end of the disk. This scheme uses a checksum to detect errors in the GPT header and partition table.

## Managing Partitions

An administrator can use a *partition editor* program to change a disk's partitions, such as by creating and deleting partitions, and changing partition types.

Red Hat Enterprise Linux provides the `parted` command as the standard command-line partition editor. You can use the `parted` partition editor with storage that uses either the MBR partitioning scheme or the GPT partitioning scheme.

The `parted` command takes as its first argument the device name that represents the entire storage device or disk to modify, followed by subcommands.

The following example uses the `print` subcommand to display the partition table on the disk that is the `/dev/sda` block device (the first QEMU disk that the system detects):

```
root@host:~# parted /dev/sda print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  10.7GB  10.7GB  primary    xfs          boot
 2       10.7GB  53.7GB  42.9GB  primary    xfs
```

To start an interactive session, execute the `parted` command as the `root` user and specify the disk device name as an argument:

```
root@host:~# parted /dev/sda
GNU Parted 3.6
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
```

Use the `print` subcommand to display the partition table:

```
(parted) print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  10.7GB  10.7GB  primary    xfs          boot
 2       10.7GB  53.7GB  42.9GB  primary    xfs
```

To exit the interactive partitioning session, use the `quit` command:

```
(parted) quit
root@host:~#
```

By default, the `parted` command displays sizes in powers of 10 (KB, MB, GB). You can change the unit size with the `unit` parameter, which accepts the following values:

- `s` for sector

- B for byte
- MiB, GiB, or TiB (powers of 2)
- MB, GB, or TB (powers of 10)

The following example shows the partition table for the /dev/sda device in sectors:

```
root@host:~# parted /dev/sda unit s print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sda: 104857600s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start      End          Size       Type      File system  Flags
 1      2048s    20971486s   20969439s  primary   xfs          boot
 2      20971520s 104857535s  83886016s  primary   xfs
```

You can specify multiple subcommands on the same line, such as the `unit` and `print` subcommands in the previous example.

## Write the Partition Table on a New Disk

To partition a new drive, first write a disk label. The disk label indicates which partitioning scheme to use.

Use the `parted` command to write an MBR disk label:

```
root@host:~# parted /dev/sdb mklabel msdos
```

Use the `parted` command to write a GPT disk label:

```
root@host:~# parted /dev/sdb mklabel gpt
```

### ⚠ Warning

The `mklabel` subcommand wipes the existing partition table. Use the `mklabel` subcommand to reuse the disk without regard to the existing data. If a new label moves the partition boundaries, then all data in existing file systems becomes inaccessible.

## Create MBR Partitions

The following instructions create an MBR disk partition. Specify the disk device to create the partition on.

To start an interactive session, run the `parted` command as the `root` user and specify the disk device name as an argument:

```
root@host:~# parted /dev/sdb
GNU Parted 3.6
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Use the `mkpart` subcommand to create a partition. The `help mkpart` subcommand displays the syntax and available options for each argument:

```
(parted) help mkpart
mkpart PART-TYPE [FS-TYPE] START END      make a partition

      PART-TYPE is one of: primary, logical, extended
      FS-TYPE is one of: udf, btrfs, nilfs2, ext4, ext3, ext2, f2fs, fat32,
fat16,
      hfsx, hfs+, hfs, jfs, swsusp, linux-swap(v1), linux-swap(v0), ntfs,
reiserfs,
      hp-ufs, sun-ufs, xfs, apfs2, apfs1, afs, amufs5, amufs4, amufs3,
amufs2,
      amufs1, amufs0, amufs, affs7, affs6, affs5, affs4, affs3, affs2, affs1,
affs0,
      linux-swap, linux-swap(new), linux-swap(old)
      START and END are disk locations, such as 4GB or 10%. Negative values
count
      from the end of the disk. For example, -1s specifies exactly the last
sector.

      'mkpart' makes a partition without creating a new file system on the
partition.
      FS-TYPE may be specified to set an appropriate partition ID.
```

Create a primary or extended partition.

```
(parted) mkpart
Partition type? primary/extended? primary
```

### Note

If you require more than four partitions on an MBR-partitioned disk, then create three primary partitions and one extended partition. The extended partition serves as a container within which you can create multiple logical partitions.

Next, you are prompted to enter the label for the file-system type. To list the supported file-system types, press **Tab** twice:

```
File system type? [ext2]? Tab Tab
affs0  affs4  amufs  amufs3  apfs2  ext3  fat32  hp-ufs  linux-swap(old)  ntfs
udf
affs1  affs5  amufs0  amufs4  asfs  ext4  hfs  jfs  linux-swap(v0)  reiserfs
xfs
affs2  affs6  amufs1  amufs5  btrfs  f2fs  hfs+  linux-swap  linux-swap(v1)
sun-ufs
affs3  affs7  amufs2  apfs1  ext2  fat16  hfsx  linux-swap(new)  nilfs2
swsusp
```

Enter the file-system type to use on the partition, such as `xfs` or `ext4`. This value is only used as a partition type label, and does not create the file system.

```
File system type? [ext2]? xfs
```

Specify the disk sector where the partition starts.

```
Start? 2048s
```

The `s` suffix provides the value in sectors, or you can use the MiB, GiB, TiB, MB, GB, or TB suffixes. The `parted` command defaults to the MB suffix. The `parted` command rounds provided values to satisfy disk constraints.

When the `parted` command starts, it retrieves the disk topology from the device, such as the disk's physical block size. The `parted` command ensures that the start position that you provide correctly aligns the partition with the disk structure, to optimize performance. If the start position results in a

misaligned partition, then the parted command displays a warning. With most disks, a start sector that is a multiple of 2048 is safe.

Specify the disk sector where the partition ends. You can specify the end as a size or as an ending location. When you provide the end position, the parted command updates the partition table on the disk with the new partition details.

```
End? 1000MB
```

Exit the parted command:

```
(parted) quit
Information: You may need to update /etc/fstab.

root@host:~#
```

After creating the partition, run the udevadm command for the system to detect the new partition and to create the associated device file in the /dev directory.

```
root@host:~# udevadm settle
```

As an alternative to using interactive mode, you can create a partition in a single, noninteractive command:

```
root@host:~# parted /dev/sdb mkpart primary xfs 2048s 1000MB
```

## Create GPT Partitions

The GPT scheme also uses the parted command to create partitions. Specify the disk device to create the partition on.

To start an interactive session, execute the parted command as the root user and specify the disk device name as an argument:

```
root@host:~# parted /dev/sdb
GNU Parted 3.6
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Use the `mkpart` subcommand to begin creating the partition. With the GPT scheme, each partition is given a name.

```
(parted) mkpart
Partition name? []? userdata
```

Enter the file-system type to use on the partition, such as `xfs` or `ext4`. This value is only used as a partition type label, and does not create the file system.

```
File system type? [ext2]? xfs
```

Specify the disk sector where the partition starts:

```
Start? 2048s
```

Specify the disk sector where the partition ends. When you provide the end position, the `parted` command updates the GPT partition on the disk with the new partition details.

```
End? 1000MB
```

Exit the `parted` command.

```
(parted) quit
Information: You may need to update /etc/fstab.

root@host:~#
```

Run the `udevadm settle` command. This command waits for the system to detect the new partition and to create the associated device file in the `/dev` directory.

```
root@host:~# udevadm settle
```

As an alternative to using interactive mode, you can create a partition in a single command:

```
root@host:~# parted /dev/sdb mkpart userdata xfs 2048s 1000MB
```

## Deleting Partitions

The following instructions apply for both the MBR and GPT partitioning schemes. Specify the disk that contains the partition to remove.

Run the `parted` command with the disk device as the only argument:

```
root@host:~# parted /dev/sdb
GNU Parted 3.6
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Identify the partition number of the partition to delete:

```
(parted) print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size   File system  Name      Flags
 1       1049kB  1000MB  999MB  xfs          userdata
```

Delete the partition. The `rm` subcommand immediately deletes the partition from the partition table on the disk.

```
(parted) rm 1
```

Exit the `parted` command:

```
(parted) quit
Information: You may need to update /etc/fstab.

root@host:~#
```

As an alternative to using interactive mode, you can delete a partition in a single command:

```
root@host:~# parted /dev/sdb rm 1
```

## Creating File Systems

After the partitioning tool creates the partition as a new block device, the next step is to add a file system to it. Red Hat Enterprise Linux supports multiple file-system types, and XFS is the recommended default.

As the root user, use the `mkfs.xfs` command to format the partition with the XFS file system:

```
root@host:~# mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=60992 blks
                                =          sectsz=512   attr=2, projid32bit=1
                                =          crc=1      finobt=1, sparse=1, rmapbt=1
                                =          reflink=1 bigtime=1 inobtcount=1 nrext64=1
                                =          exchange=0
data     =          bsize=4096   blocks=243968, imaxpct=25
        =          sunit=0     swidth=0 blks
naming   =version 2             bsize=4096   ascii-ci=0, ftype=1, parent=0
log      =internal log          bsize=4096   blocks=1566, version=2
        =          sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                 extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

To add an ext4 file system, use the `mkfs.ext4` command.

## Mounting File Systems

After you add the file system, the last step is to mount the file system to a directory in the directory structure. When you mount a file system to the directory hierarchy, user-space utilities can access or write files on the device.

### Manually Mount File Systems

Use the `mount` command to manually attach a device to a *mount point* directory location.

The `mount` command requires a device and a mount point, and can include file-system mount options. File-system options customize the behavior of the file system.

```
root@host:~# mount /dev/sdb1 /mnt
```

You also use the `mount` command to view currently mounted file systems, the mount points, and their options. To display the output of the `mount` command, do not provide a trailing slash with the partition.

```
root@host:~# mount | grep sdb1
/dev/sdb1 on /mnt type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

## Persistently Mount File Systems

Manually mounting a file system is a good way to verify that a formatted device is accessible and is working as expected. However, when the server reboots, the system does not automatically mount the file system again.

To configure the system to automatically mount the file system during system boot, add an entry to the `/etc/fstab` file. This configuration file lists the file systems to mount at system boot.

The `/etc/fstab` file is a whitespace-delimited file with six fields per line.

```
root@host:~# cat /etc/fstab
UUID=a8063676-44dd-409a-b584-68be2c9f5570   /          xfs    defaults  0 0
UUID=7a20315d-ed8b-4e75-a5b6-24ff9e1f9838   /dbdata    xfs    defaults  0 0
```

The first field specifies the device. This example uses a UUID to specify the device. File systems create and store the UUID in the partition super block at creation time. Alternatively, you could use the device file, such as `/dev/sdb1`.

The second field is the mount point, which is the specified directory where the block device is accessible. The mount point must exist; if not, then create it with the `mkdir` command.

The third field displays the file-system type, such as `xfs` or `ext4`.

The fourth field is the comma-separated list of options to apply to the device. The `defaults` option is a set of commonly used options. The `mount(8)` man page documents the other available options.

The fifth field is used by the `dump` command to back up the device. Other backup applications do not usually use this field.

The last field, the `fsck` order field, determines whether to run the `fsck` command at system boot to verify that the file systems are clean. The value in this field indicates the order in which the `fsck` command must run. For XFS file systems, set this field to 0, because XFS does not use the `fsck` command to verify its file-system status. For ext4 file systems, set it to 1 for the root file system, and to 2 for the other ext4 file systems. By using this notation, the `fsck` command processes the root file

system first, and then verifies file systems on separate disks concurrently, and file systems on the same disk in sequence.

 **Note**

An incorrect entry in the /etc/fstab file might render the machine nonbootable. Verify that an entry is valid by manually unmounting the new file system and then by using the mount command to read the /etc/fstab file, and remount the file system with that entry's mount options. If the mount command returns an error, then correct it before rebooting the machine.

Alternatively, use the `findmnt` command `--verify` option to parse the /etc/fstab file for partition usability.

When you add or remove an entry in the /etc/fstab file, run the `systemctl daemon-reload` command, or reboot the server, to ensure that the `systemd` daemon loads and uses the new configuration.

```
root@host:~# systemctl daemon-reload
```

Red Hat recommends the use of UUIDs to persistently mount file systems. Block device names can change in certain scenarios, such as if a cloud provider changes the underlying storage layer of a virtual machine, or if disks are detected in a different order on a system boot. The block device file name might change, but the UUID remains constant in the file system's super block.

Use the `lsblk` command `--fs` option to scan the block devices that are connected to a machine and to retrieve the file-system UUIDs.

```
root@host:~# lsblk --fs
  NAME   FSTYPE  FSVER LABEL      UUID                                     FSavail FSuse% MOUNTPOINTS
  sda1
  └─sda
  sda2   xfs     boot    49dd...75fdf   312M    37%   /boot
  └─sda
  sda3   xfs     root    8a90...ce0da   4.8G    48%   /
  sdb
  sdc
  sdd
  sr0   iso9660 Joliet Extension config-2 2025-06-24-19-28-21-00
```

## References

info parted (GNU Parted User Manual)

parted(8), mkfs(8), mount(8), lsblk(8), and fstab(5) man pages

For more information, refer to the *Managing File Systems* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/managing\\_file\\_systems/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/managing_file_systems/index)

## 10.2. Guided Exercise

### Create and Manage File Systems on Standard Partitions

Create storage partitions, format them with file systems, and mount the file systems when the computer boots.

#### Outcomes

- Create a partition by using the MBR partition scheme.
- Format the partition with the XFS file system.
- Persistently mount the file system.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start storage-partitions
```

#### Instructions

1. Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Create the MBR partition scheme on the /dev/sdb device.

```
[root@servera ~]# parted /dev/sdb mklabel msdos
Information: You may need to update /etc/fstab.
```

3. Create a 1 GB primary partition. For correct alignment, start the partition at the 2048 sector. Set the partition file-system type to XFS. Use the parted command in interactive mode to create the partition.

Alternatively, you can perform the same operation by using the `parted /dev/sdb mkpart primary xfs 2048s 1001 MB` noninteractive command.

- 3.1.** Use the `parted` command to create the partition on the `/dev/sdb` disk.

```
[root@servera ~]# parted /dev/sdb
GNU Parted 3.6
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
```

- 3.2.** Use the `mkpart` subcommand to create a primary partition.

```
(parted) mkpart
Partition type? primary/extended? primary
```

- 3.3.** Set the file-system type label to `xfs`.

```
File system type? [ext2]? xfs
```

- 3.4.** Start the partition at the `2048s` sector.

```
Start? 2048s
```

- 3.5.** Because the partition starts at the `2048` sector, set the end position to `1001 MB` to configure a partition size of `1000 MB` (`1 GB`).

```
End? 1001MB
```

- 3.6.** Exit the `parted` command.

```
(parted) quit
Information: You may need to update /etc/fstab.
```

- 4.** Verify your work by listing the partitions on the `/dev/sdb` device.

```
[root@servera ~]# parted /dev/sdb print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  1001MB  1000MB  primary   xfs
```

5. Use the udevadm command to register the new partition.

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

6. Format the new partition with the XFS file system.

```
[root@servera ~]# mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=61056 blks
                                =                      sectsz=512  attr=2, projid32bit=1
                                =                      crc=1     finobt=1, sparse=1, rmapbt=1
                                =                      reflink=1 bigtime=1 inobtcount=1 nnext64=1
data                          bsize=4096   blocks=244224, imaxpct=25
                                =                      sunit=0    swidth=0 blks
naming                         =version 2   bsize=4096  ascii-ci=0, ftype=1, parent=0
log                            =internal log bsize=4096  blocks=16384, version=2
                                =                      sectsz=512  sunit=0 blks, lazy-count=1
realtime                       =none        extsz=4096  blocks=0, rtextents=0
Discarding blocks...Done.
```

7. Configure the new file system to persistently mount to the /archive directory.

- 7.1. Create the /archive directory.

```
[root@servera ~]# mkdir /archive
```

- 7.2. Print the UUID of the /dev/sdb1 device, and note the UUID for future steps. The device UUID might be different on your system.

```
[root@servera ~]# lsblk --fs /dev/sdb
NAME   FSTYPE FSVER LABEL UUID ...
sdb
└─sdb1 xfs            291bb68e-91b9-4522-8566-7cc1d848babd
```

- 7.3.** Add an entry to the /etc/fstab file. Replace the UUID with the one that you retrieved in the previous step.

```
...output omitted...
UUID=291bb68e-91b9-4522-8566-7cc1d848babd /archive xfs defaults 0 0
```

- 7.4.** Update the systemd daemon for the system to register the /etc/fstab file configuration.

```
[root@servera ~]# systemctl daemon-reload
```

- 7.5.** Mount the new file system that you added to the /etc/fstab file.

```
[root@servera ~]# mount /archive
```

- 7.6.** Verify that the new file system is mounted on the /archive directory.

```
[root@servera ~]# mount | grep /archive
/dev/sdb1 on /archive type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- 8.** Reboot the servera machine. Log in to the servera machine as the student user. Verify that the /dev/sdb1 partition is mounted on the /archive directory.

- 8.1.** Reboot the servera machine.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
student@workstation:~$
```

- 8.2.** Wait for the servera machine to reboot, and log in as the student user.

```
student@workstation:~$ ssh student@servera  
...output omitted...  
[student@servera ~]$
```

- 8.3.** Verify that the /dev/sdb1 partition is mounted on the /archive directory. To display the output of the mount command, do not provide a trailing slash with the partition.

```
[student@servera ~]$ mount | grep /archive  
/dev/sdb1 on /archive type xfs  
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- 9.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish storage-partitions
```

## 10.3. Managing Swap Space

### Objectives

- Create and activate a swap space to supplement physical memory.

### Swap Space Concepts

A swap space is an area of a disk under the control of the Linux kernel memory management subsystem. The kernel uses swap space to supplement the system RAM by holding inactive pages in memory. A system's *virtual memory* encompasses the combined system RAM and swap space.

When the memory usage on a system exceeds a defined limit, the kernel searches through RAM to look for idle memory pages that are assigned to processes. The kernel writes the idle pages to the swap area and reassigns the RAM pages to other processes. If a program requires access to a page on disk, then the kernel locates another idle page of memory, writes it to disk, and recalls the required page from the swap area.

Because swap areas are on disk, swap is slow when compared with RAM. Although swap space augments system RAM, do not consider swap space as a sustainable solution for insufficient RAM for your workload.

### Swap Space Calculation

Administrators should size the swap space based on the memory workload on the system. Application vendors sometimes provide recommendations for calculating swap space. The following table provides guidance based on the total physical memory.

Table 15. RAM and Swap Space Recommendations

RAM	Swap space	Swap space if allowing for hibernation
2 GB or less	Twice the RAM	Three times the RAM
Between 2 GB and 8 GB	Same as RAM	Twice the RAM
Between 8 GB and 64 GB	At least 4 GB	1.5 times the RAM

RAM	Swap space	Swap space if allowing for hibernation
More than 64 GB	At least 4 GB	Hibernation is not recommended

The laptop and desktop hibernation function uses the swap space to save the RAM contents before powering off the system. When you turn the system back on, the kernel restores the RAM contents from the swap space and does not require a complete boot. For those systems, the swap space must be greater than the amount of RAM.

For more information about the sizing of swap space, refer to the Knowledgebase article, [What Is the Recommended Swap Size for Red Hat Platforms?](#)

## Creating Swap Space

To create a swap space, perform the following steps:

- Create a partition with a file-system type of `linux-swap`.
- Place a swap signature on the device.

### Create a Swap Partition

Use the `parted` command to create a partition of the appropriate size and set its file-system type to `linux-swap`. In the past, tools determined from the partition's file-system type whether to activate the device; however, that requirement is no longer the case. Even though utilities no longer use the partition's file-system type, administrators can determine the partition's purpose from that type.

The following example creates a 256 MB partition.

Use the `parted` command to open an interactive partitioning session:

```
root@host:~# parted /dev/sdb
GNU Parted 3.6
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
```

Use the `print` subcommand to display the partition table:

```
(parted) print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1       1049kB  1001MB  1000MB          data
```

Use the mkpart subcommand to create a partition with the swap1 name:

```
(parted) mkpart
Partition name? []? swap1
```

Set the linux-swap partition type:

```
File system type? [ext2]? linux-swap
```

Specify the start location for the partition:

```
Start? 1001MB
```

Specify the end location for the partition:

```
End? 1257MB
```

Use the print subcommand to verify the newly created partition:

```
(parted) print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1       1049kB  1001MB  1000MB          data
 2       1001MB  1257MB  256MB   linux-swap(v1)  swap1
```

Exit the parted command:

```
(parted) quit
Information: You may need to update /etc/fstab.

root@host:~#
```

As an alternative to using interactive mode, you can create a partition in a single, noninteractive command:

```
root@host:~# parted /dev/sdb mkpart swap1 linux-swap 1001MB 1257MB
```

After creating the partition, run the udevadm command for the system to detect the new partition and to create the associated device file in the /dev directory:

```
root@host:~# udevadm settle
```

## Format Swap Space

The mkswap command applies a swap signature to the device. Unlike other formatting utilities, the mkswap command writes a single block of data at the beginning of the device, and leaves the rest of the device unformatted so that the kernel can use it to store memory pages.

```
root@host:~# mkswap /dev/sdb2
Setting up swap space version 1, size = 244 MiB (255848448 bytes)
no label, UUID=39e2667a-9458-42fe-9665-c5c854605881
```

## Activating Swap Space

Use the swapon command to activate a formatted swap space.

Use the swapon command with the device as a parameter, or use the swapon command -a option to activate all the listed swap spaces in the /etc/fstab file. The swapon command does not activate the swap space persistently.

```
root@host:~# swapon /dev/sdb2
```

Use the swapon command --show option or the free command to inspect the available swap spaces.

```
root@host:~# free
              total        used        free      shared  buff/cache   available
Mem:      1873036     135044     1536040      16748      201952     1575680
Swap:     249852          0      249852
```

## Deactivate Swap Space

You can deactivate a swap space by using the swapoff command. If pages are written to the swap space, then the swapoff command tries to move those pages to other active swap spaces or back into memory. If the swapoff command cannot write data to other places, then it fails with an error, and the swap space stays active.

## Activate Swap Space Persistently

Create an entry in the /etc/fstab file to ensure that a swap space remains active at system boot.

The following example shows a typical line in the /etc/fstab file based on the previously created swap space.

```
UUID=39e2667a-9458-42fe-9665-c5c854605881    swap    swap    defaults    0 0
```

The example uses the partition UUID as the first field. When you format the device, the mkswap command displays the partition UUID. You can also use the lsblk --fs command to display the partition UUID. Alternatively, you can use the device name instead of the partition UUID.

The second field is typically reserved for the mount point. However, for swap devices, which are not accessible through the directory structure, this field takes the swap placeholder value. The fstab(5) man page uses the none placeholder value; however, a swap value provides more informative error messages if something goes wrong.

The third field is the file-system type. The file-system type for swap space is swap.

The fourth field is for options. The example uses the defaults option. The defaults option includes the auto mount option, which activates the swap space automatically at system boot.

The final two fields are the dump flag and the fsck order. Swap spaces do not require backing up or file-system checking, and so these fields should be set to zero.

When you add or remove an entry in the /etc/fstab file, run the systemctl daemon-reload command, or reboot the server, for the systemd daemon to register the new configuration.

```
root@host:~# systemctl daemon-reload
```

## Set Swap Space Priority

By default, the system uses swap spaces in priority order. The kernel uses the highest priority swap space until it is full, and then starts to use the low priority swap space.

The default priority for swap spaces is low, and new swap spaces are lower priority than previously created swap spaces. When swap spaces have the same priority, the kernel writes to them in a round-robin fashion.

To set the priority, use the `pri` option in the `/etc/fstab` file. The kernel uses the swap space with the highest priority first. The default priority is -2.

The following example shows three defined swap spaces in the `/etc/fstab` file. The kernel uses the last entry first, because its priority is set to 10. When that space is full, it uses the second entry, because its priority is set to 4. Finally, it uses the first entry, which has a default priority of -2.

```
UUID=af30cbb0-3866-466a-825a-58889a49ef33    swap      swap      defaults  0 0
UUID=39e2667a-9458-42fe-9665-c5c854605881    swap      swap      pri=4      0 0
UUID=fb7fa60-b781-44a8-961b-37ac3ef572bf    swap      swap      pri=10     0 0
```

Use the `swapon` command `--show` option to display the swap space priorities.

### References

`mkswap(8)`, `swapon(8)`, `swapoff(8)`, `mount(8)`, and `parted(8)` man pages

For more information, refer to the *Getting Started with Swap* chapter in the *Managing Storage Devices* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/managing\\_storage\\_devices/index#getting-started-with-swap](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/managing_storage_devices/index#getting-started-with-swap)

Knowledgebase: What Is the Recommended Swap Size for Red Hat Platforms?

## 10.4. Guided Exercise

### Manage Swap Space

Create and activate a swap space to supplement physical memory.

#### Outcomes

- Create a swap partition.
- Activate the swap space and make it available persistently.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start storage-swap
```

#### Instructions

1. Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Inspect the /dev/sdb disk. The disk already has a partition table and uses the GPT partitioning scheme. Also, the disk has an existing 1 GB partition.

```
[root@servera ~]# parted /dev/sdb print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB          data
```

- 3.** Create a partition named myswap with the size of 500 MB to use as a swap space. Set the partition to the linux-swap partition type.

- 3.1.** Use the noninteractive parted command to create the myswap partition. Because the disk uses the GPT partitioning scheme, you must name the partition.

Specify the start position as 1001 MB, which is the end of the existing partition. The parted command ensures that the new partition immediately follows the previous one, without any gap. Because the myswap partition starts at the 1001 MB position, set the end position to 1501 MB to configure a partition size of 500 MB.

```
[root@servera ~]# parted /dev/sdb mkpart myswap linux-swap 1001MB 1501MB
Information: You may need to update /etc/fstab.
```

- 3.2.** Print the partitions on the /dev/sdb disk. The size of the new partition is not exactly 500 MB. The difference in size is because the parted command must align the partition with the disk layout.

```
[root@servera ~]# parted /dev/sdb print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB          data
 2      1001MB  1501MB  499MB           myswap  swap
```

- 3.3.** Use the udevadm settle command for the system to register the new partition.

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

- 4.** Initialize the new partition as a swap space. The partition UUID might be different on your system.

```
[root@servera ~]# mkswap /dev/sdb2
Setting up swap space version 1, size = 476 MiB (499118080 bytes)
no label, UUID=968e395a-8d1a-4ff0-95aa-71e6671e0ecf
```

- 5.** Verify that you can activate and deactivate the myswap partition.

- 5.1.** Verify that creating and initializing the swap space does not yet enable it for use.

```
[root@servera ~]# swapon --show
[root@servera ~]#
```

- 5.2.** Activate the new swap space.

```
[root@servera ~]# swapon /dev/sdb2
```

- 5.3.** Verify that the new swap space is now available.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/sdb2  partition 476M   0B   -2
```

- 5.4.** Deactivate the swap space.

```
[root@servera ~]# swapoff /dev/sdb2
```

- 5.5.** Confirm that the swap space is deactivated.

```
[root@servera ~]# swapon --show
[root@servera ~]#
```

- 6.** Enable the new swap space at system boot.

- 6.1.** Use the `lsblk` command `--fs` option to retrieve the partition UUID of the `/dev/sdb2` device. Note the partition UUID to use in future steps. The partition UUID might be different on your system.

```
[root@servera ~]# lsblk --fs /dev/sdb2
NAME FSTYPE FSVER LABEL UUID ...
sdb2 swap 1 968e395a-8d1a-4ff0-95aa-71e6671e0ecf
```

- 6.2.** Add an entry to the /etc/fstab file. In the following command, replace the partition UUID with the one that you retrieved in the previous step.

```
...output omitted...
UUID=968e395a-8d1a-4ff0-95aa-71e6671e0ecf swap swap defaults 0 0
```

- 6.3.** Update the systemd daemon for the system to register the /etc/fstab file configuration.

```
[root@servera ~]# systemctl daemon-reload
```

- 6.4.** Enable the swap space by using the entry in the /etc/fstab file.

```
[root@servera ~]# swapon -a
```

- 6.5.** Verify that the new swap space is enabled.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/sdb2  partition 476M   0B   -2
```

- 7.** Reboot the servera machine. After the servera machine reboots, log in to the machine as the student user, and verify that the swap space is enabled.

- 7.1.** Reboot the servera machine.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
student@workstation:~$
```

- 7.2.** Wait for the servera machine to reboot and log in as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 7.3.** Verify that the swap space is enabled.

```
[student@servera ~]$ swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/sdb2  partition 476M   0B   -2
```

8. Return to the workstation machine as the student user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish storage-swap
```

## 10.5. Lab

### Manage Basic Storage

---

Create partitions, format partitions with a file system or a swap space, and mount the file system and activate the swap space when the system boots.

#### Outcomes

- Display and create partitions.
- Format a partition with an XFS file system, and persistently mount the file system.
- Create swap spaces and activate them persistently.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start storage-review
```

#### Instructions

1. The serverb machine has several unused disks. Use the `parted` command to create a GPT partition named `backup` with the size of 2 GB on the `/dev/sdb` disk.

Because the `parted` command requires the use of offsets, a size between 1.8 GB and 2.2 GB is acceptable.

Label the backup partition with the XFS file-system type.

2. Format the backup partition with an XFS file system, and persistently mount the file system on the `/backup` directory by using the partition UUID.

3. On the `/dev/sdb` disk, create two partitions named `swap1` and `swap2`, both with the size of 512 MB. For both partitions, configure the file-system type to host swap spaces.

A size between 460 MB and 564 MB is acceptable.

4. Initialize the `swap1` and `swap2` swap spaces, and configure them to activate at boot. Set the priority order so that the `swap2` partition is preferred over the `swap1` partition.

- For use by Wong Chia Cheong jason.wong76 jason.wong@trainocate.com Copyright © 2025 Red Hat, Inc. | For use by Wong Chia Cheong jason.wong76 jason.wong@trainocate.com Copyright © 2025 Red Hat, Inc.
- 5. Reboot the server machine. Confirm that the system automatically mounts the backup partition to the /backup directory. Additionally, confirm that the system activates the swap1 and swap2 swap spaces.
  - 6. After completing the tasks of this activity, return to the workstation machine as the student user.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade storage-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish storage-review
```

## Solution

---

Create partitions, format partitions with a file system or a swap space, and mount the file system and activate the swap space when the system boots.

## Outcomes

- Display and create partitions.
- Format a partition with an XFS file system, and persistently mount the file system.
- Create swap spaces and activate them persistently.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start storage-review
```

## Instructions

1. The serverb machine has several unused disks. Use the parted command to create a GPT partition named backup with the size of 2 GB on the /dev/sdb disk.

Because the parted command requires the use of offsets, a size between 1.8 GB and 2.2 GB is acceptable.

Label the backup partition with the XFS file-system type.

- 1.1. Log in to the serverb machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 1.2. Identify the unused disks. The /dev/sdb device does not have any partitions.

```
[root@serverb ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda     8:0    0   10G  0 disk
└─sda1  8:1    0    1M  0 part
└─sda2  8:2    0  200M  0 part /boot/efi
└─sda3  8:3    0  9.8G  0 part /
sdb   8:16   0    5G  0 disk
sdc     8:32   0    5G  0 disk
sdd     8:48   0    5G  0 disk
sr0    11:0   1  514K  0 rom
```

**1.3.** Confirm that the /dev/sdb disk has no label.

```
[root@serverb ~]# parted /dev/sdb print
Error: /dev/sdb: unrecognised disk label
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

**1.4.** Define the GPT partitioning scheme.

```
[root@serverb ~]# parted /dev/sdb mklabel gpt
Information: You may need to update /etc/fstab.
```

**1.5.** Create the backup partition with the size of 2 GB, and assign the xfs file-system type label.

Start the partition at the 2048 sector.

```
[root@serverb ~]# parted /dev/sdb mkpart backup xfs 2048s 2GB
Information: You may need to update /etc/fstab.
```

**1.6.** Confirm the creation of the backup partition.

```
[root@serverb ~]# parted /dev/sdb print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  2000MB  1999MB          backup
```

- 1.7.** Use the `udevadm settle` command for the system to register the new partition.

```
[root@serverb ~]# udevadm settle
```

- 2.** Format the backup partition with an XFS file system, and persistently mount the file system on the `/backup` directory by using the partition UUID.

- 2.1.** Format the `/dev/sdb1` partition with an XFS file system.

```
[root@serverb ~]# mkfs.xfs /dev/sdb1
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=121984 blks
                                =           sectsz=512  attr=2, projid32bit=1
                                =           crc=1     finobt=1, sparse=1, rmapbt=1
                                =           reflink=1 bigtime=1 inobtcount=1 nrext64=1
                                =           exchange=0
data     =           bsize=4096   blocks=487936, imaxpct=25
        =           sunit=0    swidth=0 blks
naming   =version 2             bsize=4096   ascii-ci=0, ftype=1, parent=0
log      =internal log         bsize=4096   blocks=16384, version=2
        =           sectsz=512  sunit=0 blks, lazy-count=1
realtime =none                 extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

- 2.2.** Create the `/backup` mount point.

```
[root@serverb ~]# mkdir /backup
```

- 2.3.** Before adding the file system to the `/etc/fstab` file, retrieve its partition UUID to use in further steps. The partition UUID on your system might be different.

```
[root@serverb ~]# lsblk --fs /dev/sdb1
NAME FSTYPE FSVER LABEL UUID ...
sdb1 xfs            d85f5387-9791-400a-9a74-f637e7ee68aa
```

- 2.4.** Edit the /etc/fstab file and define the file system. Use the partition UUID that you retrieved in the previous step.

```
...output omitted...
UUID=d85f5387-9791-400a-9a74-f637e7ee68aa    /backup    xfs    defaults    0 0
```

- 2.5.** Update the systemd daemon for the system to register the /etc/fstab file configuration.

```
[root@serverb ~]# systemctl daemon-reload
```

- 2.6.** Manually mount the /backup directory.

```
[root@serverb ~]# mount /backup
```

- 2.7.** Verify that the /backup directory mounts successfully.

```
[root@serverb ~]# mount | grep /backup
/dev/sdb1 on /backup type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- 3.** On the /dev/sdb disk, create two partitions named swap1 and swap2, both with the size of 512 MB. For both partitions, configure the file-system type to host swap spaces.

A size between 460 MB and 564 MB is acceptable.

- 3.1.** Retrieve the end position of the backup partition that you created in a previous step by displaying the current partition table on the /dev/sdb disk.

```
[root@serverb ~]# parted /dev/sdb print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  2000MB  1999MB  xfs        backup
```

- 3.2.** Create the first partition named swap1 with the size of 512 MB. Set the partition type label to linux-swap. Use the end position of the backup partition as the starting point for this partition: 2000 MB. Set the end position to 2512 MB, which is the start location (2000 MB) added to the partition size (512 MB).

```
[root@serverb ~]# parted /dev/sdb mkpart swap1 linux-swap 2000M 2512M
Information: You may need to update /etc/fstab.
```

- 3.3.** Create the second partition named swap2 with the size of 512 MB. Set the partition type label to linux-swap. Use the end position of the swap1 partition as the starting point for this partition: 2512 MB. Set the end position to 3024 MB, which is the start location (2512 MB) added to the partition size (512 MB).

```
[root@serverb ~]# parted /dev/sdb mkpart swap2 linux-swap 2512M 3024M
Information: You may need to update /etc/fstab.
```

- 3.4.** Display the partition table of the /dev/sdb disk.

```
[root@serverb ~]# parted /dev/sdb print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  2000MB  1999MB  xfs        backup
 2      2000MB  2512MB  513MB   swap1      swap
 3      2512MB  3024MB  512MB   swap2      swap
```

- 3.5.** Use the udevadm settle command for the system to register the new partitions.

```
[root@serverb ~]# udevadm settle
```

4. Initialize the swap1 and swap2 swap spaces, and configure them to activate at boot. Set the priority order so that the swap2 partition is preferred over the swap1 partition.
- 4.1. Use the mkswap command to initialize the swap1 partition. Note the partition UUID of the swap space to use in the future steps. You can also use the lsblk command --fs option to retrieve the partition UUID.

```
[root@serverb ~]# mkswap /dev/sdb2
Setting up swapspace version 1, size = 489 MiB (512749568 bytes)
no label, UUID=54a08491-2722-4145-911c-317c6919e53d
```

- 4.2. Use the mkswap command to initialize the swap2 partition. Note the partition UUID of the swap space to use in the future steps. You can also use the lsblk command --fs option to retrieve the partition UUID.

```
[root@serverb ~]# mkswap /dev/sdb3
Setting up swapspace version 1, size = 488 MiB (511700992 bytes)
no label, UUID=353fe8ca-1276-4443-935e-e1ae87039aef
```

- 4.3. Edit the /etc/fstab file and define the new swap spaces. To set the swap space on the swap2 partition to be preferred over the swap1 partition, give the swap2 partition a higher priority by using the pri option. Use the UUIDs that you retrieved in a previous step.

```
...output omitted...
UUID=54a08491-2722-4145-911c-317c6919e53d    swap      swap  pri=10    0  0
UUID=353fe8ca-1276-4443-935e-e1ae87039aef    swap      swap  pri=20    0  0
```

- 4.4. Update the systemd daemon for the system to register the /etc/fstab file configuration.

```
[root@serverb ~]# systemctl daemon-reload
```

- 4.5. Activate the two swap spaces.

```
[root@serverb ~]# swapon -a
```

- 4.6. Verify that the two swap spaces are activated.

```
[root@serverb ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/sdb2  partition 489M   0B   10
/dev/sdb3  partition 488M   0B   20
```

- 5.** Reboot the serverb machine. Confirm that the system automatically mounts the backup partition to the /backup directory. Additionally, confirm that the system activates the swap1 and swap2 swap spaces.

**5.1.** Reboot the serverb machine.

```
[root@serverb ~]# systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
student@workstation:~$
```

- 5.2.** Wait for the serverb machine to boot, and then log in as the student user.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 5.3.** Verify that the system automatically mounts the /dev/sdb1 partition on the /backup directory.

```
[student@serverb ~]$ mount | grep /backup
/dev/sdb1 on /backup type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- 5.4.** Verify that both swap spaces are activated.

```
[student@serverb ~]$ swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/sdb2  partition 489M   0B   10
/dev/sdb3  partition 488M   0B   20
```

- 6.** After completing the tasks of this activity, return to the workstation machine as the student user.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
student@workstation:~$
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade storage-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish storage-review
```

## 10.6. Summary

---

In this lesson, you learned to manage local storage devices by creating partitions, file systems, and swap spaces from the command line.

Partition schemes are logical divisions of block devices that enable administrators to meet specific storage requirements, such as by organizing data, improving performance, and enhancing security. The Master Boot Record (MBR) partitioning scheme is the standard on systems that run BIOS firmware, and the GUID Partition Table (GPT) scheme is used on systems that run UEFI firmware.

The `parted` command views, adds, modifies, and removes partitions on disks. It supports both the MBR and the GPT partitioning schemes.

The XFS file system is the default and recommended file system format for Red Hat Enterprise Linux 10. The `mkfs.xfs` command creates an XFS file system on a disk partition.

The `mount` command temporarily attaches a device to a mount point directory. To mount devices persistently, modify the `/etc/fstab` file, which lists and configures the devices to mount at boot.

The kernel uses swap space to supplement the system's physical memory by holding inactive memory pages on disk. The `mkswap` command initializes swap spaces on disk partitions. The `swapon` command temporarily activates swap spaces. The `swapoff` command deactivates them. To activate the swap space persistently, use the `/etc/fstab` file.

## Chapter 11.

# Managing Storage with Logical Volume Manager

---

## Goal

Use Logical Volume Manager (LVM) to manage logical volumes that can contain file systems and swap spaces.

## Sections

- Creating Logical Volumes (and Guided Exercise)
- Extending a Logical Volume (and Guided Exercise)
- Replacing Physical Volumes and Managing LVM (and Quiz)

## Lab

- Manage Storage with Logical Volume Manager

## 11.1. Creating Logical Volumes

### Objectives

- Describe how Logical Volume Manager works, and use LVM to create a logical volume, format the volume with a file system, and mount the volume.

### Logical Volume Manager Overview

Use the LVM system to create logical storage volumes as a layer on the physical storage. This storage system provides greater flexibility than using physical storage directly. LVM hides the hardware storage configuration from the software, and enables you to resize volumes without stopping applications or unmounting file systems. LVM provides comprehensive command-line tools to manage storage.

#### Physical devices

Logical volumes use physical devices for storing data. These devices might be disk partitions, whole disks, Redundant Array of Independent Disks (RAID) devices, or Storage Area Networks (SAN) disks. You must initialize a device as an LVM physical volume. An LVM physical volume must use the entire physical device.

#### Physical Volumes (PVs)

LVM uses the underlying physical device as the LVM physical volume. LVM tools segment the physical volumes into *physical extents* (PEs) to form small chunks of data that act as the smallest storage block on a PV.

#### Volume Groups (VGs)

A volume group is a storage pool that is made from one or more PVs. It is the functional equivalent of a whole disk in physical storage. A PV must be allocated only to a single VG. LVM sets the PE size automatically, although it is possible to specify it. A VG might consist of unused space and several logical volumes.

#### Logical Volumes (LVs)

Logical volumes are created from free physical extents in a VG, and are provided as the storage device for applications, users, and operating systems. LVs are a collection of *logical extents* (LEs), which map to physical extents. By default, each LE is mapped to one PE. Setting specific LV options changes this mapping; for example, mirroring causes each LE to map to two PEs.

### Logical Volume Manager Workflow

Creating LVM storage requires building structures in a logical workflow.

- Determine the physical devices to become physical volumes, and initialize these devices as LVM physical volumes.
- Create a volume group from multiple physical volumes.
- Create the logical volumes from the available space in the volume group.
- Format the logical volume with a file system and mount it, or activate it as swap space, or pass the raw volume to a database or storage server for advanced structures.

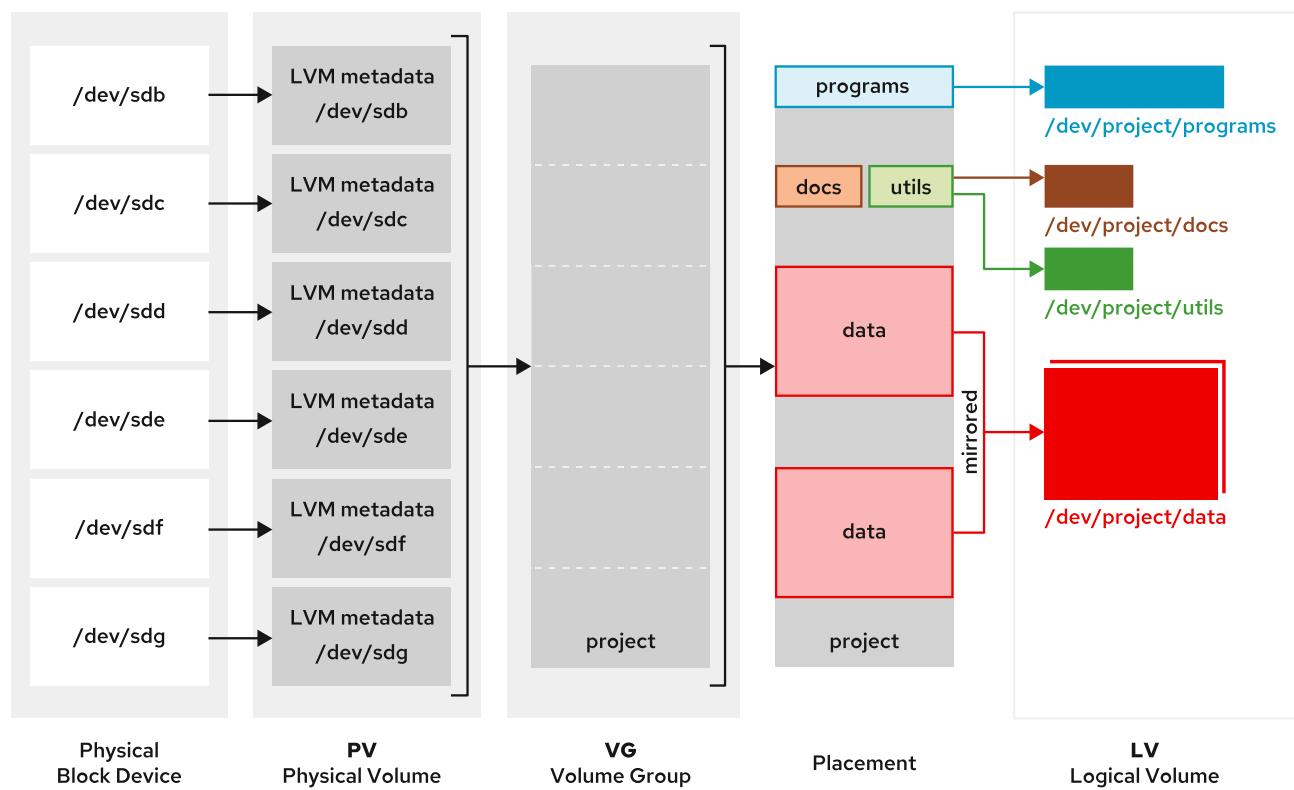


Figure 11. Logical Volume Manager workflow

### Note

The examples here use the /dev/sdb device name and its storage partitions. The device names on your classroom system might be different. Use the `lsblk`, `blkid`, or `cat /proc/partitions` commands to identify your system's devices.

## Build LVM Storage

Creating a logical volume involves creating physical device partitions, physical volumes, and volume groups. After creating an LV, format the volume and mount it to access it as storage.

### Prepare Physical Devices

### Note

Partitioning is optional. You can use the whole raw physical device as a physical volume.

Use the `parted` command to label the partition.

```
root@host:~# parted /dev/sdb mklabel gpt mkpart primary 1MiB 769MiB  
...output omitted...
```

Use the `parted` command to create a partition on the physical device.

```
root@host:~# parted /dev/sdb mkpart secondary 770MiB 1026MiB
```

Set the first partition to the Linux LVM partition type.

```
root@host:~# parted /dev/sdb set 1 lvm on
```

Set the second partition to the Linux LVM partition type.

```
root@host:~# parted /dev/sdb set 2 lvm on
```

Use the `udevadm settle` command to register the new partition with the kernel.

```
root@host:~# udevadm settle
```

## Create Physical Volumes

Use the `pvcreate` command to label the physical partition as an LVM physical volume. Label multiple devices simultaneously by using space-delimited device names as arguments to the `pvcreate` command.

This example labels the `/dev/sdb1` and `/dev/sdb2` devices as PVs that are ready for creating volume groups.

```
root@host:~# pvcreate /dev/sdb1 /dev/sdb2  
Physical volume "/dev/sdb1" successfully created.  
Physical volume "/dev/sdb2" successfully created.  
Creating devices file /etc/lvm/devices/system.devices
```

## Create a Volume Group

The `vgcreate` command builds one or more physical volumes into a volume group. The first argument is a volume group name, followed by one or more physical volumes to allocate to this VG.

This example creates the `vg01` VG by using the `/dev/sdb1` and `/dev/sdb2` PVs.

```
root@host:~# vgcreate vg01 /dev/sdb1 /dev/sdb2
Volume group "vg01" successfully created
```

## Create a Logical Volume

The `lvcreate` command creates a logical volume from the available PEs in a volume group. Use the `lvcreate` command to set the LV name and size, and the VG name to contain this logical volume.

This example creates the `lv01` LV with 300 MiB in the `vg01` VG.

```
root@host:~# lvcreate -n lv01 -L 300M vg01
Logical volume "lv01" created.
```

This command might fail if the volume group does not have enough free physical extents. The LV size rounds up to the next PE size value when the size does not exactly match.

The `lvcreate` command `-L` option requires sizes in bytes, mebibytes (binary megabytes), and gibibytes (binary gigabytes), or similar. The lowercase `-l` requires sizes that are specified as a number of physical extents.

The following commands are two choices for creating the same LV with the same size:

- `lvcreate -n lv01 -L 128M vg01`: Create an LV of size 128 MiB, rounded to the next PE.
- `lvcreate -n lv01 -l 32 vg01`: Create an LV of size 32 PEs at 4 MiB each, total 128 MiB.

## Create a File System on the Logical Volume

Specify the logical volume by using either the `/dev/vgname/lvname` traditional name or the `/dev/mapper/vgname-lvname` kernel device mapper name.

Use the `mkfs` command to create a file system on the new logical volume.

```
root@host:~# mkfs -t xfs /dev/vg01/lv01
...output omitted...
```

Create a mount point by using the `mkdir` command.

```
root@host:~# mkdir /mnt/data
```

To make the file system available persistently, add an entry to the /etc/fstab file.

```
/dev/vg01/lv01 /mnt/data xfs defaults 0 0
```

Update the systemd daemon with the new /etc/fstab configuration file.

```
root@host:~# systemctl daemon-reload
```

Mount the LV by using the mount command.

```
root@host:~# mount /mnt/data/
```

#### Note

You can mount a logical volume by name or by UUID, because LVM parses the PVs by looking for the UUID. This behavior is successful even when the VG was created by using a name, because the PV always contains a UUID.

## Display LVM Component Status

LVM provides various utilities to display the status information for PV, VG, and LV. Use the pvdisplay, vgdisplay, and lvdisplay commands to show the status information for the LVM components.

The associated pvs, vgs, and lvs commands are commonly used and show a subset of the status information, with one line for each entity.

## Display Physical Volume Information

The pvdisplay command displays information about the PVs. Use the command without arguments to list information for all PVs on the system. Providing the name of the PV as the argument to the command shows the information that is specific to the PV.

Use the pvdisplay command to display the information for the /dev/sdb1 PV:

```
root@host:~# pvdisplay /dev/sdb1
--- Physical volume ---
PV Name              /dev/sdb1
VG Name              vg01
PV Size              768.00 MiB / not usable 4.00 MiB
Allocatable          yes
PE Size              4.00 MiB
Total PE             191
Free PE              116
Allocated PE         75
PV UUID              evAimq-gs8W-6X9C-3H6q-ndDk-pvTx-QEZ9Jc
```

- 1 PV Name shows the device name.
- 2 VG Name shows the volume group where the PV is allocated.
- 3 PV Size shows the physical size of the PV, including unusable space.
- 4 PE Size shows the physical extent size.
- 5 Free PE shows the available PE size in the VG to create LVs or to extend existing LVs.

## Display Volume Group Information

The `vgdisplay` command shows the information about volume groups. To list information about all VGs, use the command without arguments. Provide the name of the VG as an argument to list information that is specific to the VG.

Use the `vgdisplay` command to display the information for the `vg01` volume group:

```
root@host:~# vgdisplay vg01
--- Volume group ---
VG Name          vg01
System ID
Format          lvm2
Metadata Areas   2
Metadata Sequence No  2
VG Access        read/write
VG Status         resizable
MAX LV           0
Cur LV           1
Open LV           1
Max PV           0
Cur PV           2
Act PV           2
VG Size          1016.00 MiB
PE Size          4.00 MiB
Total PE         254
Alloc PE / Size  75 / 300.00 MiB
Free PE / Size   179 / 716.00 MiB
VG UUID          F5a4lm-Ln8v-cYM0-YJth-f7fj-lQYZ-1AP8Zu
```

- ① VG Name shows the name of the volume group.
- ② VG Size displays the total size of the storage pool that is available for LV allocation.
- ③ Total PE shows the total size of PE units.
- ④ Free PE / Size shows the available space in the VG to create or extend LVs.

## Display Logical Volume Information

The `lvdisplay` command displays information about logical volumes. Use the command without arguments to list information for all LVs. Providing the name of the LV as an argument displays the information that is specific to the LV.

Use the `lvdisplay` command to display the information for the `/dev/vg01/lv01` volume.

```
root@host:~# lvdisplay /dev/vg01/lv01
--- Logical volume ---
LV Path              /dev/vg01/lv01          1
LV Name              lv01
VG Name              vg01          2
LV UUID              7Toe5z-ozzv-aW4I-mJYQ-2V0V-MiP2-hNGFCf
LV Write Access      read/write
LV Creation host, time servera, 2025-06-18 18:49:54 +0000
LV Status            available
# open               1
LV Size              300.00 MiB          3
Current LE           75          4
Segments             1
Allocation           inherit
Read ahead sectors   auto
- currently set to  8192
Block device         253:0
```

- ① LV Path shows the device name of the LV.
- ② VG Name shows the VG that is used for creating this LV.
- ③ LV Size shows the total size of the LV. Use the file-system tools to determine the free and used space for the LV.
- ④ Current LE shows the number of logical extents that this LV uses.

## References

fdisk(8), gdisk(8), parted(8), partprobe(8), lvm(8), pvcreate(8), vgcreate(8), lvcreate(8), mkfs(8), pvdisk(8), vgdisplay(8), and lvdisplay(8) man pages.

For further information, refer to the *Configuring and Managing Logical Volumes* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/configuring\\_and\\_managing\\_logical\\_volumes/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/configuring_and_managing_logical_volumes/index)

## 11.2. Guided Exercise

### Create Logical Volumes

Create logical volumes, format them with a file system, and mount the volumes.

#### Outcomes

- Create physical volumes, volume groups, and logical volumes with LVM tools.
- Create file systems on logical volumes and persistently mount them.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start lvm-create
```

#### Instructions

1. Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Label the /dev/sdb storage device with the gpt label, and create two physical device partitions with a size of 256 MiB. Set both the partitions to the Linux LVM partition type, and register the partitions with the kernel.

- 2.1. Use the gpt label for the /dev/sdb partition:

```
[root@servera ~]# parted /dev/sdb mklabel gpt
Information: You may need to update /etc/fstab.
```

- 2.2. Create the first partition with a size of 256 MiB.

```
[root@servera ~]# parted /dev/sdb mkpart first 1MiB 257MiB
Information: You may need to update /etc/fstab.
```

**2.3.** Set the first partition to the Linux LVM partition type.

```
[root@servera ~]# parted /dev/sdb set 1 lvm on
Information: You may need to update /etc/fstab.
```

**2.4.** Create the second partition with a size of 256 MiB.

```
[root@servera ~]# parted /dev/sdb mkpart second 257MiB 513MiB
Information: You may need to update /etc/fstab.
```

**2.5.** Set the second partition to the Linux LVM partition type.

```
[root@servera ~]# parted /dev/sdb set 2 lvm on
Information: You may need to update /etc/fstab.
```

**2.6.** Register the new partitions with the kernel.

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

**2.7.** List the partitions on the /dev/sdb storage device using MiB as the display unit. In the Number column, the 1 and 2 values correspond to the /dev/sdb1 and /dev/sdb2 device partitions. The Flags column indicates the partition type.

```
[root@servera ~]# parted /dev/sdb unit mib print
Model: QEMU QEMU HARDDISK (scsi)
Disk /dev/sdb: 5120MiB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start    End     Size   File system  Name  Flags
 1      1.00MiB  257MiB  256MiB          first  lvm
 2      257MiB   513MiB  256MiB          second  lvm
```

**3.** Label the two new partitions as physical volumes.

```
[root@servera ~]# pvcreate /dev/sdb1 /dev/sdb2
Physical volume "/dev/sdb1" successfully created.
Physical volume "/dev/sdb2" successfully created.
Creating devices file /etc/lvm/devices/system.devices
```

- 4.** Create the vg\_servera volume group by using the two new PVs.

```
[root@servera ~]# vgcreate vg_servera /dev/sdb1 /dev/sdb2
Volume group "vg_servera" successfully created
```

- 5.** Create the lv\_servera logical volume by using the vg\_servera VG with a size of 400 MiB. The /dev/vg\_servera/lv\_servera LV is created without a file system.

```
[root@servera ~]# lvcreate -n lv_servera -L 400M vg_servera
Logical volume "lv_servera" created.
```

- 6.** Format the newly created LV with the XFS file system, and mount it persistently on the /data directory.

- 6.1.** Format the lv\_servera LV with the XFS file system.

```
[root@servera ~]# mkfs -t xfs /dev/vg_servera/lv_servera
...output omitted...
meta-data=/dev/vg_servera/lv_servera isize=512    agcount=4, agsize=25600 blks
          =                      sectsz=512  attr=2, projid32bit=1
...output omitted...
data      =                      bsize=4096   blocks=102400, imaxpct=25
          =                      sunit=0     swidth=0 blks
...output omitted...
```

- 6.2.** Create the /data directory as a mount point.

```
[root@servera ~]# mkdir /data
```

- 6.3.** To persistently mount the newly created file system, add the following content in the /etc/fstab file:

```
/dev/vg_servera/lv_servera /data xfs defaults 0 0
```

- 6.4.** Update the systemd daemon with the new /etc/fstab configuration file.

```
[root@servera ~]# systemctl daemon-reload
```

**6.5.** Mount the lv\_servera LV.

```
[root@servera ~]# mount /data
```

- 7.** Verify that the mounted file system is accessible by copying the configuration files from the /etc directory.

**7.1.** Verify that you can copy the configuration files to the /data directory.

```
[root@servera ~]# cp -a /etc/*.* /data
```

**7.2.** Display the number of files in the /data directory.

```
[root@servera ~]# ls /data | wc -l  
30
```

- 8.** Display the information for the LVM.

**8.1.** View the /dev/sdb2 PV information. The output shows that the vg\_servera VG uses the PV. The PV has a size of 256 MiB and a physical extent size of 4 MiB.

The PV contains 63 PEs, of which 27 PEs are available for allocation, and 36 PEs are currently allocated to LVs. Use the following calculation for allocating the volume size in MiB:

- Total 252 MiB (63 PEs x 4 MiB)
- Free 104 MiB (26 PEs x 4 MiB)
- Allocated 148 MiB (37 PEs x 4 MiB)

```
[root@servera ~]# pvdisplay /dev/sdb2
--- Physical volume ---
PV Name          /dev/sdb2
VG Name          vg_servera
PV Size          256.00 MiB / not usable 4.00 MiB
Allocatable      yes
PE Size          4.00 MiB
Total PE         63
Free PE          26
Allocated PE     37
PV UUID          1u3ddN-vh6P-enWw-L00N-XqkS-lQVg-yWzU6N
```

- 8.2.** View the VG information for the vg\_servera VG. The output shows a VG size of 504 MiB with a PE size of 4 MiB. The available size from the VG is 104 MiB.

```
[root@servera ~]# vgdisplay vg_servera
--- Volume group ---
VG Name          vg_servera
System ID
Format          lvm2
Metadata Areas  2
Metadata Sequence No 2
VG Access       read/write
VG Status        resizable
MAX LV           0
Cur LV           1
Open LV          1
Max PV           0
Cur PV           2
Act PV           2
VG Size          504.00 MiB
PE Size          4.00 MiB
Total PE         126
Alloc PE / Size 100 / 400.00 MiB
Free PE / Size  26 / 108.00 MiB
VG UUID          7pCGdT-20Mc-gQox-ZmJg-08Rt-dExh-pJNH5N
```

- 8.3.** View the information for the lv\_servera LV. The output shows the VG name for creating the LV. It also shows an LV size of 400 MiB and an LE size of 100.

```
[root@servera ~]# lvdisplay /dev/vg_servera/lv_servera
--- Logical volume ---
LV Path              /dev/vg_servera/lv_servera
LV Name              lv_servera
VG Name              vg_servera
LV UUID              JAxU5K-zg0x-zyoa-Liti-Lc0M-6vBs-g22itT
LV Write Access      read/write
LV Creation host, time servera, 2025-06-19 20:58:52 +0000
LV Status            available
# open               1
LV Size              400.00 MiB
Current LE           100
Segments             2
Allocation           inherit
Read ahead sectors   auto
- currently set to  8192
Block device         253:0
```

- 8.4.** View the free disk space in human-readable units. The output shows the total size of 336 MiB with the available size of 310 MiB.

```
[root@servera ~]# df -h /data
Filesystem          Size  Used Avail Use% Mounted on
/dev/mapper/vg_servera-lv_servera 336M   27M  310M   8% /data
```

- 9.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish lvm-create
```

## 11.3. Extending a Logical Volume

### Objectives

- Increase the available storage to a logical volume and extend its file system or swap space to use the additional storage.

#### Increase the Storage Capacity on a Logical Volume

A benefit of using logical volumes is increasing their size without experiencing downtime. Add free physical extents to the LV from the VG, to extend the LV capacity and to expand the file system. Use the `vgdisplay` command to confirm that the volume group has enough free space for the LV extension.

Use the `lvextend` command to extend the LV:

```
root@host:~# lvextend -L +500M /dev/vg01/lv01
  Size of logical volume vg01/lv01 changed from 512.00 MiB (128 extents) to
  1012.00 MiB (253 extents).
  Logical volume vg01/lv01 successfully resized.
```

This command increases the size of the `lv01` logical volume by 500 MiB. The (+) plus sign in front of the size means adding this value to the existing size; otherwise, without the plus sign, the value defines the final size of the LV.

The `lvextend` command `-l` option expects the number of PEs as the argument.

The `lvextend` command `-L` option expects sizes in bytes, mebibytes, gibibytes, or similar.

#### Extend an XFS File System to the Logical Volume Size

The `xfs_growfs` command helps to expand the file system to occupy the extended LV. The target file system must be mounted before you use the `xfs_growfs` command. You can continue to use the file system when resizing.

Use the `xfs_growfs` command to expand the XFS file system:

```
root@host:~# xfs_growfs /mnt/data/
...output omitted...
data blocks changed from 131072 to 259072
```

### Important

Always run the `xfs_growfs` command after executing the `lvextend` command. Use the `lvextend` command `-r` option to run the two steps consecutively. An XFS file system must be mounted before resizing; if the file system is not mounted, the `-r` option creates a temporary mount point directory to perform the operation.

Alternatively, after extending the LV, you can resize the file system by using the `fsadm` command. This command is broader in scope, because it supports several other file systems.

## Extend an EXT4 File System to the Logical Volume Size

The steps for extending LV by using the ext4 file system are the same as for LV by using the XFS file system, except for the step to resize the file system.

The `resize2fs` command expands the file system to occupy the new extended LV. You can continue to use the file system when resizing.

```
root@host:~# resize2fs /dev/vg01/lv01
resize2fs 1.47.1 (20-May-2024)
Resizing the filesystem on /dev/vg01/lv01 to 259072 (4k) blocks.
The filesystem on /dev/vg01/lv01 is now 259072 (4k) blocks long.
```

The `xfs_growfs` command expands an XFS file system. This command typically takes the mount point as an argument, or otherwise can indirectly use a device name by mounting the device and then specifying the mount point.

In contrast, the `resize2fs` command resizes ext2, ext3, or ext4 file systems and takes the device name as an argument. The `xfs_growfs` command supports only online resizing, whereas the `resize2fs` command supports both online and offline resizing. Although you can resize an ext4 file system up or down, you can resize an XFS file system only up.

## Extend Swap Space Logical Volumes

You can extend the LVs that are used as swap space; however, the process differs from expanding the ext4 or XFS file system. Logical volumes that are used as swap space must be offline to extend them.

Use the swapoff command to deactivate the swap space on the LV:

```
root@host:~# swapoff -v /dev/vg01/swap  
swapoff /dev/vg01/swap
```

Use the lvextend command to extend the LV:

```
root@host:~# lvextend -L +300M /dev/vg01/swap  
  Size of logical volume vg01/swap changed from 500.00 MiB (125 extents) to  
800.00 MiB (200 extents).  
  Logical volume vg01/swap successfully resized.
```

Use the mkswap command to format the LV as a swap space:

```
root@host:~# mkswap /dev/vg01/swap  
mkswap: /dev/vg01/swap: warning: wiping old swap signature.  
Setting up swapspace version 1, size = 800 MiB (838856704 bytes)  
no label, UUID=690ef204-9d59-4b3c-b145-eb8ba591ead8
```

Use the swapon command to activate the swap space on the LV:

```
root@host:~# swapon /dev/vg01/swap
```

### Important

Be aware of potential issues when performing this operation on a live system.

Sometimes, not enough physical memory space is available to free up by flushing and relocating pages to memory when the swap space is turned off. In those scenarios, you can run into out-of-memory (OOM) errors and the system might behave unexpectedly.

Make sure that the swap space is not aggressively used when performing this operation.

Alternatively, you can create an additional LV-based swap space with the target size, enable it, and then disable the previous volume. After the previous LV is fully disabled, you can delete the LV to reclaim the extents for the VG.

### References

`lvextend(8)`, `xfs_growfs(8)`, `resize2fs(8)`, `swapoff(8)`, `mkswap(8)`, and `swapon(8)` man pages



## 11.4. Guided Exercise

### Extend a Logical Volume

Extend a logical volume to increase the available storage and extend its file system to use the additional storage.

#### Outcomes

- Extend the storage of an existing logical volume.
- Extend the file system of a logical volume to use the available storage.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start lvm-extend
```

#### Instructions

1. Log in to the servera machine as the student user and switch to the root user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Create the third partition with a size of 512 MiB on the `/dev/sdb` storage device. Set the partition to the Linux LVM partition type, and register the partition with the kernel.

- 2.1. Create the third partition with a size of 512 MiB.

```
[root@servera ~]# parted /dev/sdb mkpart third 514MiB 1026MiB
Information: You may need to update /etc/fstab.
```

- 2.2. Set the third partition to the Linux LVM partition type.

```
[root@servera ~]# parted /dev/sdb set 3 lvm on
Information: You may need to update /etc/fstab.
```

**2.3.** Register the new partition with the kernel.

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

**3.** Label the third partition as a physical volume.

```
[root@servera ~]# pvcreate /dev/sdb3
Physical volume "/dev/sdb3" successfully created.
```

**4.** Using the third partition, extend the file system on the lv\_servera LV to a total size of 700 MiB.

**4.1.** Extend the vg\_servera VG by using the /dev/sdb3 PV.

```
[root@servera ~]# vgextend vg_servera /dev/sdb3
Volume group "vg_servera" successfully extended
```

**4.2.** Extend the lv\_servera LV to 700 MiB.

```
[root@servera ~]# lvextend -L 700M /dev/vg_servera/lv_servera
Size of logical volume vg_servera/lv_servera changed from 400.00 MiB (100
extents) to 700.00 MiB (175 extents).
Logical volume vg_servera/lv_servera successfully resized.
```

**4.3.** Extend the XFS file system mounted on /data by using the free space on the LV.

```
[root@servera ~]# xfs_growfs /data
...output omitted...
data blocks changed from 102400 to 179200
```

**5.** Verify that the LV size is extended, and that the contents are still present in the volume.

**5.1.** Verify the size of the extended LV.

```
[root@servera ~]# lvdisplay /dev/vg_servera/lv_servera
--- Logical volume ---
LV Path              /dev/vg_servera/lv_servera
LV Name              lv_servera
VG Name              vg_servera
LV UUID              a7N74m-RwUN-ik3H-ZuhJ-adQW-dACp-3Qek3d
LV Write Access      read/write
LV Creation host, time servera, 2025-06-23 21:04:31 +0000
LV Status            available
# open               1
LV Size              700.00 MiB
Current LE           175
Segments             3
Allocation           inherit
Read ahead sectors   auto
- currently set to  256
Block device         253:0
```

## 5.2. Verify the new file-system size.

```
[root@servera ~]# df -h /data
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/vg_servera-lv_servera  636M   33M  604M   6% /data
```

## 5.3. Verify that the previously copied files are still present.

```
[root@servera ~]# ls /data | wc -l
30
```

## 6. Return to the workstation machine as the student user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
```

# Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish lvm-extend
```

## 11.5. Replacing Physical Volumes and Managing LVM

### Objectives

- Replace small or failing physical volumes in a volume group, and decommission inactive logical volumes, volume groups, or physical volumes when they are no longer needed.

### Replacing a Physical Volume in a Volume Group

You might face a scenario where you must replace a PV in a VG due to hardware failure, insufficient space, or other reasons. LVM provides command-line tools to perform this operation safely, even with live and mounted LVs.

#### Extend Volume Group Storage

You might need to add more disk space to extend a VG. You can add physical volumes to a VG to extend its available size.

Prepare the physical device and create the physical volume if it does not exist.

```
root@host:~# pvcreate /dev/sdb3
Physical volume "/dev/sdb3" successfully created.
```

The vgextend command adds the new PV to the VG. This command extends the vg01 VG by the /dev/sdb3 PV size.

```
root@host:~# vgextend vg01 /dev/sdb3
Volume group "vg01" successfully extended
```

#### Reduce Volume Group Storage

Reducing the storage of a VG involves removing unused PVs from the VG. The pvmove command moves data from extents on one PV to extents on another PV with enough free extents in the same VG. You can continue to use the LV from the same VG when reducing the storage.

Use the pvmove command -A option to automatically back up the metadata of the VG after a change. This option uses the vgcfgbackup command to back up metadata automatically.

```
root@host:~# pvmove -A y /dev/sdb1
/dev/sdb1: Moved: 2.74%
...output omitted...
/dev/sdb1: Moved: 100.00%
```

### **Warning**

Before using the `pvmove` command, back up the data that is stored on all LVs in the VG. An unexpected power loss during the operation might leave the VG in an inconsistent state, which might cause a loss of data on LVs.

Use the `vgreduce` command to remove a PV from a VG.

```
root@host:~# vgreduce vg01 /dev/sdb1
Removed "/dev/sdb1" from volume group "vg01"
```

## Remove LVM Storage

Use the `lvremove`, `vgremove`, or `pvremove` commands to remove an LVM component that is no longer required.

### **Warning**

The `lvremove`, `vgremove`, and `pvremove` commands are destructive, and unrecoverable operations. Use these commands only when you are sure that the LVM component is no longer needed or in use.

## Prepare the File System

Move to another file system all data that must be kept. Use the `umount` command to unmount the file system, and then remove any `/etc/fstab` entries that are associated with this file system.

```
root@host:~# umount /mnt/data
```

## Remove the Logical Volume

Use the `lvremove` command to remove a logical volume that is no longer needed. Unmount the LV file system before running this command. The command prompts for confirmation before removing the LV.

```
root@host:~# lvremove /dev/vg01/lv01
Do you really want to remove active logical volume vg01/lv01? [y/n]: y
Logical volume "lv01" successfully removed.
```

The LV's physical extents are freed and available to assign to existing or new LVs in the volume group.

## Remove the Volume Group

Use the vgremove command to remove a volume group that is no longer needed.

```
root@host:~# vgremove vg01
Volume group "vg01" successfully removed
```

The VG's physical volumes are freed and available to assign to existing or new VGs on the system.

## Remove the Physical Volumes

Use the pvremove command to remove physical volumes that are no longer needed. Use a space-delimited list of PV devices if you need to remove more than one device at a time. This command deletes the PV metadata from the partition (or disk).

```
root@host:~# pvremove /dev/sdb2 /dev/sdb3
Labels on physical volume "/dev/sdb2" successfully wiped.
Labels on physical volume "/dev/sdb3" successfully wiped.
```

The partition is now available for reallocation or reformatting.

### References

`vgextend(8)`, `pvmove(8)`, `vgcfgbackup(8)`, `vgreduce(8)`, `lvremove(8)`, `vgremove(8)`, and `pvremove(8)` man pages

## 11.6. Quiz

### Replacing Physical Volumes and Managing LVM

Choose the correct answers to the following questions:

1. Which command adds a PV to an existing VG to increase its capacity?

- a. `vgexpand /dev/vg01 /dev/sdb3`
- b. `vgextend /dev/sdb3 /dev/vg01`
- c. `vgextend /dev/vg01 /dev/sdb3`
- d. `vggrow /dev/sdb3 /dev/vg01`

2. What does the `pvmove` command do?

- a. It deletes all metadata from a PV.
- b. It removes a PV from a VG.
- c. It deletes a PV from the system.
- d. It moves data from extents on one PV to extents on another PV.

3. Which command removes a PV from a VG to reduce its capacity?

- a. `vgreduce /dev/vg01 /dev/sdb3`
- b. `vgremove /dev/vg01 /dev/sdb3`
- c. `vgreduce /dev/sdb3`
- d. `vgrescall /dev/vg01 /dev/sdb3`

4. True or False: The `lvremove`, `vgremove`, and `pvremove` commands are reversible operations.

- a. True
- b. It depends on the type of LVM component.
- c. False

5. What is the correct order to remove the entire LVM component structure?

- a. `umount > lvremove > vgremove > pvremove`
- b. `lvremove > umount > pvremove > vgremove`
- c. `vgremove > pvremove > lvremove > umount`
- d. `pvremove > vgremove > lvremove > umount`

## Solution

---

1. Which command adds a PV to an existing VG to increase its capacity?

- a. `vgexpand /dev/vg01 /dev/sdb3`
- b. `vgextend /dev/sdb3 /dev/vg01`
- c. `vgextend /dev/vg01 /dev/sdb3`
- d. `vggrow /dev/sdb3 /dev/vg01`

2. What does the `pvmove` command do?

- a. It deletes all metadata from a PV.
- b. It removes a PV from a VG.
- c. It deletes a PV from the system.
- d. **It moves data from extents on one PV to extents on another PV.**

3. Which command removes a PV from a VG to reduce its capacity?

- a. `vgreduce /dev/vg01 /dev/sdb3`
- b. `vgremove /dev/vg01 /dev/sdb3`
- c. `vgreduce /dev/sdb3`
- d. `vgreduce /dev/vg01 /dev/sdb3`

4. True or False: The `lvremove`, `vgremove`, and `pvremove` commands are reversible operations.

- a. True
- b. It depends on the type of LVM component.
- c. **False**

5. What is the correct order to remove the entire LVM component structure?

- a. `umount > lvremove > vgremove > pvremove`
- b. `lvremove > umount > pvremove > vgremove`
- c. `vgremove > pvremove > lvremove > umount`
- d. `pvremove > vgremove > lvremove > umount`

## 11.7. Lab

# Manage Storage with Logical Volume Manager

Create, manage, and extend logical volumes that can contain file systems and swap spaces.

## Outcomes

- Create a logical volume (LV) with an XFS file system.
- Resize a logical volume to a specified size.
- Persistently mount the volume.

## Prerequisites

As the student user on the workstation machine, run the following `lab start lvm-review` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start lvm-review
```

## Instructions

On the serverb machine, the `/storage/data1` directory is running out of disk space. The `vg_01_serverb` volume group (VG) has insufficient space to extend the `lv_01_serverb` logical volume (LV). To extend the VG, you must first create a secondary 1024 MiB partition on the `/dev/sdb` disk. The VG must not exceed 2048 MiB.

Then, extend the `lv_01_serverb` LV to 1024 MiB. Ensure that the `lv_01_serverb` LV remains persistently mounted on the `/storage/data1` directory.

Finally, create the `lv_02_serverb` LV with 512 MiB in the `vg_01_serverb` VG. Create the XFS file system on the newly created volume. Persistently mount the newly created logical volume on the `/storage/data2` directory.

## Important

Note especially the specification of the partition size in MiB ( $2^{20}$  bytes). If you create the partition in MB ( $10^6$  bytes), it does not satisfy the evaluation criteria, because 1 MiB = 1.048576 MB.

Although the default unit when using the `parted` command is MB, you can verify the size of the partitions in MiB units by using the `unit MiB` option.

1. On the `serverb` machine, create the secondary 1024 MiB partition on the `/dev/sdb` disk. Register the partition with the kernel.
2. Initialize the secondary partition as a physical volume.
3. Extend the `vg_01_serverb` volume group to use the secondary partition.
4. Extend the `lv_01_serverb` logical volume to 1024 MiB, and resize the XFS file system.
5. In the `vg_01_serverb` volume group, create the `lv_02_serverb` logical volume with 512 MiB. Add an XFS file system and mount it persistently on the `/storage/data2` directory.
6. Verify that the `lv_01_serverb` and `lv_02_serverb` LVs are mounted with the intended size.
7. Return to the workstation machine as the student user.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade lvm-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish lvm-review
```

## Solution

---

Create, manage, and extend logical volumes that can contain file systems and swap spaces.

## Outcomes

- Create a logical volume (LV) with an XFS file system.
- Resize a logical volume to a specified size.
- Persistently mount the volume.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start lvm-review
```

## Instructions

On the serverb machine, the /storage/data1 directory is running out of disk space. The vg\_01\_serverb volume group (VG) has insufficient space to extend the lv\_01\_serverb logical volume (LV). To extend the VG, you must first create a secondary 1024 MiB partition on the /dev/sdb disk. The VG must not exceed 2048 MiB.

Then, extend the lv\_01\_serverb LV to 1024 MiB. Ensure that the lv\_01\_serverb LV remains persistently mounted on the /storage/data1 directory.

Finally, create the lv\_02\_serverb LV with 512 MiB in the vg\_01\_serverb VG. Create the XFS file system on the newly created volume. Persistently mount the newly created logical volume on the /storage/data2 directory.

### Important

Note especially the specification of the partition size in MiB ( $2^{20}$  bytes). If you create the partition in MB ( $10^6$  bytes), it does not satisfy the evaluation criteria, because 1 MiB = 1.048576 MB.

Although the default unit when using the parted command is MB, you can verify the size of the partitions in MiB units by using the unit MiB option.

1. On the serverb machine, create the secondary 1024 MiB partition on the /dev/sdb disk. Register the partition with the kernel.

- 1.1. Log in to the serverb machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 1.2. Print the partition sizes in MiB to determine where the first partition ends for the /dev/sdb partition.

```
[root@serverb ~]# parted /dev/sdb unit MiB print
...output omitted...

Number  Start   End     Size    File system  Name      Flags
 1        1.00MiB 1025MiB 1024MiB          primary
```

- 1.3. Create the secondary 1024 MiB partition on the /dev/sdb partition.

```
[root@serverb ~]# parted /dev/sdb mkpart secondary 1025MiB 2049MiB
...output omitted...
```

- 1.4. Set the secondary partition to the lvm partition type.

```
[root@serverb ~]# parted /dev/sdb set 2 lvm on
...output omitted...
```

- 1.5. Register the new partition with the kernel.

```
[root@serverb ~]# udevadm settle
```

2. Initialize the secondary partition as a physical volume.

- 2.1. Initialize the secondary partition as a PV.

```
[root@serverb ~]# pvcreate /dev/sdb2
Physical volume "/dev/sdb2" successfully created.
```

- 3.** Extend the vg\_01\_serverb volume group to use the secondary partition.

- 3.1.** Extend the vg\_01\_serverb VG by using the /dev/sdb2 PV.

```
[root@serverb ~]# vgextend vg_01_serverb /dev/sdb2
Volume group "vg_01_serverb" successfully extended
```

- 4.** Extend the lv\_01\_serverb logical volume to 1024 MiB, and resize the XFS file system.

- 4.1.** Extend the lv\_01\_serverb LV to 1024 MiB.

Alternatively, you can use the lvcreate command -L +624M option to resize the LV.

```
[root@serverb ~]# lvextend -L 1024M /dev/vg_01_serverb/lv_01_serverb
Size of logical volume vg_01_serverb/lv_01_serverb changed from 400.00 MiB
(100 extents) to 1.00 GiB (256 extents).
Logical volume vg_01_serverb/lv_01_serverb successfully resized.
```

- 4.2.** Extend the XFS file system to consume the remaining space on the LV.

#### Note

The xfs\_growfs command introduces an extra step to extend the file system. An alternative would be to use the lvextend command -r option.

```
[root@serverb ~]# xfs_growfs /storage/data1
...output omitted...
data blocks changed from 102400 to 262144
```

- 5.** In the vg\_01\_serverb volume group, create the lv\_02\_serverb logical volume with 512 MiB. Add an XFS file system and mount it persistently on the /storage/data2 directory.

- 5.1.** Create the lv\_02\_serverb LV with 512 MiB from the vg\_01\_serverb VG.

```
[root@serverb ~]# lvcreate -n lv_02_serverb -L 512M vg_01_serverb
Logical volume "lv_02_serverb" created.
```

- 5.2.** Create the XFS file system on the lv\_02\_serverb LV.

```
[root@serverb ~]# mkfs -t xfs /dev/vg_01_serverb/lv_02_serverb  
...output omitted...
```

**5.3.** Create the /storage/data2 directory.

```
[root@serverb ~]# mkdir /storage/data2
```

**5.4.** Add the following line to the end of the /etc/fstab file:

```
/dev/vg_01_serverb/lv_02_serverb /storage/data2 xfs defaults 0 0
```

**5.5.** Update the systemd daemon with the new /etc/fstab configuration file.

```
[root@serverb ~]# systemctl daemon-reload
```

**5.6.** Mount the lv\_02\_serverb LV.

```
[root@serverb ~]# mount /storage/data2
```

**6.** Verify that the lv\_01\_serverb and lv\_02\_serverb LVs are mounted with the intended size.

**6.1.** Use the df command to verify the lv\_01\_serverb LV size.

```
[root@serverb ~]# df -h /storage/data1  
Filesystem Size Used Avail Use% Mounted on  
/dev/mapper/vg_01_serverb-lv_01_serverb 960M 40M 921M 5% /storage/data1
```

**6.2.** Verify the lv\_02\_serverb LV size.

```
[root@serverb ~]# df -h /storage/data2  
Filesystem Size Used Avail Use% Mounted on  
/dev/mapper/vg_01_serverb-lv_02_serverb 448M 35M 414M 8% /storage/data2
```

**6.3.** Verify the lv\_01\_serverb LV details.

```
[root@serverb ~]# lvdisplay /dev/vg_01_serverb/lv_01_serverb
--- Logical volume ---
LV Path          /dev/vg_01_serverb/lv_01_serverb
LV Name          lv_01_serverb
VG Name          vg_01_serverb
LV UUID          0BT2Tv-GwSV-1ioB-I01y-ctAk-mPF1-7vFxmo
LV Write Access  read/write
LV Creation host, time serverb, 2025-06-25 18:31:53 +0000
LV Status        available
# open           1
LV Size          1.00 GiB
Current LE      256
Segments         2
Allocation       inherit
Read ahead sectors auto
- currently set to 256
Block device    253:0
```

#### 6.4. Verify the lv\_02\_serverb LV details.

```
[root@serverb ~]# lvdisplay /dev/vg_01_serverb/lv_02_serverb
--- Logical volume ---
LV Path          /dev/vg_01_serverb/lv_02_serverb
LV Name          lv_02_serverb
VG Name          vg_01_serverb
LV UUID          RpJMUc-xSeh-Dcvj-Jp0a-nqvA-bIvT-DJzK3x
LV Write Access  read/write
LV Creation host, time serverb, 2025-06-25 18:36:51 +0000
LV Status        available
# open           1
LV Size          512.00 MiB
Current LE      128
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device    253:1
```

#### 7. Return to the workstation machine as the student user.

```
[root@serverb ~]# exit
[student@serverb ~]$ exit
logout
Connection to serverb closed.
student@workstation:~$
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work.

Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade lvm-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish lvm-review
```

## 11.8. Summary

---

In this lesson, you learned about the Logical Volume Manager and how it uses physical volumes, volume groups, and logical volumes to provide flexible storage space. Each component has specific commands for its management.

The `pvcreate` command creates the PVs; the `vgcreate` command creates the VGs; and the `lvcreate` command creates the LVs.

The `vgextend` command adds a PV to an existing VG, and the `lvextend` command increases the size of a LV.

The `resize2fs` and `xfs_growfs` commands resize the file system on a LV after its size is modified.

The `pvmove` command moves data from one PV to another, which frees up space on the original PV and enables it to be removed or repurposed.

The `lvremove` command deletes the LVs; the `vgremove` command deletes the VGs; and the `pvremove` command deletes the PVs.

## Chapter 12.

# Controlling and Troubleshooting the Boot Process

---

## Goal

Manage how the system boots to control which services start and to troubleshoot and repair boot-time problems.

## Sections

- Managing the Boot Loader and Kernel Command Line (and Guided Exercise)
- Exploring the Boot Process and Selecting a Boot Target (and Guided Exercise)
- Repairing Damaged File Systems at Boot Time (and Guided Exercise)

## Lab

- Control and Troubleshoot the Boot Process

## 12.1. Managing the Boot Loader and Kernel Command Line

### Objectives

- Describe how the boot loader selects and loads the operating system, reinstall the boot loader, and modify the boot-loader configuration and kernel command-line arguments.

### The Boot Loader

Modern computer systems are complex combinations of hardware and software. Starting from an undefined, powered-down state to a running system with a login prompt requires many pieces of hardware and software to work together.

When a computer is powered on, it loads a firmware boot system from a ROM chip on the motherboard. In modern x86 machines, this firmware boot system is typically either *Unified Extensible Firmware Interface* (UEFI) or *Basic Input/Output System* (BIOS). BIOS was more common in machines manufactured before 2020, and UEFI is the most common system present in current motherboards.

The job of the firmware boot system is to load a boot loader from a disk and to pass the control of the boot process to the boot loader. Boot loaders are programs designed to interface with the firmware boot system, set the system for boot, and then start the kernel to continue the system initialization. The default boot loader for Red Hat Enterprise Linux 10 is the GRand Unified Bootloader version 2 (GRUB2).

#### GRUB2 with UEFI

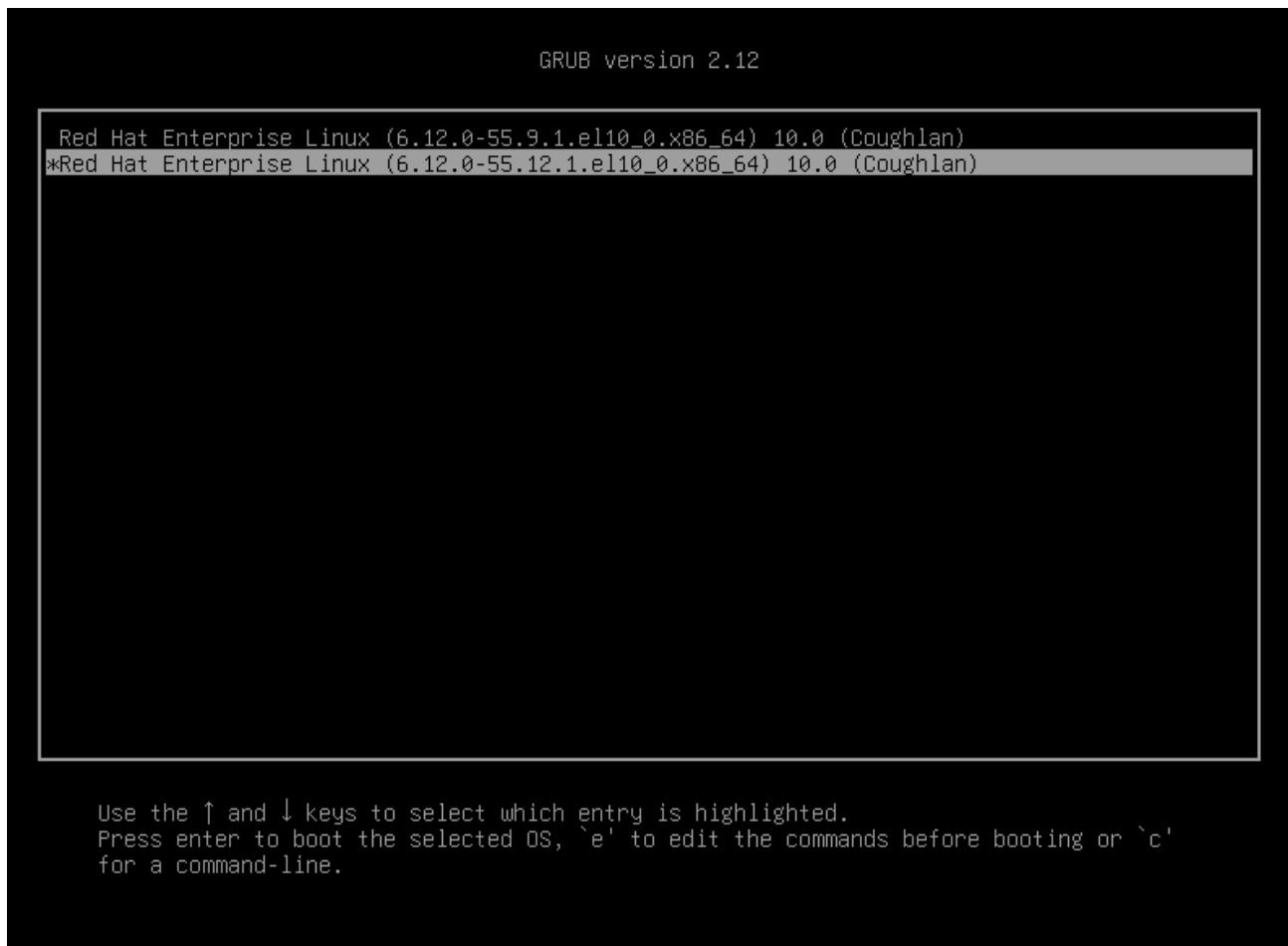
In a UEFI system, the boot firmware reads settings from nonvolatile RAM (NVRAM) to determine which disk and Extensible Firmware Interface (EFI) application to load. The disk typically contains an EFI system partition, which is usually mounted on the /boot/efi directory. GRUB2 generates files that the UEFI system can load. After the GRUB2 boot-loader EFI application has been loaded, the system moves to the next boot stage.

#### GRUB2 with BIOS

In a BIOS system, the boot firmware reads a specially formatted boot sector on a disk for the boot.img boot-loader image. This boot-loader image loads the core.img image from the unpartitioned MBR space before the first partition. The core.img image loads extra modules that can initialize file systems, and the system moves to the next boot stage. GRUB2 generates and installs these images into the MBR space on the disk.

## Further Boot Stages

After GRUB2 has fully initialized and loaded disks, it presents a boot menu to the user.



**Figure 12. GRUB2 menu**

From this boot menu, you can select which preconfigured kernel to boot, as well as apply any kernel arguments to boot. The Linux kernel package typically generates these boot-menu entries when you install a new version.

To modify a boot entry from the GRUB2 menu, you can use the built-in editor. You can access the editor in GRUB2 by pressing **E**.

The image shows the GRUB2 editor interface. At the top, it says "GRUB version 2.12". Below that is a large text area containing a kernel command line. The command line includes: "load\_video", "set gfxpayload=keep", "insmod gzio", "linux (\$root)/boot/vmlinuz-6.12.0-55.12.1.el10\_0.x86\_64 root=UUID=15507695-22bb-4c65-94e6-a4\38e095983f console=tty0 console=ttyS0,115200n8 no\_timer\_check crashkernel=2G-64G:256M,64G-:5\12M", and "initrd (\$root)/boot/initramfs-6.12.0-55.12.1.el10\_0.x86\_64.img \$tuned\_initrd". At the bottom of the screen, there is a message: "Minimum Emacs-like screen editing is supported. TAB lists completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for a command-line or ESC to discard edits and return to the GRUB menu."

**Figure 13. GRUB2 editor**

The changes made within the GRUB2 editor apply to only that boot instance. To make any changes persistent in future boot instances, you can make those changes by using the `grubby` command.

## Controlling GRUB2 from a Booted System

You can modify the default configuration of GRUB2 by using the `grubby` command. You can use the `grubby` command to perform various tasks, such as adding or removing kernel arguments, changing the default boot-menu entry, and viewing information about existing boot-menu entries.

To view information about an existing entry, you can use the `grubby` command `--info` option.

```
user@host:~$ grub --info 1
index=1 ①
kernel="/boot/vmlinuz-6.12.0-55.12.1.el10_0.x86_64" ②
args="console=tty0 ... crashkernel=2G-64G:256M,64G-:512M $tuned_params" ③
root="UUID=15507695-22bb-4c65-94e6-a438e095983f" ④
initrd="/boot/initramfs-6.12.0-55.12.1.el10_0.x86_64.img $tuned_initrd" ⑤
title="Red Hat Enterprise Linux (6.12.0-55.12.1.el10_0.x86_64) 10.0 (Coughlan)" ⑥
id="b70d796f931842cb9776f658cd068435-6.12.0-55.12.1.el10_0.x86_64" ⑦
```

- ① The index value is the entry number for the boot-menu list, starting at 0.
- ② The kernel value is the path to the Linux kernel that GRUB2 loads.
- ③ The args value is a list of kernel arguments that are passed to the Linux kernel.
- ④ The root value is the block device that the kernel and the initramfs image exist on.
- ⑤ The initrd value is the path to the initial RAM file system image, which the early stages of Linux boot within.
- ⑥ The title value is the name that is shown for the boot entry in the GRUB2 boot menu.
- ⑦ The id value is a unique identifier for this boot entry.

To change the default boot entry, so that GRUB2 automatically chooses it on boot, you can use the `grubby` command `--set-default-index` option.

```
root@host:~# grub --set-default-index 0
The default is /boot/loader/entries/ffffffffffffffffff-6.12.0-
55.9.1.el10_0.x86_64.conf with index 0 and kernel /boot/vmlinuz-6.12.0-
55.9.1.el10_0.x86_64
```

## Kernel Command-line Arguments

You can pass various arguments to the Linux kernel to configure the system at boot. In some cases, a command-line argument is the only way to change a configuration, such as defining how the kernel behaves if it crashes. The available kernel command-line arguments and exact behavior depend on the version of the Linux kernel that you have installed.

To add a command-line argument, or make other changes related to the kernel, you can use the `grubby` command `--update-kernel` option.

The following example adds the `rhgb quiet` command-line argument to suppress kernel boot messages by using the `--args` option:

```
root@host:~# grubby --update-kernel /boot/vmlinuz-6.12.0-55.9.1.el10_0.x86_64 \
--args="rhgb quiet"
```

To remove the command-line argument, add the `rhgb quiet` argument by using the `--remove-args` option:

```
root@host:~# grubby --update-kernel /boot/vmlinuz-6.12.0-55.9.1.el10_0.x86_64 \
--remove-args="rhgb quiet"
```

## References

`grubby(8)`, `kernel-command-line(7)`, `bootparam(7)`, and `dracut.cmdline(7)` man pages

For more information, refer to the *Configuring Kernel Command-line Parameters* chapter in the *Managing, Monitoring, and Updating the Kernel* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/managing\\_monitoring\\_and\\_updating\\_the\\_kernel/index#configuring-kernel-command-line-parameters](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/managing_monitoring_and_updating_the_kernel/index#configuring-kernel-command-line-parameters)

## 12.2. Guided Exercise

### Manage the Boot Loader and Kernel Command Line

Modify the boot-loader configuration to control the boot process.

#### Outcomes

- Reboot the system and examine the GRUB2 menu.
- View the current boot-loader configuration from the command line.
- Reconfigure the boot loader to select a different kernel as the default.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start boot-grub
```

#### Instructions

1. Reboot the servera machine, and then inspect the GRUB2 menu. Select the default boot option.
  - 1.1. Open the servera machine console, and then reboot the machine by sending **Ctrl + Alt + Del** to your system by using the relevant button or menu entry.
  - 1.2. After the boot loader initializes, press **Esc** to stop the boot-loader timer. The selected entry is the default kernel for booting the machine.
  - 1.3. Press **E** to open the editor for the initial command-line arguments for the default boot option. Do not make any changes to the configuration.
  - 1.4. Press **Ctrl + X** or use the on-screen keyboard to send **F10** to the console to boot the system with the default configuration.
2. View information about the default kernel boot configuration.
  - 2.1. Log in to the servera machine as the **root** user. Use **redhat** as the password.

```
servera login: root
Password: redhat
...output omitted...
[root@servera ~]#
```

**2.2.** Get the index of the default kernel by using the grubby command:

```
[root@servera ~]# grubby --default-index
1
```

**2.3.** View the information of the kernel at index 1:

```
[root@servera ~]# grubby --info 1
index=1
kernel="/boot/vmlinuz-6.12.0-55.12.1.el10_0.x86_64"
args="console=tty0 console=ttyS0,115200n8 no_timer_check crashkernel=2G-
64G:256M,64G-:512M $tuned_params"
root="UUID=15507695-22bb-4c65-94e6-a438e095983f"
initrd="/boot/initramfs-6.12.0-55.12.1.el10_0.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (6.12.0-55.12.1.el10_0.x86_64) 10.0 (Coughlan)"
id="b70d796f931842cb9776f658cd068435-6.12.0-55.12.1.el10_0.x86_64"
```

**3.** Use the grubby command to select the kernel at index 0 for boot. Note the kernel of this index to use in the next step.

```
[root@servera ~]# grubby --set-default-index 0
The default is /boot/loader/entries/ffffffffffffffffff-6.12.0-
55.9.1.el10_0.x86_64.conf with index 0 and kernel /boot/vmlinuz-6.12.0-
55.9.1.el10_0.x86_64
```

**4.** Use the grubby command to add the `rhgb quiet` kernel command-line argument to the new default kernel. Reboot the system to see the effects.

**4.1.** Add the `rhgb quiet` kernel command-line argument to the new default kernel.

```
[root@servera ~]# grubby --update-kernel \
/boot/vmlinuz-6.12.0-55.9.1.el10_0.x86_64 --args="rhgb quiet"
```

**4.2.** Reboot the servera machine.

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

5. The servera machine now boots by using the newly set default kernel. Note that the initial round of kernel messages are now suppressed and the first messages that you see are from the systemd unit.
6. Revert the changes to the command-line arguments by using the grubby command and change back to the most recent kernel. Reboot the system again and verify that it boots correctly.

**6.1.** Log in to the servera machine as the root user. Use redhat as the password

```
servera login: root  
Password: redhat  
...output omitted...  
[root@servera ~]#
```

**6.2.** Change the default kernel index to 1. Note the kernel of this index to use in the next step.

```
[root@servera ~]# grub --set-default 1  
The default is /boot/loader/entries/b70d796f931842cb9776f658cd068435-6.12.0-  
55.12.1.el10_0.x86_64.conf with index 1 and kernel /boot/vmlinuz-6.12.0-  
55.12.1.el10_0.x86_64
```

**6.3.** Revert the changes to the /boot/vmlinuz-6.12.0-55.9.1.el10\_0.x86\_64 kernel.

```
[root@servera ~]# grub --update-kernel \  
/boot/vmlinuz-6.12.0-55.9.1.el10_0.x86_64 --remove-args="rhgb quiet"
```

**6.4.** Reboot the system to verify that it boots correctly.

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

7. Close the servera machine console.

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish boot-grub
```

## 12.3. Exploring the Boot Process and Selecting a Boot Target

### Objectives

- Describe how a Red Hat Enterprise Linux system starts, change the default boot target, and manually boot the system to a target other than the default.

### System Initialization in Red Hat Enterprise Linux 10

After the system is powered on, and the boot loader is started, the system initialization process begins.

The following list gives a high-level overview of the system initialization process in Red Hat Enterprise Linux 10.

- The boot loader loads the kernel and the `initramfs` image from disk and places them in memory. The `initramfs` image is an archive with the kernel modules for all the required hardware at boot, initialization scripts, and more. In RHEL 10, the `initramfs` image contains a bootable `root` file system with a running kernel and a `systemd` unit. You can use the `lsinitrd` command to inspect the `initramfs` image and to extract its contents.
- The boot loader passes control to the kernel, along with any specified options on the kernel command line, and the location of the `initramfs` image in memory.
- The kernel initializes all hardware for which it can find a driver in the `initramfs` image, and then executes the `/sbin/init` script from the `initramfs` image as PID 1. You can override this script by using the `init=` kernel parameter. In RHEL 10, the `/sbin/init` script is a symbolic link to the `systemd` unit.
- The `systemd` unit from the `initramfs` image executes all units for the `initrd.target` target. This unit includes mounting the `root` file system on the `/sysroot` directory, which is listed in the `/etc/fstab` file on the disk.
- The kernel switches (pivots) the `root` file system from the `initramfs` image to the `root` file system in the `/sysroot` directory. The `systemd` unit then re-executes itself by using the installed copy of the `systemd` unit on the disk.
- Then, the `systemd` unit is in full control of the boot process. The `systemd` unit looks for a default target, which is either configured on the system or is passed from the kernel command line. The `systemd` unit then starts (and stops) units to comply with the configuration for that target, and automatically resolves dependencies between units.

- These targets typically start a text-based login or a graphical login screen, thus completing the boot process.

## Selecting a systemd Target

A systemd target is a set of systemd units that the system must start to reach an intended state.

The following table lists the most important targets.

**Table 16. Commonly Used Targets**

Target	Purpose
graphical.target	This target supports multiple users, and provides graphical and text-based logins.
multi-user.target	This target supports multiple users, and provides text-based logins only.
rescue.target	This target provides a single-user system to enable repairing your system.
emergency.target	This target starts the most minimal system for repairing your system when the <code>rescue.target</code> unit fails to start.

A target can be a part of another target. For example, the `graphical.target` unit includes the `multi-user.target` unit, which depends on the `basic.target` unit and others.

You can view these dependencies with the following command:

```
user@host:~$ systemctl list-dependencies graphical.target | grep target
graphical.target
* └─multi-user.target
*   ├─basic.target
*   ├─paths.target
*   ├─slices.target
*   ├─sockets.target
*   ├─sysinit.target
*   ├─cryptsetup.target
*   ├─integritysetup.target
*   ├─local-fs.target
...output omitted...
```

To list the available targets, use the following command:

```
user@host:~$ systemctl list-units --type=target --all
UNIT                      LOAD    ACTIVE   SUB     DESCRIPTION
-----  
basic.target               loaded  active   active  Basic System
...output omitted...
cloud-config.target         loaded  active   active  Cloud-config availability
cloud-init.target           loaded  active   active  Cloud-init target
cryptsetup-pre.target       loaded  inactive dead    Local Encrypted Volumes  
(Pre)
cryptsetup.target            loaded  active   active  Local Encrypted Volumes
...output omitted...
```

## Select a Target at Runtime

On a running system, administrators can switch to a different target by using the `systemctl isolate` command.

```
root@host:~# systemctl isolate multi-user.target
```

Isolating a target stops all services that the target does not require (and its dependencies), and starts any required services that are not yet started.

Not all targets can be isolated. You can isolate only targets where the `AllowIsolate=yes` option is set in their unit files.

For example, you can isolate the graphical target:

```
user@host:~$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
...output omitted...
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
```

But you cannot isolate the cryptsetup target:

```
user@host:~$ systemctl cat cryptsetup.target
# /usr/lib/systemd/system/cryptsetup.target
...output omitted...
[Unit]
Description=Local Encrypted Volumes
Documentation=man:systemd.special(7)
```

## Set a Default Target

When the system starts, the `systemd` unit activates the `default.target` target. Typically, the `default.target` file in the `/etc/systemd/system` directory is a symbolic link to either the `graphical.target` or the `multi-user.target` targets.

Instead of editing this symbolic link manually, the `systemctl` command provides two subcommands to manage this link: `get-default` and `set-default`.

Use the `systemctl` command to get the default target:

```
root@host:~# systemctl get-default
multi-user.target
```

Use the `systemctl` command to set `graphical.target` as the default target:

```
root@host:~# systemctl set-default graphical.target
Removed '/etc/systemd/system/default.target'.
Created symlink '/etc/systemd/system/default.target' →
'/usr/lib/systemd/system/graphical.target'.
```

Use the `systemctl` command to verify the default target:

```
root@host:~# systemctl get-default
graphical.target
```

## Use the Emergency and Rescue Targets

By appending either the `systemd.unit=rescue.target` or `systemd.unit=emergency.target` options to the kernel command line from the boot loader, the system enters into a rescue or emergency shell instead of a standard boot process. Both of these shells require the `root` password.

The emergency target mounts the `root` file system in read-only mode. At this point, the `root` user cannot change the `/etc/fstab` file until the drive is remounted in read/write mode by using the `mount -o remount,rw /` command.

In contrast, the rescue target waits for the `sysinit.target` unit to complete, so that more of the system is initialized, such as the logging service or the mounting of file systems with read/write access.

Administrators can use these shells to fix issues that prevent the system from booting as expected, for example, a dependency loop between services, or an incorrect entry in the `/etc/fstab` file. After you exit from these shells, the system continues with the standard boot process.

## Select a Different Target at Boot Time

To select a different target at boot time, append the `systemd.unit=target.target` option to the kernel command line from the boot loader.

For example, to boot the system into a rescue shell where you can change the system configuration with almost no services running, append the following option to the kernel command line from the boot loader:

```
systemd.unit=rescue.target
```

This configuration change affects only a single boot, and is a useful tool to troubleshoot the boot process.

To use this method to select a different target, follow these steps:

1. Boot or reboot the system.
2. In the GRUB2 menu, interrupt the countdown by pressing any key (except **Enter**, which initiates a regular boot).
3. Move the cursor to the kernel entry to start.
4. Press **E** to edit the current entry.
5. Move the cursor to the line that starts with `linux`, which is the kernel command line.
6. Append `systemd.unit=target.target`; for example,  
`systemd.unit=emergency.target`.
7. Press **Ctrl + X** to boot with these changes.

## Power off and Reboot

To power off or reboot a running system from the command line, you can use the `systemctl` command.

The `systemctl poweroff` command stops all running services, unmounts all file systems (or remounts them as read-only when they cannot be unmounted), and then powers down the system.

The `systemctl reboot` command stops all running services, unmounts all file systems, and then reboots the system.

You can also use the shorter version of these commands, `poweroff` and `reboot`, which are symbolic links to the `systemctl` unit equivalents.

### Note

The `systemctl halt` and `halt` commands are also available to stop the system. Unlike the `poweroff` command, these commands do not power off the system; they bring down a system to a point where it is safe to power it off manually.

## References

bootup(7), dracut.bootup(7), lsinitrd(1), systemd.target(5), systemd.special(7),  
sulogin(8), and systemctl(1) man pages

For more information, refer to the *Managing systemd* chapter in the *Using systemd Unit Files to Customize and Optimize Your System* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_systemd\\_unit\\_files\\_to\\_customize\\_and\\_optimize\\_your\\_system/index#managing-systemd](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_systemd_unit_files_to_customize_and_optimize_your_system/index#managing-systemd)

## 12.4. Guided Exercise

### Explore the Boot Process and Select a Boot Target

Change the default boot target, and manually boot the system to a target other than the default.

#### Outcomes

- Update the system default target and use a temporary target from the boot loader.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start boot-selecting
```

#### Instructions

- From a terminal in the workstation web console, confirm that the graphical target is the default boot target.

```
student@workstation:~$ systemctl get-default  
graphical.target
```

- Manually switch to the multi-user target without rebooting. Use the sudo command to escalate the privileges. Use student as the password.

```
student@workstation:~$ sudo systemctl isolate multi-user.target  
[sudo] password for student: student
```

- The system now presents a text-based console instead of a graphical interface. Log in as the root user. Use redhat as the password.

```
workstation login: root  
Password: redhat  
root@workstation:~#
```

- Configure the workstation machine to automatically boot into the multi-user target, and then reboot the workstation machine to verify the configuration. When done, change the systemd

unit target back to the graphical target.

**4.1.** Set the default target to the multi-user target.

```
root@workstation:~# systemctl set-default multi-user.target
Removed '/etc/systemd/system/default.target'.
Created symlink '/etc/systemd/system/default.target' →
'/usr/lib/systemd/system/multi-user.target'.
```

**4.2.** Reboot the workstation machine.

```
root@workstation:~# systemctl reboot
```

**4.3.** After the system reboots, wait for the GRUB2 menu to time out, or press **Enter** to skip the countdown. The system presents a text-based console and not a graphical login screen.

**4.4.** Log in as the **root** user. Use **redhat** as the password.

```
workstation login: root
Password: redhat
root@workstation:~#
```

**4.5.** Set the default systemd unit target back to the graphical target.

```
root@workstation:~# systemctl set-default graphical.target
Removed '/etc/systemd/system/default.target'.
Created symlink '/etc/systemd/system/default.target' →
'/usr/lib/systemd/system/graphical.target'.
```

**5.** The previous step concludes the first part of the exercise, where you practice setting the default systemd target.

In this second part of the exercise, you practice recovering the system by booting into the **rescue** target.

Access the boot loader by rebooting the workstation machine again. From the boot-loader menu, boot into the **rescue** target.

**5.1.** Reboot the workstation machine.

```
root@workstation:~# systemctl reboot
```

- 5.2.** When the boot-loader menu appears, press any key to interrupt the countdown (except **Enter**, which initiates a regular boot).
- 5.3.** Press **Esc** to interrupt the countdown.
- 5.4.** Press **E** to edit the current default entry.
- 5.5.** Using the cursor keys, go to the line that starts with the `linux` text.
- 5.6.** Press **End** or **Ctrl + E** to move the cursor to the end of the line.
- 5.7.** Append the `systemd.unit=rescue.target` option to the end of the line.

#### Note

If the text in the console is difficult to see, then consider changing the resolution when editing the kernel line in the boot-loader entry.

To change the console resolution, add either the `video=640x480` or `vga=ask` parameter on the line that starts with the `linux` word, after the `rd.break` parameter. For most consoles, a resolution of `640x480` is enough. By using the `vga=ask` parameter, you can choose a more suitable resolution for your environment.

- 5.8.** Press **Ctrl + X** to boot by using the modified configuration.
- 5.9.** Log in to rescue mode. You might need to press **Enter** to get a clean prompt. Use `redhat` as the password.

```
Give root password for maintenance
(or press Control-D to continue): redhat
root@workstation:~#
```

- 6.** Confirm that the root file system is in read/write mode.

```
root@workstation:~# mount
...output omitted...
/dev/sda3 on / type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
...output omitted...
```

7. Press **Ctrl + D** to continue with the boot process.

The system boots into graphical mode.

#### Note

In its default configuration, the graphical target boots the system into a graphical login screen where you can log in as any user. In the lab environment, the graphical target is configured to login automatically as the student user.

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish boot-selecting
```

## 12.5. Repairing Damaged File Systems at Boot Time

### Objectives

- Repair file systems that are misconfigured or damaged and that cause the boot process to stop prematurely.

### File-system Issues

During the boot process, the `systemd` unit mounts the persistent file systems that are defined in the `/etc/fstab` file.

Errors in the `/etc/fstab` file or corrupted file systems can block a system from completing the boot process. In some failure scenarios, the system breaks out of the boot process and opens an emergency shell that requires the `root` user's password.

The following list describes some common file-system mounting issues when parsing the `/etc/fstab` file during the boot process:

#### Corrupted file system

The `systemd` unit attempts to repair the file system. If the problem cannot be automatically repaired, then the system opens an emergency shell.

#### Nonexistent storage device, device UUID, or mount point

The `systemd` unit times out waiting for the storage device to become available. If the device does not respond, then the system opens an emergency shell.

### Checking and Repairing File-system Corruption

Red Hat Enterprise Linux 10 provides tools to check and repair file-system issues. In the case of minor issues that are detected during boot, these utilities attempt to repair the file system automatically. For journal file systems, such as XFS and ext4, it might be only necessary to replay the journal, which is usually a quick operation. In the case of more severe file-system corruption issues, a system administrator can use the tools to check and repair file systems manually.

The tool that helps to keep file systems healthy during boot is generically known as the *file system check tool*, or the `fsck` command. However, the tool that repairs the XFS file system does not follow the same naming convention. Although there is an `fsck .xfs` binary file on the system, its only purpose is to satisfy boot requirements and immediately exits with an exit code of 0.

The `xfs_repair` command can check and repair issues with XFS file systems that an automatic repair does not fix.

```
root@host:~# xfs_repair block-device
```

To perform check and repair operations on ext4 file systems, use the `fsck.ext4` command, which is a hard link to the `e2fsck` command that covers legacy ext3 and ext2 file systems as well. Use the `-p` option to automatically repair minor issues without user intervention.

```
root@host:~# fsck.ext4 -p block-device
```

 **Warning**

The proper procedure for repairing a file system is to mount and unmount the file system so that it replays the log files to avoid corrupted or residual data. Additionally, the file system must be unmounted before running the `xfs_repair` or `e2fsck` commands to avoid potential data loss or corruption.

## Repairing File-system Issues at Boot

To access a system that cannot complete booting because of file-system issues, the `systemd` unit architecture provides an emergency boot target, which opens an emergency shell that requires the root password for access.

The following example shows the boot process output when the system finds a file-system issue and switches to the emergency target:

```
...output omitted...
[*      ] A start job is running for /dev/sda2 (27s / 1min 30s)
[ TIME ] Timed out waiting for device /dev/sda2.
[DEPEND] Dependency failed for /mnt/mountfolder
[DEPEND] Dependency failed for Local File Systems.
[DEPEND] Dependency failed for Mark need to relabel after reboot.
...output omitted...
[ OK    ] Started Emergency Shell.
[ OK    ] Reached target Emergency Mode.
...output omitted...
Give root password for maintenance
(or press Control-D to continue):
```

The systemd unit failed to mount the /dev/sda2 device and timed out. Because the device is not available, the system opens an emergency shell for maintenance access.

To repair file-system issues when your system opens an emergency shell, first locate the errant file system, and then find and repair the fault. Now reload the systemd unit configuration to retry the automatic mounting.

Use the mount command to find which file systems are currently mounted by the systemd unit.

```
root@host:~# mount
...output omitted...
/dev/sda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
...output omitted...
```

Depending on which stage of the boot process the system is in when the issue appears, the root file system might still be mounted in read-only mode. If the root file system is mounted with the ro (read-only) option, then you cannot edit the /etc/fstab file. Temporarily remount the root file system with the rw (read/write) option to edit the /etc/fstab file. The remount option changes the mount parameters without unmounting the file system.

```
root@host:~# mount -o remount,rw /
```

Try to mount all the file systems that are listed in the /etc/fstab file by using the mount command --all option. This option processes every file-system entry, but skips those file systems that are already mounted. The command displays any errors that occur when mounting a file system.

```
root@host:~# mount --all
mount: /mnt/mountfolder: mount point does not exist.
```

In this scenario, where the `/mnt/mountfolder` directory does not exist, create the `/mnt/mountfolder` directory before reattempting the mount. Other error messages can occur, including typing errors in the entries, or incorrect device names or UUIDs.

After you correct all issues in the `/etc/fstab` file, inform the `systemd` daemon to register the new `/etc/fstab` file by using the `systemctl daemon-reload` command.

```
root@host:~# systemctl daemon-reload
```

Then, reattempt mounting all the entries.

```
root@host:~# mount --all
```

The `systemd` unit processes the `/etc/fstab` file by transforming each entry into a mount type, and then starting the unit as a service. The `systemctl daemon-reload` command requests the `systemd` unit to rebuild and reload all unit configurations.

If the `mount` command `--all` option succeeds without further errors, then the final test is to verify that the file systems mount successfully during a system boot. Reboot the system and wait for the boot to complete.

```
root@host:~# systemctl reboot
```

For a quick test of the `/etc/fstab` file, use the `nofail` mount entry option. Using the `nofail` option in an `/etc/fstab` entry enables the system to boot even if that file-system mount is unsuccessful.

This option must not be used with production file systems that must always mount. With the `nofail` option, an application could start when its file-system data is missing, with possibly severe consequences.

## References

`xfs_repair(8)`, `fsck(8)`, `e2fsck(8)`, `systemd-fsck(8)`, `systemd-fstab-generator(8)`, and `systemd.mount(5)` man pages

For more information, refer to the *Checking and Repairing a File System* chapter in the *Managing File Systems* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/managing\\_file\\_systems/index#checking-and-repairing-a-file-system](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/managing_file_systems/index#checking-and-repairing-a-file-system)

## 12.6. Guided Exercise

### Repair a Damaged File System at Boot Time

Repair a misconfigured or damaged file system that causes the boot process to stop prematurely.

#### Outcomes

- Diagnose file-system issues and recover the system.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start boot-repairing
```

#### Instructions

- Access the `servera` machine console and reboot the machine. Notice that the boot process does not complete.
  - Reboot the `servera` machine, by sending `Ctrl + Alt + Del` to your system by using the relevant button or menu entry. The boot process fails to complete.
- Open an emergency shell from the boot-loader menu.

##### Note

Alternatively, you can skip this step by waiting for the boot process to complete and for the system to open an emergency shell. If you take this approach, then proceed to Step 3.

- When the boot-loader menu appears, press any key to interrupt the countdown, except the `Enter` key.
- Press `E` to edit the current default entry.
- Use the cursor keys to go to the line that starts with the `linux` text.
- Press `End` or `Ctrl + E` to move the cursor to the end of the line.
- Append the `systemd.unit=emergency.target` string to the end of the line.

 Note

If the text in the console is difficult to see, then consider changing the resolution when editing the kernel line in the boot-loader entry.

To change the console resolution, add either the `video=640x480` or `vga=ask` parameter on the line that starts with the `linux` word, after the `systemd.unit=emergency.target` parameter. For most consoles, a resolution of `640x480` is enough. By using the `vga=ask` parameter, you can choose a more suitable resolution for your environment.

**2.6.** Press `Ctrl + X` to boot by using the modified configuration.

**3.** Log in to the emergency shell. Use `redhat` as the password.

```
...output omitted...
Give root password for maintenance
(or press Control-D to continue): redhat
[root@servera ~]#
```

**4.** Determine which file systems the `systemd` unit currently mounts.

The `systemd` unit mounts the `root` file system in read-only mode.

```
[root@servera ~]# mount
...output omitted...
/dev/sda3 on / type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
...output omitted...
```

**5.** Remount the `root` file system in read/write mode.

```
[root@servera ~]# mount -o remount,rw /
```

**6.** Try to mount all the other file systems. The `mount` command `-a` option mounts all the listed file systems in the `/etc/fstab` file that are not yet mounted.

```
[root@servera ~]# mount -a  
mount: /fakeroot: mount point does not exist.  
      dmesg(1) may have more information after failed mount system call.
```

7. Edit the /etc/fstab file to fix the issue.

**7.1.** Modify the incorrect line from the /etc/fstab file to correctly mount the root file system.  
Replace the /fakeroot mount point with the / mount point.

The device UUID of the root file system might differ on your system.

```
UUID=15507695-22bb-4c65-94e6-a438e095983f      /      xfs      defaults      0  0  
...output omitted...
```

- 7.2. Reload the systemd unit for the system to register the new /etc/fstab file configuration.

```
[root@servera ~]# systemctl daemon-reload
```

8. Verify that the /etc/fstab file is now correct by attempting to mount all entries.

```
[root@servera ~]# mount -a
```

9. Reboot the system and wait for the boot to complete. The system should now boot as expected.

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

10. Verify that you can log in as the student user. Use student as the password.

```
servera login: student  
Password: student  
[student@servera ~]$
```

11. Close the servera machine console.

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish boot-repairing
```

## 12.7. Lab

# Control and Troubleshoot the Boot Process

Manage the booting of a system, to control the services and to repair boot-time problems by troubleshooting the issues.

## Outcomes

- Diagnose and fix boot issues.
- Set the default systemd unit target.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start boot-review
```

## Instructions

1. Open the serverb machine console. Reboot the serverb machine.  
The boot process fails to complete. Fix the boot issue so that the system boots as expected.
2. Change the default systemd unit target on the serverb machine for the system to automatically start a graphical interface when it boots.  
No graphical interface is installed on the serverb machine. Set only the default target, and do not install the packages.
3. After completing the tasks of this activity, return to the workstation machine as the student user.

## Evaluation

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade boot-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish boot-review
```

## Solution

---

Manage the booting of a system, to control the services and to repair boot-time problems by troubleshooting the issues.

## Outcomes

- Diagnose and fix boot issues.
- Set the default systemd unit target.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start boot-review
```

## Instructions

1. Open the `serverb` machine console. Reboot the `serverb` machine.

The boot process fails to complete. Fix the boot issue so that the system boots as expected.

- 1.1. Reboot the `serverb` machine, by sending `Ctrl` + `Alt` + `Del` to your system by using the relevant button or menu entry.

Notice that a start job does not seem to complete and the system eventually opens an emergency shell.

- 1.2. Log in to the emergency shell. Use `redhat` as the password.

```
...output omitted...
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

- 1.3. Check whether the root file system needs to be mounted in read/write mode.

```
[root@serverb ~]# mount  
/dev/sda3 on / type xfs  
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)  
...output omitted...
```

- 1.4.** Try to mount all the file systems. Use the `mount` command `-a` option to mount all the listed file systems in the `/etc/fstab` file that are not yet mounted. The device UUID might be different on your system.

```
[root@serverb ~]# mount -a  
mount: /olddata: can't find UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc.
```

- 1.5.** Edit the `/etc/fstab` file to remove or comment out the incorrect line that mounts the `/olddata` mount point.

```
...output omitted...  
# UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc /olddata xfs defaults 0 0
```

- 1.6.** Reload the `systemd` unit to register the new `/etc/fstab` file configuration.

```
[root@serverb ~]# systemctl daemon-reload
```

- 1.7.** Verify that the `/etc/fstab` file is now correct by attempting to mount all entries.

```
[root@serverb ~]# mount -a
```

- 1.8.** Reboot the `serverb` machine and wait for the boot to complete. The system must now boot as expected.

```
[root@serverb ~]# systemctl reboot
```

- 1.9.** Close the `serverb` machine console.

- 2.** Change the default `systemd` unit target on the `serverb` machine for the system to automatically start a graphical interface when it boots.

No graphical interface is installed on the `serverb` machine. Set only the default target, and do not install the packages.

- 2.1.** From the workstation machine, log in to the `serverb` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

**2.2.** Set the graphical target as the default target.

```
[root@serverb ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target →
/usr/lib/systemd/system/graphical.target.
```

**2.3.** Verify that the graphical target is set.

```
[root@serverb ~]# systemctl get-default
graphical.target
```

- 3.** After completing the tasks of this activity, return to the workstation machine as the student user.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
student@workstation:~$
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade boot-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish boot-review
```

## 12.8. Summary

---

In this lesson, you learned about the Red Hat Enterprise Linux boot process, how to control which services start at boot, and how to diagnose and repair file-system issues during boot time.

The GRUB2 boot loader is the default boot loader for RHEL 10. It is responsible for loading the kernel and the `initramfs` image, and passing control to the kernel. GRUB2 also provides a menu interface to select which boot instance to launch, and to modify boot settings at runtime. You can use the `grubby` command to perform various boot management tasks, such as adding or removing kernel arguments, changing the default boot-menu entry, and viewing information about existing boot-menu entries.

The `systemd` unit takes full control of the last part of the boot sequence, based on the default target. The `systemctl get-default` command provides information about the default `systemd` unit target, and the `set-default` subcommand sets a new default target. Administrators can switch to a different target during runtime by using the `systemctl isolate` command.

The boot process could be interrupted by a file-system issue, such as a corrupted file system or an incorrect `/etc/fstab` file configuration. The `emergency` target boots into a minimal environment that enables troubleshooting. The `xfs_repair` command checks and repairs issues with XFS file systems. The `/etc/fstab` file might list a nonexistent device, giving an error at boot time and interrupting the boot process.

## Chapter 13.

### Recovering Superuser Access

---

#### Goal

Gain administrative access to a system when the superuser password is unknown or is locked.

#### Sections

- Regaining Superuser Access and Resetting the Root Password (and Guided Exercise)

## 13.1. Regaining Superuser Access and Resetting the Root Password

### Objectives

- Gain administrative access to a system when the superuser password is unknown or is locked.

### Resetting the root Password from the Boot Loader

One task that every system administrator should be able to accomplish is resetting a root user's lost password. This task is trivial if the administrator is still logged in, either as the root user or as an unprivileged user with full Sudo access. However, this task is a little more complex when the administrator is not logged in because it requires rebooting the system into a special state known as a *rescue environment* or *rescue mode*.

In Red Hat Enterprise Linux 10, you can enter rescue mode to change a forgotten root password by using one of two methods:

- With rescue media (recommended)
- Without rescue media

#### With Rescue Media

The recommended approach is to use rescue media, which enables you to boot a small Red Hat Enterprise Linux environment from installation media, such as a CD-ROM or an ISO file, instead of from the system's hard drive.

To boot into rescue mode, you must be able to use one of the following installation media devices or files to boot the system:

- For a physical machine, use installation media such as a CD-ROM or a USB flash device.
- For a virtual machine, use ISO files.

When using rescue media, be sure that the image is the same as the system. For example, for a RHEL 10 system, you must use a RHEL 10 image.

As part of the network installation media, Red Hat provides a minimum installation image: Red Hat Enterprise Linux 10.0 Boot ISO. You can use this boot ISO image as rescue media to reboot the system and reset the root password.

To use this image as rescue media, follow these steps:

1. Reboot the system.
2. During the system reboot, select the boot .iso image to boot from. The process to select alternate media to boot from varies based on your environment.

3. From the GRUB2 menu, go to **Troubleshooting > Rescue a Red Hat Enterprise Linux system.**

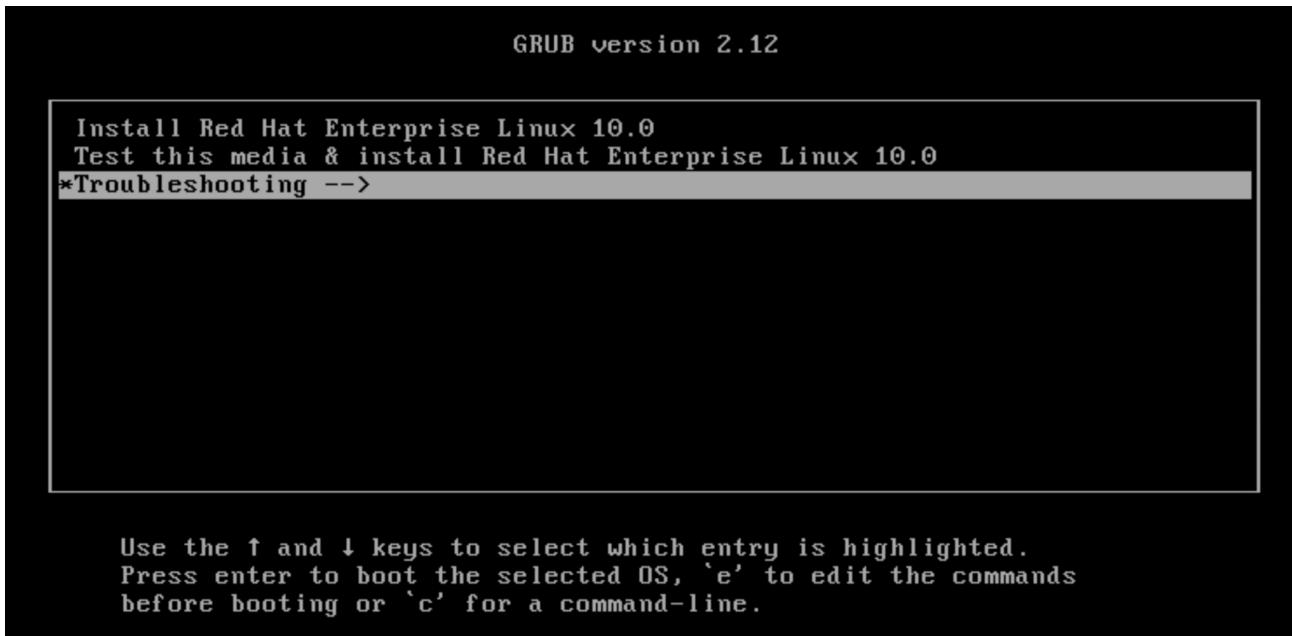


Figure 14. The GRUB2 menu

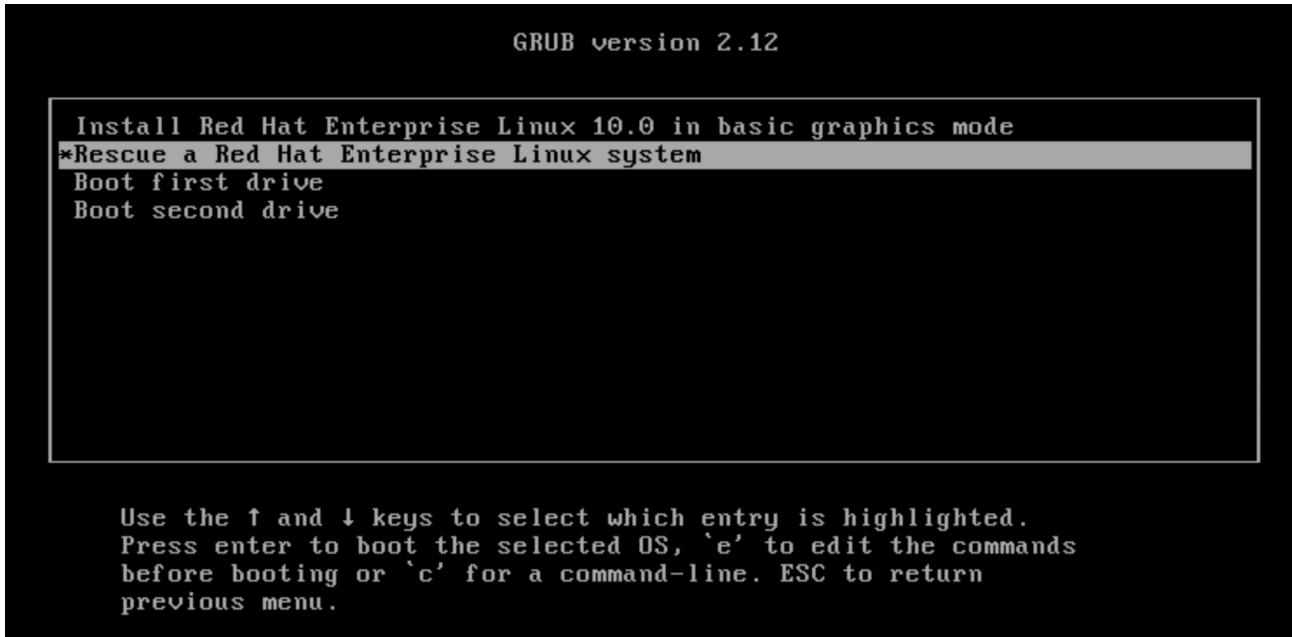


Figure 15. Selecting rescue mode

4. In the rescue environment menu, select Continue by pressing **1** and then press **Enter** to mount the system normally.

```
Starting installer, one moment...
libreport is not available in this environment - bug reporting disabled
anaconda 40.22.3.26-1.el10_0 for Red Hat Enterprise Linux 10.0 started.
* installation log files are stored in /tmp during the installation
* shell is available on TTY2 and in second TMUX pane (ctrl+b, then press 2)
* when reporting a bug add logs from /tmp as separate text/plain attachments
=====
=====
Rescue

The rescue environment will now attempt to find your Linux installation and
mount it under the directory : /mnt/sysroot. You can then make any changes
required to your system. Choose '1' to proceed with this step.
You can choose to mount your file systems read-only instead of read-write by
choosing '2'.
If for some reason this process does not work choose '3' to skip directly to a
shell.

1) Continue
2) Read-only mount
3) Skip to shell
4) Quit (Reboot)

Please make a selection from the above: 1
```

Figure 16. Entering rescue mode

- At the shell prompt, change the root directory to the /mnt/sysroot directory.

```
bash-5.2# chroot /mnt/sysroot
```

- Change the root password to the desired password.

```
bash-5.2# passwd root
New password: provide_password
Retype new password: provide_password
passwd: password updated successfully
```

- Ensure that all unlabeled files, including the /etc/shadow file at this point, is relabeled during boot.

### Important

Because the system has not yet enabled SELinux, any file that you create is not labeled with an SELinux context. Some tools, such as the passwd command, first create a temporary file, and then replace it with the file that is intended for editing, which effectively creates a file without an SELinux context. For this reason, when you use the passwd command in rescue mode, the /etc/shadow file is not labeled with an SELinux context.

```
bash-5.2# touch /.autorelabel
```

8. Type the `exit` command twice to reboot your system. The first command exits the `chroot` command, and the second command exits rescue mode and reboots the system.

```
bash-5.2# exit  
exit  
bash-5.2# exit
```

At this point, the system continues booting, performs a full SELinux relabeling, and then reboots again.

## Without Rescue Media

As an alternative to using rescue media, you can also access the rescue environment by configuring the system to emulate rescue mode. To achieve this system state, pass the `init /bin/bash` option to the kernel, and then remount the `root` file-system in read/write mode.

### Warning

This method is risky and must be performed with caution.

1. Reboot the system.
2. Interrupt the boot-loader countdown by pressing `Esc`.
3. Boot with the `init=/bin/bash` argument.
  - a. In the GRUB2 menu, move the cursor to the kernel entry to boot.
  - b. Press `E` to edit the selected entry.
  - c. Move the cursor to the kernel command line (the line that starts with the `linux` text).
  - d. Remove any `console=` options from the line.
  - e. Move to the end of the line by pressing `Ctrl + E`.
  - f. Add the string `init=/bin/bash` to the end after space.
  - g. Press `Ctrl + X` to boot with the changes.
4. Wait for the system to finish booting.
5. The `root (/)` file system is mounted as read-only. Changing the `root` password requires the `root` file system to be mounted as read/write. Remount the `root` file system in read/write mode:

```
bash-5.2# mount -o remount,rw /
```

6. Set a new `root` password.

```
bash-5.2# passwd
```

7. Create the hidden `.autorelabel` file to relabel the SELinux context in all files at the next system boot.

```
bash-5.2# touch /.autorelabel
```

8. Reboot the system.

```
bash-5.2# exec /sbin/init
```

The system continues booting, performs a full SELinux relabeling, and then reboots again.

### References

`dracut cmdline(7)` man page.

[How to Change a Forgotten or Lost Root Password?](#)

## 13.2. Guided Exercise

# Regain Superuser Access and Reset the Root Password

Gain administrative access to a system when the superuser password is unknown or is locked.

## Outcomes

- Reset the root user's password.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start rootpw-recover
```

## Instructions

- The password for the root user on the servera machine must be redhat.

Locate the servera machine console and try to log in as the root user. Use redhat as the password.

```
servera login: root  
Password: redhat  
Login incorrect
```

- Reset the root user's password to redhat without using rescue media. First, reboot the servera machine, and interrupt the countdown in the boot-loader menu.

- Locate the servera console, as appropriate for your classroom environment, and then open the console.

Send **Ctrl** + **Alt** + **Del** to your system by using the relevant button or menu entry.

- When the boot-loader menu appears, press **Esc** to interrupt the countdown.

- Edit the highlighted kernel boot-loader entry to stop the boot process just after the kernel mounts all the file systems, but before it passes control to the systemd unit.

- Use the cursor keys to select the kernel entry.

**3.2.** Press **E** to edit the current entry.

**3.3.** Move the cursor to the line that starts with the `linux` text.

**3.4.** Remove any `console=` option from the line.

 **Important**

If you do not remove the `console=` options from the GRUB2 entry for the rescue kernel, then the `root` prompt might start on the wrong console and you might not be able to access it.

**3.5.** Press **Ctrl + E** to move the cursor to the end of the line.

**3.6.** Append a space followed by the `init=/bin/bash` option to the end of the line.

 **Note**

If the text in the console is difficult to see, then consider changing the resolution when editing the kernel line in the boot-loader entry.

To change the console resolution, add either the `video=640x480` or `vga=ask` parameter on the line that starts with the `linux` word, after the `rd.break` parameter. For most consoles, a resolution of `640x480` is enough. By using the `vga=ask` parameter, you can choose a more suitable resolution for your environment.

**3.7.** Press **Ctrl + X** to boot by using the modified configuration.

**4.** Wait for the system to finish booting. At the `bash-5.2#` prompt, remount the `root (/)` file system as read/write.

```
bash-5.2# mount -o remount,rw /
```

**5.** Change the `root` password to `redhat`.

```
bash-5.2# passwd
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: password updated successfully
```

6. Configure the system to automatically perform a full SELinux relabeling after booting. This step is necessary because the `passwd` command recreates the `/etc/shadow` file without an SELinux context.

```
bash-5.2# touch /.autorelabel
```

7. Reboot the servera machine.

The system runs an SELinux relabel operation, and then reboots automatically. Wait for the servera machine to boot.

```
bash-5.2# exec /sbin/init
```

8. When the system is ready, verify that you can log in as the `root` user with `redhat` as the password.

```
servera login: root
Password: redhat
...output omitted...
[root@servera ~]#
```

9. Close the servera console.

## Finish

- On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish rootpw-recover
```

## 13.3. Summary

---

In this lesson, you learned to recover or reset the root user's password.

You can use the boot.iso image as the rescue media to reset the root password. Boot into rescue mode, and then mount the system in read/write mode. At the shell prompt, change the root directory to the /mnt/sysroot directory and set the root user's password.

You can also recover or reset the root password without using rescue media. Pass the init /bin/bash option to the kernel, and then remount the root file system in read/write mode to change the root password. Relabel all files on the next reboot to set the correct SELinux context.

## Chapter 14.

# Managing Network Security

---

### Goal

Control network connections to services by using the system firewall, and network services that can bind to particular ports by using SELinux.

### Sections

- Managing Server Firewalls (and Guided Exercise)
- SELinux Port Labeling (and Guided Exercise)

### Lab

- Manage Network Security

## 14.1. Managing Server Firewalls

### Objectives

- Accept or reject connections to local network services by using the `firewalld` service.

### Linux Firewall Architecture Concepts

The Linux kernel provides the `netfilter` framework for network traffic operations such as packet filtering, network address translation, and port translation. The `nftables` packet classification framework builds on the `netfilter` framework to apply firewall rules to network traffic.

In Red Hat Enterprise Linux 10, the `nftables` framework is the system firewall core.

#### Note

The `nftables` framework is the successor to the `iptables` and `ip6tables` utilities, among others. It offers many improvements in convenience, features, and performance over the previous packet-filtering tools. You can convert the earlier `iptables` configuration files to their `nftables` equivalents by using the `iptables-translate` and `ip6tables-translate` utilities.

### The Firewall Service

The `firewalld` service is a dynamic firewall manager, and is the recommended front end to the `nftables` framework. Red Hat Enterprise Linux 10 includes the `firewalld` package.

The `firewalld` service simplifies firewall management by classifying network traffic into zones. A network packet's assigned zone depends on criteria such as the source IP address of the packet or the incoming network interface. Each zone has its own list of ports and services that are either open or closed.

#### Note

For machines that often change networks, the `NetworkManager` service can automatically set the firewall zone for a connection. This service is useful when switching between home, work, or public wireless networks. A user might want their system's `sshd` service to be reachable when connected to their home or corporate networks, but not when connected to a public wireless network in the local coffee shop.

The firewalld service inspects the source address for every incoming packet into the system. If the source address is assigned to a specific zone, then the rules for that zone apply. If the source address is not assigned to a zone, then the firewalld service associates the packet with the zone for the incoming network interface, and the rules for that zone apply. If the network interface is not associated with a zone, then the firewalld service sends the packet to the default zone.

The default zone is not a separate zone; you configure one of the existing zones to be assigned as the default zone. Initially, the firewalld service designates the public zone as the default, and maps the `lo` loopback interface to the trusted zone.

Most zones allow traffic through the firewall if the local destination of the traffic matches a list of ports and protocols, such as 631/udp, or a predefined service configuration, such as the `ssh` service.

Normally, if the traffic does not match a permitted port and protocol or service, then it is rejected. An exception is the trusted zone, which permits all traffic by default.

## Predefined Zones

The firewalld service uses predefined zones, which you can customize. By default, all zones allow any incoming traffic that is part of an existing session that the system initiated, and also allow all outgoing traffic.

The following table shows the initial zone configuration:

**Table 17. Default Configuration of Firewall Zones**

Zone name	Default configuration
trusted	Allow all incoming traffic.
home	Reject incoming traffic unless related to outgoing traffic or matching the <code>ssh</code> , <code>mdns</code> , <code>ipp-client</code> , <code>samba-client</code> , or <code>dhcpcv6-client</code> predefined services.
internal	Reject incoming traffic unless related to outgoing traffic or matching the <code>ssh</code> , <code>mdns</code> , <code>ipp-client</code> , <code>samba-client</code> , or <code>dhcpcv6-client</code> predefined services (same as the home zone to start with).

Zone name	Default configuration
work	Reject incoming traffic unless related to outgoing traffic or matching the ssh, ipp-client, or dhcpcv6-client predefined services.
public	Reject incoming traffic unless related to outgoing traffic or matching the ssh or dhcpcv6-client predefined services. The default zone for newly added network interfaces.
external	Reject incoming traffic unless related to outgoing traffic or matching the ssh predefined service. Outgoing IPv4 traffic that is forwarded through this zone is <i>masqueraded</i> to appear that it originated from the IPv4 address of the outgoing network interface.
dmz	Reject incoming traffic unless related to outgoing traffic or matching the ssh predefined service.
block	Reject all incoming traffic unless related to outgoing traffic.
drop	Drop all incoming traffic unless related to outgoing traffic (do not even respond with ICMP errors).

For a list of available predefined zones and their intended use, see the `firewalld.zones(5)` man page.

## Predefined Services

The `firewalld` service includes predefined configurations for commonly used services, to simplify setting firewall rules.

For example, instead of researching the relevant ports for an NFS server, use the predefined `nfs` configuration to create rules for the correct ports and protocols.

The following table lists some predefined service configurations that might be active in your default `firewalld` zone:

**Table 18. Selected Predefined Firewall Services**

<b>Service name</b>	<b>Configuration</b>
ssh	Local SSH server. Traffic to 22/tcp.
dhcpv6-client	Local DHCPv6 client. Traffic to 546/udp on the fe80::/64 IPv6 network.
ipp-client	Local Internet Printing Protocol (IPP) printing. Traffic to 631/udp.
samba-client	Local Windows file and print sharing client. Traffic to 137/udp and 138/udp.
mdns	Multicast DNS (mDNS) local-link name resolution. Traffic to 5353/udp to the 224.0.0.251 (IPv4) or ff02::fb (IPv6) multicast addresses.
cockpit	Red Hat Enterprise Linux web-based interface for managing and monitoring your local and remote systems. Traffic to 9090/tcp.

The `firewalld` package includes many predefined service configurations. You can list the services with the `firewall-cmd --get-services` option.

```
root@host:~# firewall-cmd --get-services
0-AD RH-Satellite-6 RH-Satellite-6-capsule afp alvr amanda-client amanda-k5-
client amqp amqps anno-1602 anno-1800 apcupsd aseqnet audit ausweisapp2 bacula
bacula-client bareos-director bareos-filedaemon bareos-storage bb bgp bitcoin
bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc bittorrent-lsd ceph ceph-
exporter ceph-mon cfengine checkmk-agent civilization-iv civilization-v cockpit
collectd
...output omitted...
```

If the predefined service configurations are not appropriate for your scenario, then you can manually specify the required ports and protocols. You can use the web console graphical interface to review predefined services and manually define more ports and protocols.

## Configure the Firewall Daemon

The following list shows two ways for system administrators to interact with the `firewalld` service:

- The web console graphical interface
- The `firewall-cmd` command-line tool

## Configure Firewall Services with the Web Console

To manage firewall services with the web console, you must log in and escalate privileges, by clicking the **Limited access** or **Turn on administrative access** buttons.

Then, enter your password when prompted. The administrative mode elevates privileges based on your user's sudo configuration. Remember to toggle back to limited access mode after you perform the system task that requires administrative privileges.

Click the Networking option in the left navigation menu to display the **Firewall** section on the main networking page. Click the **Edit rules and zones** button to go to the Firewall page.

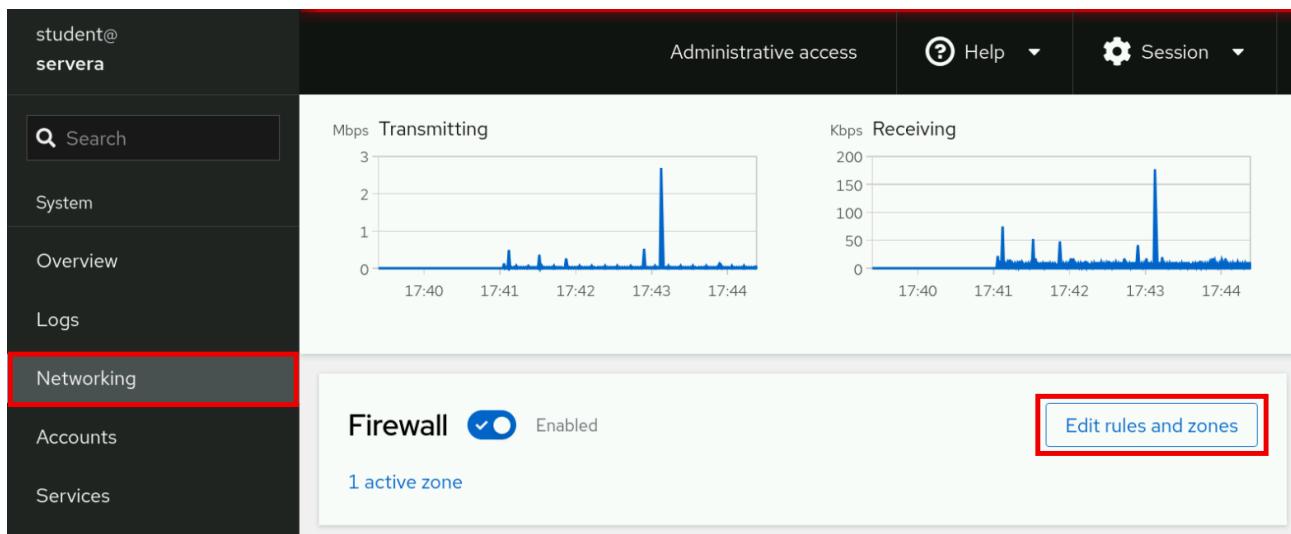


Figure 17. The web console networking page

The Firewall page displays active zones and their allowed services. Click the arrow icon ( ) to the left of a service name to view its details. To add a service to a zone, click the **Add services** button in the upper right corner of the applicable zone.

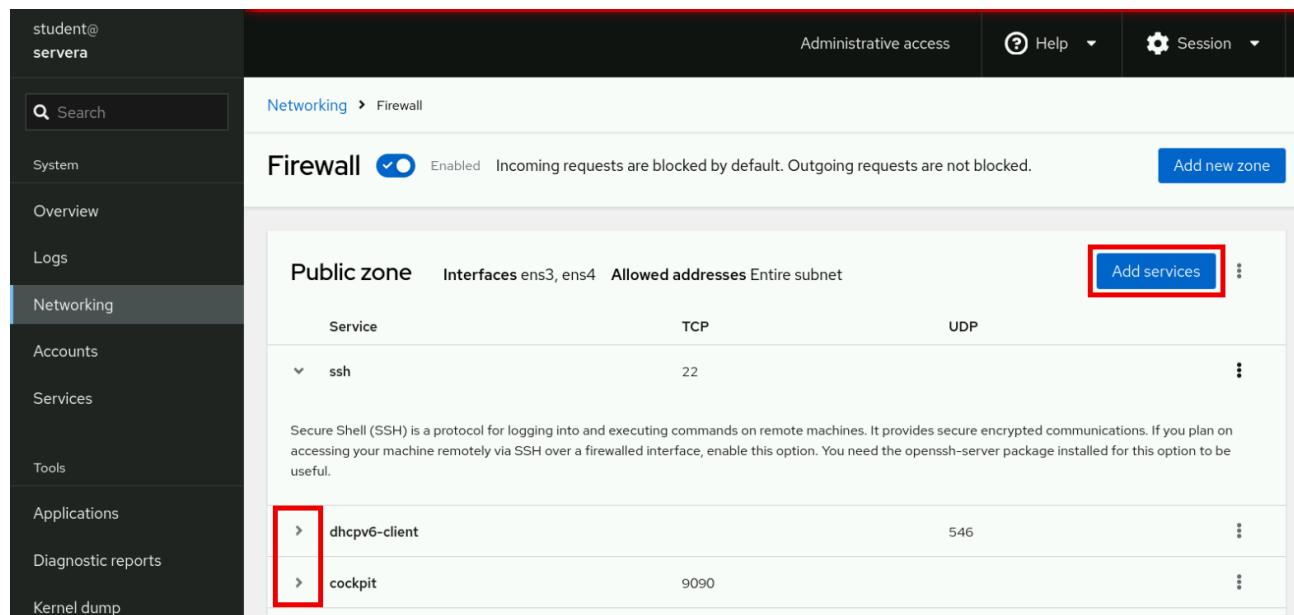


Figure 18. The web console firewall page

The **Add Services** page displays the available predefined services.

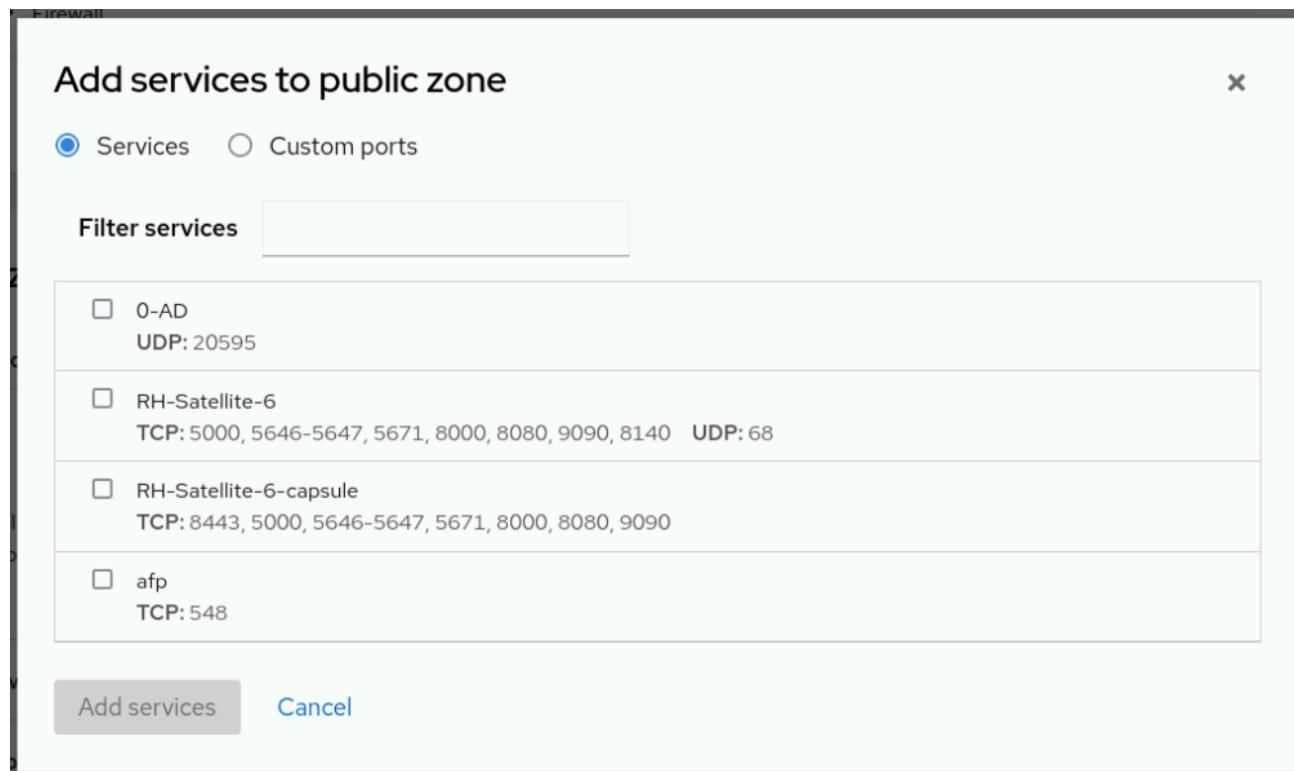


Figure 19. The web console add services menu

To select a service, scroll through the list or enter a selection in the **Filter services** text box.

In the following example, the `http` string filters the options to show web-related services. Select the checkbox to the left of the service to allow it through the firewall. Click the **Add services** button to complete the process.

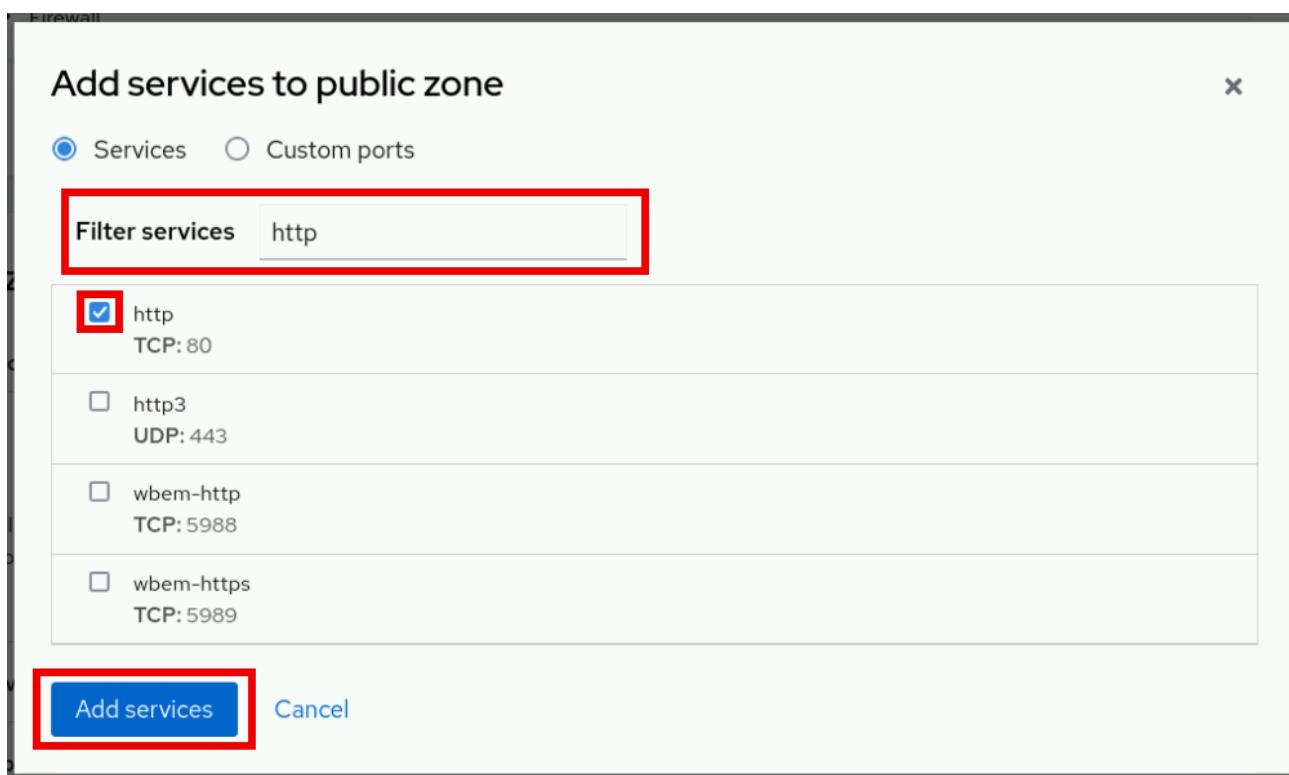


Figure 20. The web console add services menu options

The interface returns to the Firewall page, where you can review the updated allowed services list.

The screenshot shows the 'Networking > Firewall' page. On the left is a sidebar with 'student@servera' and links for 'Search', 'System', 'Overview', 'Logs', 'Networking' (which is selected and highlighted in blue), 'Accounts', 'Services', 'Tools', 'Applications', and 'Diagnostic reports'. The main area has a header with 'Administrative access', 'Help', 'Session', and a 'Firewall' toggle switch that is turned on. Below the header, it says 'Incoming requests are blocked by default. Outgoing requests are not blocked.' and has a 'Add new zone' button. The main content area is titled 'Public zone' and shows a table of allowed services. The table has columns for 'Service', 'TCP', and 'UDP'. Services listed include 'ssh' (TCP 22), 'dhcpv6-client' (UDP 546), 'cockpit' (TCP 9090), and 'http' (TCP 80). The 'http' row is highlighted with a red box. There is also a 'Add services' button and a three-dot menu icon.

Figure 21. The web console firewall overview

## Configure the Firewall from the Command Line

The `firewall-cmd` command interacts with the `firewalld` daemon.

It is installed as part of the `firewalld` package, and is available for administrators who prefer to work on the command line, or for working on systems without a graphical environment, or for scripting a

firewall setup.

Most commands work on the runtime configuration, unless the `--permanent` option is specified. If the `--permanent` option is specified, then you must activate the setting by also running the `firewall-cmd` command `--reload` option, which applies the current permanent configuration as the new runtime configuration. Many of the listed commands take the `--zone=ZONE` option to find which zone they affect. Where a netmask is required, use Classless Inter-Domain Routing (CIDR) notation, such as `192.168.1/24`.

The following table lists some `firewall-cmd` commands:

firewall-cmd commands	Explanation
<code>--get-default-zone</code>	Query the current default zone.
<code>--set-default-zone=ZONE</code>	Set the default zone, which changes both the runtime and the permanent configuration.
<code>--get-zones</code>	List all available zones.
<code>--get-active-zones</code>	List all zones in use (with an interface or source that is tied to them), along with their interface and source information.
<code>--add-source=CIDR [--zone=ZONE]</code>	Route all traffic from the IP address or from the network with their netmask to the specified zone. If no <code>--zone=</code> option is provided, then the default zone is used.
<code>--remove-source=CIDR [--zone=ZONE]</code>	Remove the rule that routes all traffic from the zone that comes from the IP address or network. If no <code>--zone=</code> option is provided, then the default zone is used.

firewall-cmd commands	Explanation
--add-interface=INTERFACE [ --zone=ZONE]	Route all traffic from <i>INTERFACE</i> to the specified zone. If no <i>--zone=</i> option is provided, then the default zone is used.
--change-interface=INTERFACE [ --zone=ZONE]	Associate the interface with <i>ZONE</i> instead of its current zone. If no <i>--zone=</i> option is provided, then the default zone is used.
--list-all [ --zone=ZONE]	List all configured interfaces, sources, services, and ports for <i>ZONE</i> . If no <i>--zone=</i> option is provided, then the default zone is used.
--list-all-zones	Retrieve all information for all zones (interfaces, sources, ports, and services).
--add-service=SERVICE [ --zone=ZONE]	Allow traffic to <i>SERVICE</i> . If no <i>--zone=</i> option is provided, then the default zone is used.
--add-port=PORT/PROTOCOL [ --zone=ZONE]	Allow traffic to the <i>PORT/PROTOCOL</i> ports. If no <i>--zone=</i> option is provided, then the default zone is used.
--remove-service=SERVICE [ --zone=ZONE]	Remove <i>SERVICE</i> from the allowed list for the zone. If no <i>--zone=</i> option is provided, then the default zone is used.
--remove-port=PORT/PROTOCOL [ --zone=ZONE]	Remove the <i>PORT/PROTOCOL</i> ports from the allowed list for the zone. If no <i>--zone=</i> option is provided, then the default zone is used.

firewall-cmd commands	Explanation
--reload	Drop the runtime configuration and apply the permanent configuration.
--runtime-to-permanent	Create permanent rules from the runtime configuration.

The following example sets the default zone to dmz:

```
root@host:~# firewall-cmd --set-default-zone=dmz
```

Assign all traffic from the 192.168.0.0/24 network to the internal zone:

```
root@host:~# firewall-cmd --permanent --zone=internal --add-source=192.168.0.0/24
```

Open the network ports for the mysql service on the internal zone:

```
root@host:~# firewall-cmd --permanent --zone=internal --add-service=mysql
```

Update the changes to the firewalld daemon:

```
root@host:~# firewall-cmd --reload
```

As another example, to add all the incoming traffic from the 172.25.25.11 single IPv4 address to the public zone, use the following commands:

```
root@host:~# firewall-cmd --permanent --zone=public --add-source=172.25.25.11/32
```

Update the changes to the firewalld daemon:

```
root@host:~# firewall-cmd --reload
```

### Note

For situations where the basic syntax is not enough, you can add *rich-rules* to write complex rules. If even the rich-rules syntax is not enough, then you can use Direct Configuration rules (which use a mixture of raw nft syntax and firewalld rules). These advanced configurations are beyond the scope of this lesson.

### References

`firewall-cmd(1)`, `firewalld(1)`, `firewalld.zone(5)`, `firewalld.zones(5)`, and `nft(8)` man pages

For more information, refer to the *Configuring Firewalls and Packet Filters* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/configuring\\_firewalls\\_and\\_packet\\_filters/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/configuring_firewalls_and_packet_filters/index)

## 14.2. Guided Exercise

### Manage a Server Firewall

Accept or reject connections to local network services by using the `firewalld` service.

#### Outcomes

- Configure firewall rules to control access to services.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start netsecurity-firewalls
```

#### Instructions

- Log in to the `servera` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Install the `httpd` and `mod_ssl` packages. These packages provide the Apache web server and the necessary extensions for the web server to serve content over SSL.

```
[root@servera ~]# dnf install httpd mod_ssl
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- Create the `/var/www/html/index.html` file. Add one line of text: `I am servera.`

```
[root@servera ~]# echo 'I am servera.' > /var/www/html/index.html
```

4. Start and enable the httpd service.

```
[root@servera ~]# systemctl enable --now httpd
Created symlink '/etc/systemd/system/multi-user.target.wants/httpd.service' →
'/usr/lib/systemd/system/httpd.service'.
```

5. Open another terminal window to log in as the student user on the workstation machine. Try to access the web server that is running on the servera machine by using both the 80/TCP clear-text port and the 443/TCP SSL encapsulated port. Accessing the web server fails from both ports.

- 5.1. Use the curl command, which fails to access the web server from the 80/TCP port.

```
student@workstation:~$ curl http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80 after 3 ms: Could
not connect to server
```

- 5.2. Use the curl command -k option, for avoiding strict host checking; it fails through the 443/TCP port.

```
student@workstation:~$ curl -k https://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 443 after 2 ms:
Could not connect to server
```

6. On the servera machine, verify that the firewalld service is enabled and running.

```
[root@servera ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled;
  preset: enabled)
    Active: active (running) since Wed 2025-07-09 11:55:22 UTC; 5min ago
      Invocation: 45bec9233c7b4294a4315f735ef115c8
        Docs: man:firewalld(1)
  ...output omitted...
```

7. On the servera machine, add the https service to the public firewall zone.

- 7.1. Verify that the default firewall zone is set to the public zone.

```
[root@servera ~]# firewall-cmd --get-default-zone  
public
```

- 7.2.** If the earlier step does not return public as the default zone, then correct it with the following command:

```
[root@servera ~]# firewall-cmd --set-default-zone public  
success
```

- 7.3.** Add the https service to the permanent configuration for the public network zone. Confirm your configuration.

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

- 7.4.** Update the firewalld service with the configuration.

```
[root@servera ~]# firewall-cmd --reload  
success
```

- 7.5.** Verify that the https service is added to the public zone.

```
[root@servera ~]# firewall-cmd --permanent --zone=public --list-all  
public (default)  
  target: default  
  ingress-priority: 0  
  egress-priority: 0  
  icmp-block-inversion: no  
  interfaces:  
  sources:  
    services: cockpit dhcpcv6-client https ssh  
  ports:  
  protocols:  
  forward: yes  
  masquerade: no  
  forward-ports:  
  source-ports:  
  icmp-blocks:  
  rich rules:
```

- 8.** Return to the workstation machine as the student user.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

9. Verify that you can access the web server that is running on the servera machine through the 443/TCP port.

- 9.1. Try to access web server by using the `http://servera.lab.example.com` URL. The `curl` command still fails because it references the 80/TCP port.

```
student@workstation:~$ curl http://servera.lab.example.com  
curl: (7) Failed to connect to servera.lab.example.com port 80 after 2 ms: Could  
not connect to server
```

- 9.2. Verify that you can access the web server by using the `https://servera.lab.example.com` URL through the 443/TCP port without strict host checking. The `curl` command must succeed.

```
student@workstation:~$ curl -k https://servera.lab.example.com  
I am servera.
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish netsecurity-firewalls
```

## 14.3. SELinux Port Labeling

### Objectives

- Verify and adjust SELinux labels on specific network ports so that permitted network services can bind to those ports to listen for client connections.

### SELinux Port Labeling

In addition to file context and process type labeling, SELinux labels network ports with an SELinux context. SELinux controls network access by labeling the network ports and including rules in a service's targeted policy.

For example, the SSH targeted policy includes the 22/TCP port with the `ssh_port_t` context label. In the HTTP policy, the default 80/TCP and 443/TCP ports use the `http_port_t` context label.

When a targeted process attempts to open a port for listening, SELinux verifies that the policy includes entries that enable the binding of the process and the context. SELinux can then block a rogue service from taking over ports that other legitimate network services use.

### Manage SELinux Port Labeling

If a service attempts to listen on a nonstandard port, and the port is not labeled with the correct SELinux type, then SELinux might block the attempt. You can correct this problem by changing the SELinux context on the port.

Typically, the targeted policy already labels all expected ports with the correct type.

For example, because port 8008/TCP is often used for web applications, that port is already labeled with the `http_port_t` context, which is the default port type for a web server. Individual ports can be labeled with only one port context.

### List Port Labels

Use the `grep` command to filter the port number.

```
root@host:~# grep gopher /etc/services
gopher      70/tcp                                # Internet Gopher
gopher      70/udp
```

Use the `semanage` command to list the current port label assignments.

```
root@host:~# semanage port -l
...output omitted...
http_cache_port_t      tcp    8080, 8118, 8123, 10001-10010
http_cache_port_t      udp    3130
http_port_t            tcp    80, 81, 443, 488, 8008, 8009, 8443, 9000
...output omitted...
```

Use the grep command to filter the SELinux port label by using the service name.

```
root@host:~# semanage port -l | grep ftp
ftp_data_port_t        tcp    20
ftp_port_t             tcp    21, 989, 990
ftp_port_t             udp    989, 990
tftp_port_t            udp    69
```

A port label can appear in the list many times for each supported networking protocol.

Use the grep command to filter the SELinux port label by port number.

```
root@host:~# semanage port -l | grep -w 70
gopher_port_t          tcp    70
gopher_port_t          udp    70
```

## Manage Port Bindings

Use the semanage command to assign new port labels, remove port labels, or modify existing ones.

### Important

Most services in RHEL provide an SELinux policy module, which includes the service default port contexts. You cannot change default port labels by using the semanage command. Instead, you must modify and reload the targeted service policy module. Writing and generating policy modules are beyond the scope of this lesson.

You can label a new port with an existing port context label.

The semanage port command -a option adds a new port label; the -t option denotes the type; and the -p option denotes the protocol.

```
root@host:~# semanage port -a -t port_label -p tcp/udp PORTNUMBER
```

In the following example, enable the gopher service to listen on the 71/TCP port:

```
root@host:~# semanage port -a -t gopher_port_t -p tcp 71
```

To view changes to the default policy, use the semanage port command -C option.

```
root@host:~# semanage port -l -C
SELinux Port Type          Proto    Port Number
gopher_port_t                tcp        71
```

The targeted policies include many port types.

Service-specific SELinux man pages are named by using the service name plus \_selinux. These man pages include service-specific information about SELinux types, Booleans, and port types, and are not installed by default.

To view a list of all the available SELinux man pages, install the package.

```
root@host:~# dnf -y install selinux-policy-doc
```

Use the man command -k option to search for the \_selinux string.

```
root@host:~# man -k _selinux
```

Use the semanage command -d option to delete a port label.

In the following example, remove the binding of port 71/TCP to the gopher\_port\_t type:

```
root@host:~# semanage port -d -t gopher_port_t -p tcp 71
```

To change a port binding, when requirements change, use the -m option. This option is more efficient than deleting the earlier binding and adding the latest one.

In the following example, modify the 71/TCP port label from the gopher\_port\_t to the http\_port\_t context label:

```
root@host:~# semanage port -m -t http_port_t -p tcp 71
```

Verify the changes by using the semanage command.

```
root@host:~# semanage port -l -C
SELinux Port Type          Proto    Port Number

http_port_t                tcp      71
root@host:~# semanage port -l | grep http
http_cache_port_t          tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t          udp      3130
http_port_t                tcp      71, 80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t        tcp      5988
pegasus_https_port_t       tcp      5989
```

## References

semanage(8), semanage-port(8), and \*\_selinux(8) man pages

## 14.4. Guided Exercise

### SELinux Port Labeling

Verify and adjust SELinux labels on specific network ports so that permitted network services can bind to those ports to listen for client connections.

#### Outcomes

- Configure a web server to host content by using a nonstandard port.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start netsecurity-ports
```

#### Instructions

Your organization is deploying a new custom web application, which is running on a nonstandard port, the 82/TCP port.

A junior administrator already configured the application on the `servera` machine. However, the web server content is not accessible. Resolve the web server access issue.

- Log in to the `servera` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- Try to fix the web content problem by restarting the `httpd` service.

- Restart the `httpd` service. This command fails.

```
[root@servera ~]# systemctl restart httpd
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for
details.
```

- 2.2.** View the status of the httpd service. Notice the permission denied error. Press **Q** to exit the session.

```
[root@servera ~]# systemctl status -l httpd
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset:
disabled)
     Active: failed (Result: exit-code) since Sun 2025-06-29 20:27:35 UTC; 56s
ago
   Invocation: 70d8f71da2694aba94b4f0a2612d9e2e
     Docs: man:httpd.service(8)
   Process: 5998 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
   Main PID: 5998 (code=exited, status=1/FAILURE)
     Status: "Reading configuration..."
   Mem peak: 3.8M
     CPU: 54ms

Jun 29 20:27:35 servera systemd[1]: Starting httpd.service - The Apache HTTP
Server...
Jun 29 20:27:35 servera (httpd)[5998]: httpd.service: Referenced but unset
environment variable evaluates to an empty string: OPTIONS
Jun 29 20:27:35 servera httpd[5998]: (13)Permission denied: AH00072: make_sock:
could not bind to address [::]:82
Jun 29 20:27:35 servera httpd[5998]: (13)Permission denied: AH00072: make_sock:
could not bind to address 0.0.0.0:82
Jun 29 20:27:35 servera httpd[5998]: no listening sockets available, shutting
down
Jun 29 20:27:35 servera httpd[5998]: AH00015: Unable to open logs
Jun 29 20:27:35 servera systemd[1]: httpd.service: Main process exited,
code=exited, status=1/FAILURE
Jun 29 20:27:35 servera systemd[1]: httpd.service: Failed with result 'exit-
code'.
Jun 29 20:27:35 servera systemd[1]: Failed to start httpd.service - The Apache
HTTP Server.
```

- 2.3.** Verify whether SELinux is blocking the httpd service from binding to the 82/TCP port.

```
[root@servera ~]# sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
-----
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket
port 82.

***** Plugin bind_ports (99.5 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 82
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 82
    where PORT_TYPE is one of the following: http_cache_port_t, http_port_t,
jboss_management_port_t, jboss.messaging_port_t, ntop_port_t, puppet_port_t.
...output omitted...
Raw Audit Messages
type=AVC msg=audit(1751228855.940:1271): avc: denied { name_bind } for
pid=5998 comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
...output omitted...
```

3. Configure SELinux to allow the httpd service to bind to the 82/TCP port, and then restart the httpd service.

### 3.1. Find an appropriate port type for the 82/TCP port.

The `http_port_t` context label includes the default 80/TCP and 443/TCP HTTP ports. This type is the correct port type for the web server.

```
[root@servera ~]# semanage port -l | grep http
http_cache_port_t          tcp    8080, 8118, 8123, 10001-10010
http_cache_port_t          udp    3130
http_port_t                tcp    80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t        tcp    5988
pegasus_https_port_t       tcp    5989
```

### 3.2. Assign the 82/TCP port with the `http_port_t` context label.

```
[root@servera ~]# semanage port -a -t http_port_t -p tcp 82
```

### 3.3. Restart the httpd service.

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

- Verify that you can now access the web server that runs on the 82/TCP port.

```
[root@servera ~]# curl http://servera.lab.example.com:82
Hello
```

- Open another terminal window on the workstation machine. Try to access the new web service from the workstation machine.

```
student@workstation:~$ curl http://servera.lab.example.com:82
curl: (7) Failed to connect to servera.lab.example.com port 82 after 2 ms:
Could not connect to server
```

- On the servera machine, open the 82/TCP port on the firewall.

- Open the 82/TCP port persistently, for the default zone on the firewall.

```
[root@servera ~]# firewall-cmd --permanent --add-port=82/tcp
success
```

- Load the firewalld service with the changes.

```
[root@servera ~]# firewall-cmd --reload
success
```

- On the workstation machine, verify that you can access the web server.

```
student@workstation:~$ curl http://servera.lab.example.com:82
Hello
```

- Return to the workstation system as the student user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

**Finish**

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish netsecurity-ports
```

## 14.5. Lab

# Manage Network Security

Control network connections to services by using the system firewall and the services that can bind to particular ports by using SELinux.

### Outcomes

- Configure the firewall and SELinux for a web server.

### Prerequisites

As the student user on the workstation machine, run the following `lab start netsecurity-review` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start netsecurity-review
```

### Instructions

Your company decided to run a new web application. This application listens on the 80/TCP and 1001/TCP ports. All changes that you make must persist across a reboot.

#### Important

The Red Hat Online Learning environment needs the 5900/TCP port to remain available to use the graphical interface. This port is also present under the `vnc-server` service. If you accidentally lock yourself out of the `serverb` machine, then you can try to recover by using the `ssh` command to your `serverb` machine from your `workstation` machine. The configuration on your machines already includes a custom zone named `ROL` that opens these ports.

- On the `workstation` machine, try to access the web server at `http://serverb.lab.example.com` and the virtual host at `http://serverb.lab.example.com:1001`. The index files are provisioned for each host, but the tests currently fail.
- On the `serverb` machine, check for the issues that might be preventing access to the web server.
- Configure SELinux to allow the `httpd` service to listen on the 1001/TCP port.

- For use by Wong Chia Cheong jason.wong76 jason.wong@trainoate.com Copyright © 2025 Red Hat, Inc.
4. Open another terminal window from the workstation machine. Try to access the web server by using the `http://serverb.lab.example.com` and `http://serverb.lab.example.com:1001` URLs.
  5. On the `serverb` machine, determine whether the correct ports are added in the `firewallld` service.
  6. Add the 1001/TCP port to the permanent configuration for the public network zone.
  7. Return to the workstation machine as the student user.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
student@workstation:~$
```

8. On the workstation machine, confirm that you can access the web server at the `http://serverb.lab.example.com` through the 80/TCP port and that it returns SERVER B. Verify that you can access the virtual host at `http://serverb.lab.example.com:1001` through the 1001/TCP port and that it returns VHOST 1.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade netsecurity-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish netsecurity-review
```

## Solution

Control network connections to services by using the system firewall and the services that can bind to particular ports by using SELinux.

## Outcomes

- Configure the firewall and SELinux for a web server.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start netsecurity-review
```

## Instructions

Your company decided to run a new web application. This application listens on the 80/TCP and 1001/TCP ports. All changes that you make must persist across a reboot.

### Important

The Red Hat Online Learning environment needs the 5900/TCP port to remain available to use the graphical interface. This port is also present under the vnc-server service. If you accidentally lock yourself out of the serverb machine, then you can try to recover by using the ssh command to your serverb machine from your workstation machine. The configuration on your machines already includes a custom zone named ROL that opens these ports.

- On the workstation machine, try to access the web server at <http://serverb.lab.example.com> and the virtual host at <http://serverb.lab.example.com:1001>. The index files are provisioned for each host, but the tests currently fail.
  - Try to access the <http://serverb.lab.example.com> web server through the 80/TCP port. The web server should return SERVER B, but fails to connect.

```
student@workstation:~$ curl http://serverb.lab.example.com
curl: (7) Failed to connect to serverb.lab.example.com port 80 after 3 ms: Could
not connect to server
```

- 1.2.** Try to access the `http://serverb.lab.example.com:1001` virtual host through the 1001/TCP port. The virtual host should return VHOST 1, but fails to connect as well.

```
student@workstation:~$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001 after 2 ms:
Could not connect to server
```

- 2.** On the `serverb` machine, check for the issues that might be preventing access to the web server.

- 2.1.** Log in to the `serverb` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 2.2.** Check whether the `httpd` service is active.

```
[root@serverb ~]# systemctl is-active httpd
inactive
```

- 2.3.** Try to start and enable the `httpd` service. The `httpd` service fails to start.

```
[root@serverb ~]# systemctl enable --now httpd
Created symlink '/etc/systemd/system/multi-user.target.wants/httpd.service' →
'/usr/lib/systemd/system/httpd.service'.
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for
details.
```

- 2.4.** Check the status of the `httpd` service. Press `Q` to exit the session.

```
[root@serverb ~]# systemctl status httpd.service
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset:
disabled)
     Active: failed (Result: exit-code) since Mon 2025-07-14 12:54:15 UTC; 1min
35s ago
   Invocation: 8baa10be5ed844afa8c1254f710e5b62
     Docs: man:httpd.service(8)
   Process: 4202 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
   Main PID: 4202 (code=exited, status=1/FAILURE)
     Status: "Reading configuration..."
   Mem peak: 3.6M
     CPU: 53ms

Jul 14 12:54:15 serverb systemd[1]: Starting httpd.service - The Apache HTTP
Server...
Jul 14 12:54:15 serverb (httpd)[4202]: httpd.service: Referenced but unset
environment variable evaluates to an empty string: OPTIONS
Jul 14 12:54:15 serverb httpd[4202]: (13)Permission denied: AH00072: make_sock:
could not bind to address [::]:1001
Jul 14 12:54:15 serverb httpd[4202]: (13)Permission denied: AH00072: make_sock:
could not bind to address 0.0.0.0:1001
Jul 14 12:54:15 serverb httpd[4202]: no listening sockets available, shutting
down
Jul 14 12:54:15 serverb httpd[4202]: AH00015: Unable to open logs
Jul 14 12:54:15 serverb systemd[1]: httpd.service: Main process exited,
code=exited, status=1/FAILURE
Jul 14 12:54:15 serverb systemd[1]: httpd.service: Failed with result 'exit-
code'.
Jul 14 12:54:15 serverb systemd[1]: Failed to start httpd.service - The Apache
HTTP Server.
```

- 2.5.** Check whether SELinux is blocking the httpd service from binding to the 1001/TCP port.

```
[root@serverb ~]# sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
-----
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket
port 1001.

***** Plugin bind_ports (99.5 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 1001
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 1001
    where PORT_TYPE is one of the following: http_cache_port_t, http_port_t,
jboss_management_port_t, jboss.messaging_port_t, ntop_port_t, puppet_port_t.

...output omitted...
```

### 3. Configure SELinux to allow the httpd service to listen on the 1001/TCP port.

#### 3.1. Use the semanage command to find the correct port type.

```
[root@serverb ~]# semanage port -l | grep 'http'
http_cache_port_t          tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t          udp      3130
http_port_t                tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t        tcp      5988
pegasus_https_port_t       tcp      5989
```

#### 3.2. Bind the 1001/TCP port to the http\_port\_t context label.

```
[root@serverb ~]# semanage port -a -t http_port_t -p tcp 1001
```

#### 3.3. Confirm that the 1001/TCP port is bound to the http\_port\_t context label.

```
[root@serverb ~]# semanage port -l | grep '^http_port_t'
http_port_t                tcp      1001, 80, 81, 443, 488, 8008, 8009, 8443, 9000
```

#### 3.4. Enable and start the httpd service.

```
[root@serverb ~]# systemctl enable --now httpd
```

#### 3.5. Verify that the httpd service is active.

```
[root@serverb ~]# systemctl is-active httpd  
active
```

**3.6.** Verify that the httpd service is enabled.

```
[root@serverb ~]# systemctl is-enabled httpd  
enabled
```

- 4.** Open another terminal window from the workstation machine. Try to access the web server by using the `http://serverb.lab.example.com` and `http://serverb.lab.example.com:1001` URLs.

**4.1.** Try to access the `http://serverb.lab.example.com` web server through the 80/TCP port. The web server should return SERVER B.

```
student@workstation:~$ curl http://serverb.lab.example.com  
SERVER B
```

**4.2.** Try to access the `http://serverb.lab.example.com:1001` virtual host through the 1001/TCP port. Access to the web server fails.

```
student@workstation:~$ curl http://serverb.lab.example.com:1001  
curl: (7) Failed to connect to serverb.lab.example.com port 1001 after 2 ms:  
Could not connect to server
```

- 5.** On the serverb machine, determine whether the correct ports are added in the firewalld service.

**5.1.** Verify that the default firewall zone is set to the public zone.

```
[root@serverb ~]# firewall-cmd --get-default-zone  
public
```

**5.2.** If the previous step does not return public as the default zone, then correct it with the following command:

```
[root@serverb ~]# firewall-cmd --set-default-zone public  
success
```

**5.3.** Determine the open ports that are listed in the public network zone.

```
[root@serverb ~]# firewall-cmd --zone=public --list-all
public (default, active)
  target: default
  ingress-priority: 0
  egress-priority: 0
  icmp-block-inversion: no
  interfaces: ens3
  sources:
    services: cockpit dhcpcv6-client http ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

- 6.** Add the 1001/TCP port to the permanent configuration for the public network zone.

- 6.1.** Add the 1001/TCP port to the public network zone.

```
[root@serverb ~]# firewall-cmd --permanent --zone=public --add-port=1001/tcp
success
```

- 6.2.** Reload the firewall configuration.

```
[root@serverb ~]# firewall-cmd --reload
success
```

- 6.3.** Verify that the 1001/TCP port is added to the firewalld service.

```
[root@serverb ~]# firewall-cmd --zone=public --list-all
public (default, active)
  target: default
  ingress-priority: 0
  egress-priority: 0
  icmp-block-inversion: no
  interfaces: ens3
  sources:
    services: cockpit dhcpcv6-client http ssh
  ports: 1001/tcp
  protocols:
    forward: yes
    masquerade: no
    forward-ports:
    source-ports:
    icmp-blocks:
    rich rules:
```

7. Return to the workstation machine as the student user.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
student@workstation:~$
```

8. On the workstation machine, confirm that you can access the web server at the `http://serverb.lab.example.com` through the 80/TCP port and that it returns SERVER B. Verify that you can access the virtual host at `http://serverb.lab.example.com:1001` through the 1001/TCP port and that it returns VHOST 1.

- 8.1. Access the web server at `http://serverb.lab.example.com` through the 80/TCP port.

```
student@workstation:~$ curl http://serverb.lab.example.com
SERVER B
```

- 8.2. Access the virtual host at `http://serverb.lab.example.com:1001` through the 1001/TCP port.

```
student@workstation:~$ curl http://serverb.lab.example.com:1001
VHOST 1
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work.

Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade netsecurity-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish netsecurity-review
```

## 14.6. Summary

---

In this lesson, you learned about securing network connections to services by using the system firewall, and enforcing the SELinux targeted policy to protect service ports through port labeling.

The `firewalld` service is a dynamic firewall manager that works as a front end for the Linux kernel's `netfilter` framework. The `firewalld` service manages network traffic rules for packet filtering, network address translation, and port translation. The `firewalld` service classifies network traffic into zones. The `firewalld` service is managed by the `firewall-cmd` command-line tool or by the web console graphical interface.

SELinux controls network access by labeling network ports and including rules in a service's targeted policy. If a service attempts to listen on a nonstandard port, and the port is not labeled with the correct SELinux type, then SELinux might block the attempt. The `semanage port` command manages the SELinux policy with regard to port labeling.

## Chapter 15.

### Accessing Network-attached Storage

---

#### Goal

Access network-attached storage that the Network File System (NFS) protocol provides, either manually or by using the automounter.

#### Sections

- Mounting NFS File Systems (and Guided Exercise)
- Automounting Storage Devices (and Guided Exercise)

#### Lab

- Access Network-attached Storage

## 15.1. Mounting NFS File Systems

### Objectives

- Find, mount, and unmount remote NFS exports.

### Accessing Exported NFS Directories

The *Network File System* is an internet standard protocol that Linux, UNIX, and similar operating systems use as their native network file system. NFS is an open standard that supports native Linux permissions and file-system attributes.

By default, Red Hat Enterprise Linux 10 uses NFS version 4.2. RHEL 10 fully supports both NFSv3 and NFSv4 protocols. NFSv3 might use either a TCP or a UDP transport protocol. NFSv4 supports only TCP connections.

NFS servers export directories. NFS clients mount exported directories to an existing local mount point directory.

NFS clients can mount exported directories by using one of the following methods:

- *Manually* mount the exported directory by using the `mount` command.
- *Persistently* mount the directory at boot by using the `/etc/fstab` file.
- *Automounting* the directory statically on demand by using the `autofs` service.

The `mount` command does not mount the exported directory to persist after a reboot.

The `/etc/fstab` file mounts the exported directory persistently to remain mounted after a reboot. When mounting exported directories by using the `/etc/fstab` file, the following issues might occur, for example:

- The storage timeouts might increase if network issues occur.
- The timeouts also increase when the affected NFS file systems are not in use.
- Boot time issues might occur if the network is down or unreachable.

Automounting the exported directory is discussed in the next section of this chapter.

You must install the `nfs-utils` package for manual mounting, or automounting the exported NFS directories.

```
root@host:~# dnf install nfs-utils
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

RHEL 10 also supports mounting the shared directories from Microsoft Windows systems by using the same methods as the NFS protocol, by using either the *Server Message Block* (SMB) or the *Common Internet File System* (CIFS) protocols. Mounting options are protocol-specific, and depend on your Windows Server or Samba Server configuration. Mounting the shared directories from Microsoft Windows systems is beyond the scope of this course.

## Query Exported NFS Directories

The NFS protocol changed significantly between NFSv3 and NFSv4. The method to query a server to view the available exports is different for each protocol version.

NFSv3 uses the *Remote Procedure Call* (RPC) protocol, which requires a file server that supports NFSv3 connections to run the `rpcbind` service. An NFSv3 client connects to the `rpcbind` service through the 111 port on the NFS server. The server responds with the current port for the NFS service.

Use the `showmount` command to query the available exports on an RPC-based NFSv3 server.

```
root@host:~# showmount --exports server

Export list for server
/shares/test1
/shares/test2
```

The NFSv4 protocol eliminates the use of the legacy RPC protocol for NFS transactions.

NFSv4 introduces an *export tree* that contains all the paths for the server's exported directories. To view all the exported directories, mount the root (/) of the server's export tree.

```
root@host:~# mount server:/export mountpoint
```

Mounting the export tree provides browseable paths for all exported directories, but does not mount any of the exported directories.

```
root@host:~# ls mountpoint
```

To mount an NFSv4 export when browsing the mounted export tree, change the current directory to an exported directory path. Alternatively, use the `mount` command with an exported directory's full path name to mount a single exported directory. Exported directories that use Kerberos security do not allow mounting or accessing a directory when browsing an export tree, even though you can view the export's path name. Mounting Kerberos-protected shares requires additional server configuration and the use of Kerberos user credentials, which are discussed in the *Red Hat Security: Identity Management and Active Directory Integration* (RH362) training course.

## Manually Mount Exported NFS Directories

After you identify the NFS export to mount, create a local mount point if it does not yet exist. Although the `/mnt` directory is available for use as a temporary mount point, the recommended practice is to not use `/mnt` for long-term or persistent mounting.

```
root@host:~# mkdir mountpoint
```

As with local volume file systems, mount the NFS export to access its contents.

NFS shares can be mounted temporarily or permanently, and only by a privileged user.

```
root@host:~# mount -t nfs -o rw,sync server:/export mountpoint
```

The `-t nfs` option specifies the NFS file-system type. However, when the `mount` command detects the `server:/export` syntax, the command defaults to the NFS type. With the `-o` option, you can add a list of comma-separated options to the `mount` command. In the example, the `rw` option specifies that the exported file system is mounted with read/write access. The `sync` option specifies synchronous transactions to the exported file system. This method is strongly recommended for all production network mounts where transactions must be completed or else return as failed.

Manual mounts are useful for providing temporary access to an exported directory, or for testing the NFS export before persistently mounting it.

## Persistently Mount Exported NFS Directories

Use the `/etc/fstab` file to persistently mount an NFS export.

This file uses syntax that is similar to the manual mounting.

```
...output omitted...
server:/export mountpoint nfs rw 0 0
```

The mount command obtains the NFS server and mount options from the /etc/fstab file.

```
root@host:~# mount mountpoint
```

## Unmount Exported NFS Directories

As a privileged user, unmount an NFS export with the umount command.

```
root@host:~# umount mountpoint
```

Unmounting a share does not remove the NFS export entry from the /etc/fstab file. Entries in the /etc/fstab file are persistent and are remounted during boot.

A mounted directory can sometimes fail to unmount, and return an error that the device is busy. The device might be busy because an application has a file that is open, or a shell has a working directory in the mounted file system.

To resolve the error, close the application that has a file that is open, or check active shell windows, and use the cd command to leave the mounted file system. If subsequent attempts to unmount the file system still fail, then use the lsof command to query the mount point.

The lsof command returns a list of open file names and the process which is preventing from unmounting the exported directory.

```
root@host:~# lsof mountpoint
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
program 5534 user txt REG 252.4 910704 128 /home/user/program
```

With this information, gracefully close any processes that are using files on this file system, and retry the unmount. In critical scenarios only, when an application cannot be closed gracefully, kill the process to close the file.

Alternatively, use the umount command -f option to force the unmount, which can cause the loss of unwritten data for all open files.

## References

mount(8), umount(8), showmount(8), fstab(5), mount.nfs(8), nfsconf(8), and rpcbind(8) man pages

## 15.2. Guided Exercise

### Mount NFS File Systems

Find, mount, and unmount remote NFS exports.

#### Outcomes

- Manually mount the NFS export directory.
- Mount the NFS export directory persistently.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start nfsclient-nfs
```

#### Instructions

1. Log in to the servera machine as the student user and switch to the root user. Use student as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. On the servera machine, manually mount the serverb:/shares/public NFS server on the /public directory, and then unmount it.

- 2.1. Create the /public mount point on the servera machine.

```
[root@servera ~]# mkdir /public
```

- 2.2. Manually mount the /shares/public NFS export from the serverb machine on the /public directory.

```
[root@servera ~]# mount -t nfs serverb:/shares/public /public
```

**2.3.** List the contents of the mounted NFS export.

```
[root@servera ~]# ls -l /public
total 4
-rw-r--r--. 1 root root 14 Jul 11 22:12 hello
```

**2.4.** Display the mount command options for the mounted NFS export.

```
[root@servera ~]# mount | grep public
serverb:/shares/public on /public type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
```

**2.5.** Unmount the NFS export.

```
[root@servera ~]# umount /public
```

**3.** Persistently mount the serverb:/shares/public NFS export on the /public directory by using the /etc/fstab file.

**3.1.** Append an entry to the /etc/fstab file for the serverb:/shares/public export to mount on the /public directory. Mount the NFS export to synchronously mount the export with read/write permission.

```
serverb:/shares/public /public nfs rw,sync 0 0
```

**3.2.** Load the systemd daemon with the new configuration.

```
[root@servera ~]# systemctl daemon-reload
```

**3.3.** Mount the NFS export.

```
[root@servera ~]# mount /public
```

**3.4.** List the contents of the exported directory.

```
[root@servera ~]# ls -l /public
total 4
-rw-r--r--. 1 root root 14 Jul 11 22:12 hello
```

**3.5.** Reboot the servera machine.

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

- 4.** After the servera machine finishes rebooting, log in as the student user and test the persistently mounted NFS export.

**4.1.** Log in to the servera machine as the student user.

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

**4.2.** Test the NFS export that is mounted on the /public directory.

```
[student@servera ~]$ ls -l /public
total 4
-rw-r--r--. 1 root root 14 Jul 11 22:12 hello
```

**4.3.** Display the content of the /public/hello file.

```
[student@servera ~]$ cat /public/hello
Hello, World!
```

- 5.** Return to the workstation machine as the student user.

```
[admin1@servera ~]$ exit
logout
Connection to servera closed.
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish nfsclient-nfs
```

## 15.3. Automounting Storage Devices

### Objectives

- Automatically mount storage devices when they are accessed and unmount them when they are no longer in active use, by configuring the automounter.

### Mount NFS Exports with the Automounter

The *automounter* (*autofs*) is a service that automatically mounts file systems and NFS exports on demand, and automatically unmounts them when the mounted resources are no longer in current use.

The automounter function was created to solve the problem that nonprivileged users do not have sufficient permissions to use the `mount` command. Without using the `mount` command, nonprivileged users cannot access removable media such as CDs, DVDs, and removable disk drives. Furthermore, if a local or remote file system is not mounted at boot by using the `/etc/fstab` file, then a nonprivileged user cannot mount and access those unmounted file systems.

The automounter configuration files are populated with file-system mount information, in a similar way to the `/etc/fstab` file entries. Although the `/etc/fstab` file mounts the file systems during system boot, and they remain mounted until system shutdown or other intervention, the automounter does not necessarily mount during system boot. The automounter mounts file systems on demand, when a user or application attempts to enter the file-system mount point to access files.

### Automounter Benefits

Automounter uses equivalent resources to file systems that are mounted at boot, because a file system uses resources only when a program is reading or writing open files. Mounted and idle file systems, and unmounted file systems, use almost no resources.

The advantage of an automounter is that, by unmounting the file system each time that it is no longer in use, the file system is protected from unexpected corruption when it is open. When the file system is directed to mount again, the *autofs* service uses the most current mount configuration, unlike the `/etc/fstab` file, which might still use a configuration that was mounted months ago during the last system boot. Additionally, if your NFS server configuration includes redundant servers and paths, then the automounter can select the fastest connection each time that a new file system is requested.

### The Automounter Service

The automounter service supports the same local and remote file systems as in the /etc/fstab file, including NFS and SMB file sharing protocols, and supports the same protocol-specific mount options, including security parameters. File systems that are mounted through the automounter are available by default to all users, but can be restricted through access permission options.

Because the automounter is a client-side configuration that uses the standard mount and umount commands to manage file systems, automounted file systems in use exhibit the same behavior to file systems that are mounted by using the /etc/fstab file. The difference is that an automounter file system remains unmounted until the mount point is accessed, which causes the file system to mount immediately, and to remain mounted when the file system is in use. When all files on the file system are closed, and all users and processes leave the mount point directory, the automounter unmounts the file system after a minimal timeout.

## Direct and Indirect Map Use Cases

The autofs service supports both direct and indirect mount-point mapping, to handle the two types of mounting.

A *direct* mount is when a file system mounts to an unchanging, known mount point location. A direct mount point exists as a permanent directory.

An *indirect* mount is when the mount point location is not known until the mount demand occurs. Although indirect mount points appear to exist, the autofs service creates them when the mount demand occurs, and deletes them again when the demand ended and the file system is unmounted.

## Configure the Automounter Service

To use the autofs service, you must install the autofs and nfs-utils packages.

```
root@host:~# dnf install autofs nfs-utils
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

## The Master Map File

The autofs service uses the /etc/auto.master file as the default master map configuration file. You can use the /etc/autofs.conf file to change the map file for the autofs service.

Use the /etc/auto.master.d directory to configure the master map file. This file identifies the base directory for mount points, and identifies the mapping file to create the automounts.

The name of the master map file is mostly arbitrary (although typically meaningful), and it must have the `.autofs` extension for the subsystem to recognize it. You can place multiple entries in a single master map file; alternatively, you can create multiple master map files, each with its own logically grouped entries.

The master map uses the following format for automounting:

```
root@host:~# cat /etc/auto.master.d/demo.autofs
mountpoint    map-file
```

Replace the `mountpoint` variable with the directory to use as the mount point.

Replace the `map-file` variable with the file to use as the map file. The map file must be created before starting the `autofs` service.

## The Map File

Map files configure the properties of individual on-demand mount points.

The automounter creates the directories if they do not exist. If the directories exist before the automounter starts, then the automounter does not remove them when it exits. If a timeout is specified, then the directory is automatically unmounted if the directory is not accessed for the timeout period.

If you use a single directory name for the mount point, then the directory is mounted as an indirect mount. If you use the full path for the mount point, then the directory is mounted as a direct mount.

The map file uses the following format for automounting:

```
root@host:~# cat /etc/auto.demo
mountpoint    mount-options    source-location
```

Replace the `mountpoint` variable with the directory to use as the mount point.

Replace the `mount-options` variable with the options to use for mounting the automount.

Replace the `source-location` variable with the source location of the mount.

### Create a Direct Map File

A direct map file maps an NFS export to an absolute path mount point. Only one direct map file is necessary, and can contain any number of direct maps.

To use directly mapped mount points, include the following content in the master map file:

```
root@host:~# cat /etc/direct.autofs  
/- /etc/auto.direct
```

The naming convention for a map file is `/etc/auto.name`, where *name* reflects the content of the map.

All direct map entries use the `/ -` path as the base directory. In this case, the `/etc/auto.direct` mapping file contains the mount details.

The following example shows a sample entry in the map file:

```
root@host:~# cat /etc/auto.direct  
/mnt/docs -rw, sync hosta:/shares/docs
```

The mount point is always an absolute path. The rest of the mapping file uses the same structure.

In this example, the `/mnt` directory exists, and the `autofs` service does not manage it. The `autofs` service creates and removes the full `/mnt/docs` directory automatically.

### Create an Indirect Map File

Each mapping file identifies the mount point, mount options, and the source location to mount for a set of automounts.

For indirect automounting, include the following content in the master map file:

```
root@host:~# cat /etc/auto.master.d/indirect.autofs  
/shares /etc/auto.indirect
```

This entry uses the `/shares` directory as the base mount point for the indirect automount. The `/etc/auto.indirect` map file contains the mount details.

The following example shows a sample entry in the map file:

```
root@host:~# cat /etc/auto.indirect  
work -rw, sync hosta:/shares/work
```

The naming convention for a map file is `/etc/auto.name`, where *name* reflects the content of the map.

The fully qualified mount point is the `/shares/work` directory. The `autofs` service creates and removes the `/shares` and `/shares/work` directories as needed.

In this example, the local mount point mirrors the server's directory structure. However, this mirroring is not required; the local mount point can have an arbitrary name. The `autofs` service does not enforce a specific naming structure on the client.

Mount options start with a dash character (-) and are comma-separated with no white space. The file-system mount options for manual mounting are also available when automounting. In this example, the automounter mounts the export with read/write access (`rw` option), and the server is synchronized immediately during write operations (`sync` option).

Useful automounter-specific options include the `-fstype=` or `-strict` options. Use the `fstype` option to specify the file-system type, for example NFS4 or XFS, and use the `strict` option to treat errors as fatal when mounting file systems.

The source location for NFS exports follows the `host:/pathname` pattern, in this example `hosta:/shares/work`. For this automount to succeed, the NFS server, `hosta`, must export the directory with read/write access, and the user that requests access must have standard Linux file permissions on the directory. If the `hosta` machine exports the directory with read-only access, then the client gets read-only access even if the client requested read/write access.

## Using Wildcards in Map Files

When an NFS server exports multiple subdirectories within a directory, the automounter can then be configured to access any of those subdirectories with a single mapping entry.

Continuing the previous example, if the `hosta:/shares` export contains two or more subdirectories, and they are accessible with the same mount options, then the entry for the `/etc/auto.indirect` file might appear as follows:

```
root@host:~# cat /etc/auto.indirect
* -rw, sync  hosta:/shares/&
```

The mount point (or key) is an asterisk character (\*), and the subdirectory on the source location is an ampersand character (&). Everything else in the entry is the same.

When a user attempts to access the `/shares/work` directory, the \* key (which is `work` in this example) replaces the ampersand in the source location, and the `hosta:/shares/work` export is

mounted. As with the indirect map file example, the `autofs` service creates and removes the work directory automatically.

## Start the Automounter Service

After you have configured the master map and map files, use the `systemctl` command to start and enable the `autofs` service.

```
root@host:~# systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service →
/usr/lib/systemd/system/autofs.service.
```

## The Alternative Automount Method

The `systemd` daemon can automatically create unit files for entries in the `/etc/fstab` file that include the `x-systemd.automount` option. Use the `systemctl daemon-reload` command to update the `systemd` daemon of the new configuration file. Then, use the `systemctl start unit.automount` command to enable the automount configuration.

The naming of the unit is based on its mount location. For example, if the mount point is `/remote/finance`, then the unit file is named `remote-finance.automount`. The `systemd` daemon mounts the file system when the `/remote/finance` directory is initially accessed.

This method can be simpler than installing and configuring the `autofs` service. However, a `systemd.automount` unit can support only absolute path mount points, similar to the `autofs` direct maps.

### References

`autofs(5)`, `automount(8)`, `auto.master(5)`, `mount.nfs(8)`, and `systemd.automount(5)` man pages

For more information, refer to the *Mounting File Systems On Demand* chapter in the *Managing File Systems* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/managing\\_file\\_systems/index#mounting-file-systems-on-demand](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/managing_file_systems/index#mounting-file-systems-on-demand)

## 15.4. Guided Exercise

### Automount Storage Devices

---

Automatically mount storage devices when they are accessed and unmount them when they are no longer in active use, by configuring the automounter.

#### Outcomes

- Install the required packages for the automounter.
- Configure direct and indirect automounter maps, with resources from a preconfigured NFSv4 server.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start nfsclient-autofs
```

#### Instructions

1. Log in to the `servera` machine as the student user and install the `autofs` package. Use `student` as the password.

- 1.1. Log in to the `servera` machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 1.2. Install the `autofs` package. Use `student` as the password.

```
[student@servera ~]$ sudo dnf install autofs
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

2. Manually mount the `serverb:/shares` NFS export on the `/home/student/nfs` directory. Use the `grep` command to view all instances of the text RED in the `/home/student/nfs` directory.

- 2.1.** Manually mount the serverb:/shares NFS export on the /home/student/nfs directory.

```
[student@servera ~]$ sudo mount -t nfs serverb:/shares /home/student/nfs
```

- 2.2.** Find all instances of the text RED in the /home/student/nfs directory.

```
[student@servera ~]$ grep -r RED /home/student/nfs
/home/student/nfs/west/field.txt:You are standing in an open field west of a RED
house, with a boarded front door.
/home/student/nfs/west/mailbox/leaflet.txt:WELCOME TO RED HAT!
/home/student/nfs/west/mailbox/leaflet.txt:RED HAT Enterprise Linux is an
operating system of powerful tools and convenience. With it, you will explore
some of the most amazing technology ever seen. No computer should be without
one!
/home/student/nfs/south/house.txt:You are facing the south side of a RED house.
There is no door here, and all the windows are boarded.
```

- 3.** Create both a direct and an indirect autofs mount for the serverb:/shares NFS export. The direct mount must mount on the /home/student/direct directory. The indirect mount must mount on the /home/student/indirect directory. Synchronously mount both the mounts in read/write mode.

- 3.1.** Switch to the root user. Use student as the password.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 3.2.** Add the following content in the /etc/auto.master file:

```
/ - /etc/auto.direct
/home/student/indirect /etc/auto.indirect
```

- 3.3.** Configure a map file to mount the serverb:/shares NFS export for the direct automount.

Create the /etc/auto.direct file with the following content:

```
/home/student/direct -fstype=nfs,rw,sync serverb:/shares
```

- 3.4.** Configure a map file to mount the serverb:/shares NFS export for the indirect automount. Create the /etc/auto.indirect file with the following content:

```
* -fstype=nfs,rw,sync serverb:/shares/&
```

**3.5.** Start and enable the autofs service.

```
[root@servera ~]# systemctl enable --now autofs  
Created symlink '/etc/systemd/system/multi-user.target.wants/autofs.service' →  
'/usr/lib/systemd/system/autofs.service'.
```

**3.6.** Return to the servera machine as the student user.

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

- 4.** In the /home/student/direct directory, view all instances of the text RED. The NFS export does not mount until you access files within the mount.

**4.1.** Verify that no exports on the serverb machine are mounted on the direct directory.

```
[student@servera ~]$ mount | grep serverb  
serverb:/shares on /home/student/nfs type nfs4  
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,  
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,  
addr=172.25.250.11)  
serverb:/shares/south on /home/student/nfs/south type nfs4  
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,  
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,  
addr=172.25.250.11)  
serverb:/shares/west on /home/student/nfs/west type nfs4  
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,  
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,  
addr=172.25.250.11)
```

**4.2.** Find all instances of the text RED in the /home/student/direct directory.

```
[student@servera ~]$ grep -r RED /home/student/direct
/home/student/direct/south/house.txt:You are facing the south side of a RED
house. There is no door here, and all the windows are boarded.
/home/student/direct/west/field.txt:You are standing in an open field west of a
RED house, with a boarded front door.
/home/student/direct/west/mailbox/leaflet.txt:WELCOME TO RED HAT!
/home/student/direct/west/mailbox/leaflet.txt:RED HAT Enterprise Linux is an
operating system of powerful tools and convenience. With it, you will explore
some of the most amazing technology ever seen. No computer should be without
one!
```

#### 4.3. Verify that the NFS exports are now mounted.

```
[student@servera ~]$ mount | grep serverb
serverb:/shares on /home/student/nfs type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
serverb:/shares/south on /home/student/nfs/south type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
serverb:/shares/west on /home/student/nfs/west type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
serverb:/shares on /home/student/direct type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
serverb:/shares/south on /home/student/direct/south type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
serverb:/shares/west on /home/student/direct/west type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
```

5. In the /home/student/indirect directory, find all instances of the text RED. You cannot find the text RED in the directories that are not yet accessed.

- 5.1. Find all instances of the text RED before you access any directory in the NFS export. No output is produced, because automount did not mount the NFS export.

```
[student@servera ~]$ grep -r RED /home/student/indirect
```

- 5.2.** Access the /home/student/indirect/west directory by listing the contents of the directory.

```
[student@servera ~]$ ls /home/student/indirect/west  
field.txt  mailbox
```

- 5.3.** Find the instances of the text RED. You cannot find instances of the text RED in the /home/student/indirect/south directory.

```
[student@servera ~]$ grep -r RED /home/student/indirect  
/home/student/indirect/west/field.txt:You are standing in an open field west of  
a RED house, with a boarded front door.  
/home/student/indirect/west/mailbox/leaflet.txt:WELCOME TO RED HAT!  
/home/student/indirect/west/mailbox/leaflet.txt:RED HAT Enterprise Linux is an  
operating system of powerful tools and convenience. With it, you will explore  
some of the most amazing technology ever seen. No computer should be without  
one!
```

- 5.4.** Access the /home/student/indirect/south directory by listing the contents of the directory.

```
[student@servera ~]$ ls /home/student/indirect/south  
house.txt
```

- 5.5.** Verify that all instances of the text RED are displayed.

```
[student@servera ~]$ grep -r RED /home/student/indirect  
/home/student/indirect/south/house.txt:You are facing the south side of a `RED  
house. There is no door here, and all the windows are boarded.'  
/home/student/indirect/west/field.txt:You are standing in an open field west of  
a RED house, with a boarded front door.  
/home/student/indirect/west/mailbox/leaflet.txt:WELCOME TO RED HAT!  
/home/student/indirect/west/mailbox/leaflet.txt:RED HAT Enterprise Linux is an  
operating system of powerful tools and convenience. With it, you will explore  
some of the most amazing technology ever seen. No computer should be without  
one!
```

- 6.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish nfsclient-autofs
```

## 15.5. Lab

# Access Network-attached Storage

---

Access network-attached storage that the NFS protocol provides, either manually or by using the automounter.

### Outcomes

- Install the required packages to set up the automounter.
- Configure an automounter indirect map, with resources from a preconfigured NFSv4 server.

### Prerequisites

As the student user on the workstation machine, run the following `lab start nfsclient-review` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start nfsclient-review
```

### Instructions

An IT support company uses the `serverb` machine to share the management, production, and operation subdirectories of the `/shares` directory. Configure the `servera` machine to access the exported directories in the `/remote` directory.

Users of the `managers`, `production`, and `operators` groups must have read/write permissions to the `/shares/management`, `/shares/production`, and `/shares/operation` exported directories, respectively. The `manager1` user belongs to the `managers` group, the `dbuser1` belongs to the `production` group, and the `contractor1` user belongs to the `operators` group.

1. On the `servera` machine, install the `autofs` package.
2. On the `servera` machine, manually mount the `serverb:/shares` NFS export on the `/mnt` directory. This task helps you verify that exports are configured with the correct access and permissions. Unmount the `/mnt` directory.
3. On the `servera` machine, create an automounter indirect map to mount NFS exports along with all the subdirectories in the `/remote` directory.

Use the `/etc/auto.master.d/shares.autofs` file for the master map and the `/etc/auto.shares` file for the mapping file. Synchronously mount the `serverb:/shares/` & NFS export in the read/write mode.

Start the `autofs` service and configure it to start automatically.

- 4.** As the `manager1` user on the `servera` machine, verify that you can access the `/remote/management` directory.

List the contents of the `/remote/management` directory and view the content of the `/remote/management/Welcome.txt` file. Then, create the `/remote/management/test.txt` file with TEST as the text.

Return as the `root` user on the `servera` machine.

- 5.** As the `dbuser1` user on the `servera` machine, verify that you can access the `/remote/production` directory.

List the contents of the `/remote/production` directory and view the content of the `/remote/production/Welcome.txt` file. Then, create the `/remote/production/test.txt` file with TEST2 as the text.

Return as the `root` user on the `servera` machine.

- 6.** As the `contractor1` user on the `servera` machine, verify that you can access the `/remote/operation` directory.

List the contents of the `/remote/operation` directory and view the content of the `/remote/operation/Welcome.txt` file. Then, create the `/remote/operation/test.txt` file with TEST3 as the text.

Return as the `root` user on the `servera` machine.

- 7.** Verify that the NFS exports are now mounted because you accessed the exported directories.
- 8.** Return to the workstation machine as the student user.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade nfsclient-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish nfsclient-review
```

## Solution

---

Access network-attached storage that the NFS protocol provides, either manually or by using the automounter.

## Outcomes

- Install the required packages to set up the automounter.
- Configure an automounter indirect map, with resources from a preconfigured NFSv4 server.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start nfsclient-review
```

## Instructions

An IT support company uses the `serverb` machine to share the management, production, and operation subdirectories of the `/shares` directory. Configure the `servera` machine to access the exported directories in the `/remote` directory.

Users of the `managers`, `production`, and `operators` groups must have read/write permissions to the `/shares/management`, `/shares/production`, and `/shares/operation` exported directories, respectively. The `manager1` user belongs to the `managers` group, the `dbuser1` belongs to the `production` group, and the `contractor1` user belongs to the `operators` group.

**1.** On the `servera` machine, install the `autofs` package.

**1.1.** Log in to the `servera` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

**1.2.** Install the `autofs` package.

```
[root@servera ~]# dnf install autofs  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...  
Complete!
```

- 2.** On the servera machine, manually mount the serverb:/shares NFS export on the /mnt directory. This task helps you verify that exports are configured with the correct access and permissions. Unmount the /mnt directory.

- 2.1.** On the servera machine, mount the serverb:/shares NFS export on the /mnt directory.

```
[root@servera ~]# mount -t nfs serverb:/shares /mnt
```

- 2.2.** List the contents of the /mnt directory.

```
[root@servera ~]# ls -l /mnt  
total 0  
drwxrws---. 2 manager1    managers   25 Jul 14 20:23 management  
drwxrws---. 2 contractor1 operators  25 Jul 14 20:23 operation  
drwxrws---. 2 dbuser1     production 25 Jul 14 20:23 production
```

- 2.3.** Unmount the export.

```
[root@servera ~]# umount /mnt
```

- 3.** On the servera machine, create an automounter indirect map to mount NFS exports along with all the subdirectories in the /remote directory.

Use the /etc/auto.master.d/shares.autofs file for the master map and the /etc/auto.shares file for the mapping file. Synchronously mount the serverb:/shares/ & NFS export in the read/write mode.

Start the autofs service and configure it to start automatically.

- 3.1.** Create a master map file named /etc/auto.master.d/shares.autofs with the following content:

```
/remote /etc/auto.shares
```

- 3.2.** Create an indirect map file named /etc/auto.shares with the following content:

```
* -rw, sync, fstype=nfs4 serverb:/shares/&
```

**3.3.** Start and enable the autofs service.

```
[root@servera ~]# systemctl enable --now autofs  
Created symlink '/etc/systemd/system/multi-user.target.wants/autofs.service' →  
'/usr/lib/systemd/system/autofs.service'.
```

**3.4.** Verify that no exported directory is mounted yet. The directories are mounted when you access them.

```
[root@servera ~]# mount | grep nfs  
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
```

**4.** As the manager1 user on the servera machine, verify that you can access the /remote/management directory.

List the contents of the /remote/management directory and view the content of the /remote/management/Welcome.txt file. Then, create the /remote/management/test.txt file with TEST as the text.

Return as the root user on the servera machine.

**4.1.** Switch to the manager1 user.

```
[root@servera ~]# su - manager1
```

**4.2.** List the contents of the /remote/management directory.

```
[manager1@servera ~]$ ls -l /remote/management  
total 4  
-rw-r--r--. 1 manager1 managers 58 Jul 14 20:23 Welcome.txt
```

**4.3.** Verify that you can display the content of the /remote/management/Welcome.txt file.

```
[manager1@servera ~]$ cat /remote/management/Welcome.txt  
### Welcome to Management Folder on SERVERB ###
```

**4.4.** Verify write permission by creating the /remote/management/test.txt file.

```
[manager1@servera ~]$ echo TEST1 > /remote/management/test.txt
```

**4.5.** Exit from the manager1 user session.

```
[manager1@servera ~]$ exit  
logout  
[root@servera ~]#
```

**5.** As the dbuser1 user on the servera machine, verify that you can access the /remote/production directory.

List the contents of the /remote/production directory and view the content of the /remote/production/Welcome.txt file. Then, create the /remote/production/test.txt file with TEST2 as the text.

Return as the root user on the servera machine.

**5.1.** Log in to the dbuser user on the servera machine.

```
[root@servera ~]# su - dbuser1
```

**5.2.** List the contents of the /remote/production directory.

```
[dbuser1@servera ~]$ ls -l /remote/production  
total 4  
-rw-r--r--. 1 dbuser1 production 46 Jul 14 20:23 Welcome.txt
```

**5.3.** Verify that you can display the content of the /remote/production/Welcome.txt file.

```
[dbuser1@servera ~]$ cat /remote/production/Welcome.txt  
### Welcome to Production Folder on SERVERB ###
```

**5.4.** Verify write permission by creating the /remote/production/test.txt file.

```
[dbuser1@servera ~]$ echo TEST2 > /remote/production/test.txt
```

**5.5.** Return to the servera machine as the root user.

```
[dbuser1@servera ~]$ exit  
logout  
[root@servera ~]#
```

- 6.** As the contractor1 user on the servera machine, verify that you can access the /remote/operation directory.

List the contents of the /remote/operation directory and view the content of the /remote/operation/Welcome.txt file. Then, create the /remote/operation/test.txt file with TEST3 as the text.

Return as the root user on the servera machine.

- 6.1.** Log in to the contractor1 user on the servera machine.

```
[root@servera ~]# su - contractor1
```

- 6.2.** List the contents of the /remote/operation directory.

```
[contractor1@servera ~]$ ls -l /remote/operation  
total 4  
-rw-r--r--. 1 contractor1 operators 45 Jul 14 01:13 Welcome.txt
```

- 6.3.** Verify that you can display the content of the /remote/operation/Welcome.txt file.

```
[contractor1@servera ~]$ cat /remote/operation/Welcome.txt  
### Welcome to Operation Folder on SERVERB ###
```

- 6.4.** Verify write permission by creating the /remote/operation/test.txt file.

```
[contractor1@servera ~]$ echo TEST3 > /remote/operation/test.txt
```

- 6.5.** Return to the servera machine as the root user.

```
[contractor1@servera ~]$ exit  
logout  
[root@servera ~]#
```

- 7.** Verify that the NFS exports are now mounted because you accessed the exported directories.

```
[root@servera ~]# mount | grep nfs
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
serverb:/shares/management on /remote/management type nfs4
(rw,relatime,sync,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
serverb:/shares/production on /remote/production type nfs4
(rw,relatime,sync,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
serverb:/shares/operation on /remote/operation type nfs4
(rw,relatime,sync,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,
timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,
addr=172.25.250.11)
```

8. Return to the workstation machine as the student user.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work.

Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade nfsclient-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish nfsclient-review
```

## 15.6. Summary

---

In this lesson, you learned about the Network File System, and how to mount and unmount file systems manually and persistently.

An NFS server exports directories for remote mounting. An NFS client mounts exported directories to a local mount point directory.

The automounter service (`autofs`) mounts and unmounts file systems and NFS exports on demand.

The `autofs` service provides support to users who do not have permission to mount NFS shares.

The automounter unmounts file systems when they are no longer in use to protect them from unexpected corruption.

The `autofs` service supports direct and indirect mount-point mapping. A *direct* mount is a known location with a permanent mount point. An *indirect* mount is a mount location that is not known until the mount demand occurs.

## **Chapter 16.**

### **Installing Red Hat Enterprise Linux**

---

#### **Goal**

Install Red Hat Enterprise Linux in package mode, either interactively or by using Kickstart.

#### **Sections**

- Installing Red Hat Enterprise Linux Interactively (and Guided Exercise)
- Automating Red Hat Enterprise Linux Installation with Kickstart (and Guided Exercise)
- Installing Red Hat Enterprise Linux (Quiz)

## 16.1. Installing Red Hat Enterprise Linux Interactively

---

### Objectives

- Describe installing Red Hat Enterprise Linux interactively in package mode.

### Installation Media

Red Hat provides different forms of installation media that you can download from the Customer Portal website by using your active subscription.

- A binary image file in ISO 9660 format that contains the Anaconda Red Hat Enterprise Linux installation program, and the BaseOS and AppStream package repositories. These repositories contain the necessary packages to complete the installation without additional repositories.
- A smaller boot ISO image file that contains only the Anaconda installer, and which requires network access during installation to download packages from online repositories that are hosted by HTTP, FTP, or NFS servers.
- Source code (human-readable programming language instructions) for Red Hat Enterprise Linux. The source DVDs have no documentation. This image helps to compile or develop your software according to the Red Hat Enterprise Linux version. The availability of the operating system source code offers transparency and helps developers to build their own software by using Red Hat Enterprise Linux as a foundation.

Red Hat Enterprise Linux 10 supports the following architectures:

- AMD and Intel 64-bit architectures (x86-64-v3)
- The 64-bit ARM architecture (ARMv8.0-A)
- IBM Power Systems, Little Endian (POWER9)
- 64-bit IBM Z (z14)

After downloading your preferred image, create bootable installation media by following the instructions in the [Interactively Installing RHEL from Installation Media](#) guide.

For example, to install RHEL on a bare-metal server, you could copy one of the ISO images to a USB drive and boot from that.

### Build Customized Images with RHEL Image Builder

Red Hat Enterprise Linux image builder helps to create customized images of Red Hat Enterprise Linux. RHEL image builder enables administrators to build custom system images for deployment on cloud platforms or on virtual environments for specialized use cases.

Use the `composer-cli` command or the Red Hat web console interface to access RHEL image builder.

## Disk and Memory Requirements

To install Red Hat Enterprise Linux 10, you must have a minimum of 10 GiB of space on the partition where RHEL will be installed.

The minimum RAM requirement depends on the installation type:

Installation type	Recommended minimum RAM size
Local media installation (USB, DVD)	1.5 GiB for x86-64-v3, ARMv8.0-A, and z14 architectures 3 GiB for POWER9 architecture
NFS network installation	1.5 GiB for x86-64-v3, ARMv8.0-A, and z14 architectures 3 GiB for POWER9 architecture
HTTP, HTTPS, or FTP network installation	3 GiB for x86-64-v3 and z14 architectures 4 GiB for ARMv8.0-A and POWER9 architecture

## Install Red Hat Enterprise Linux Manually

By using the binary DVD or boot ISO, administrators install a new RHEL system on a bare-metal server or on a virtual machine. The Anaconda program supports two installation methods: manual and automated.

- The manual installation interacts with the user to query how Anaconda installs and configures the system.
- The automated installation uses a *Kickstart* file to direct Anaconda how to install the system.

## Install RHEL by Using the Graphical Interface

Anaconda starts as a graphical application when you boot the system from the binary DVD or from the boot ISO.

At the **WELCOME TO RED HAT ENTERPRISE LINUX 10** screen, select the language, and click **Continue**.

Individual users can choose a preferred language after installation.

Anaconda presents the **INSTALLATION SUMMARY** window, the central interface to customize parameters before beginning the installation.

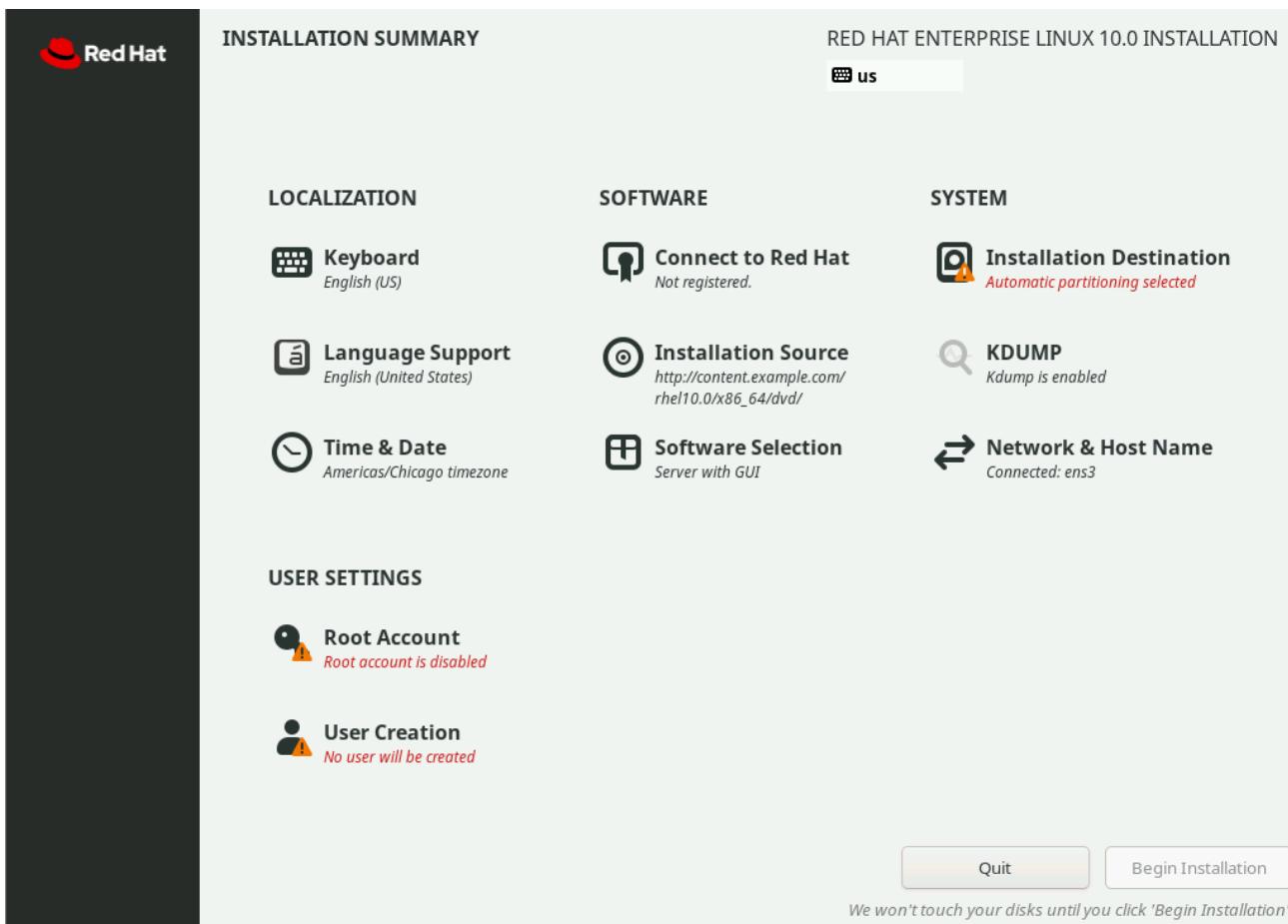


Figure 22. Installation summary window

From this window, configure the installation parameters by selecting the icons in any order. Select an item to view or to edit. In any item, click **Done** to return to this main screen.

Anaconda marks mandatory items with a triangle warning symbol and message. The orange status bar at the bottom of the screen reminds you to complete the required information before the installation begins.

Complete the following items as needed:

### Keyboard

Add keyboard layouts.

### Language Support

Select additional languages to install.

### Time & Date

Select the system's location by selecting the location from a list. Specify the local time zone even when using *Network Time Protocol (NTP)*. Alternatively, you can leave NTP disabled and set the time and date manually.

### Connect to Red Hat

Register the system with your Red Hat account and select the *system purpose*. The system purpose feature enables the registration process to automatically attach the most appropriate subscription to the system. You must first connect to the network by using the **Network & Host Name** icon to register the system. Unless you remove the corresponding checkmark, the system might also be enrolled in Red Hat Insights by default.

### Installation Source

Provide the source package location that Anaconda requires for installation. The **Installation Source** field already refers to the DVD when using the binary DVD.

### Software Selection

Select the base environment to install, and include any additional software. The **Minimal Install** environment installs only the essential packages to run Red Hat Enterprise Linux.

### Installation Destination

You must select at least one disk that has enough space to install Red Hat Enterprise Linux. To complete this task quickly, the administrator can rely on automatic storage configuration, which uses Logical Volume Management by default. Custom configuration is also available to help advanced users to create complex storage layouts. Note that even in custom mode, Anaconda enables you to automatically create partitions or volumes. Letting the installer determine the size is recommended because the installer considers the overall disk and RAM size.

The default radio button for automatic partitioning allocates the selected storage devices by using all available space.

## KDUMP

The *kdump* kernel crash dump feature collects information about the state of the system memory when the kernel crashes. Red Hat engineers analyze a *kdump* file to identify the cause of a crash. Use this Anaconda item to enable or to disable *kdump*.

## Network & Host Name

Detected network connections are listed on the left. Select a connection to display its details. By default, Anaconda activates the network automatically.

## Root Account

Starting in Red Hat Enterprise Linux 10, the `root` account is disabled by default. To enable it, select the `Enable root account` option and then set your `root` password.

### Note

By default, SSH access by the `root` account does not allow password-based authentication. However, password-based authentication can be allowed in Anaconda.

## User Creation

If the `root` account is disabled by default, then creating a non-root account with administrative privileges is mandatory. The non-root account with administrative privileges can use the `sudo` command to run commands as the `root` account. The non-root account with administrative privileges is added to the `wheel` group.

Creating a local, non-root user account is a recommended practice but is optional unless the `root` account is locked. You can also create accounts after the installation is complete if the `root` account is not disabled.

After you complete the installation configuration, resolve all warnings and click **Begin Installation**.

Clicking **Quit** halts the installation without applying any changes to the system.

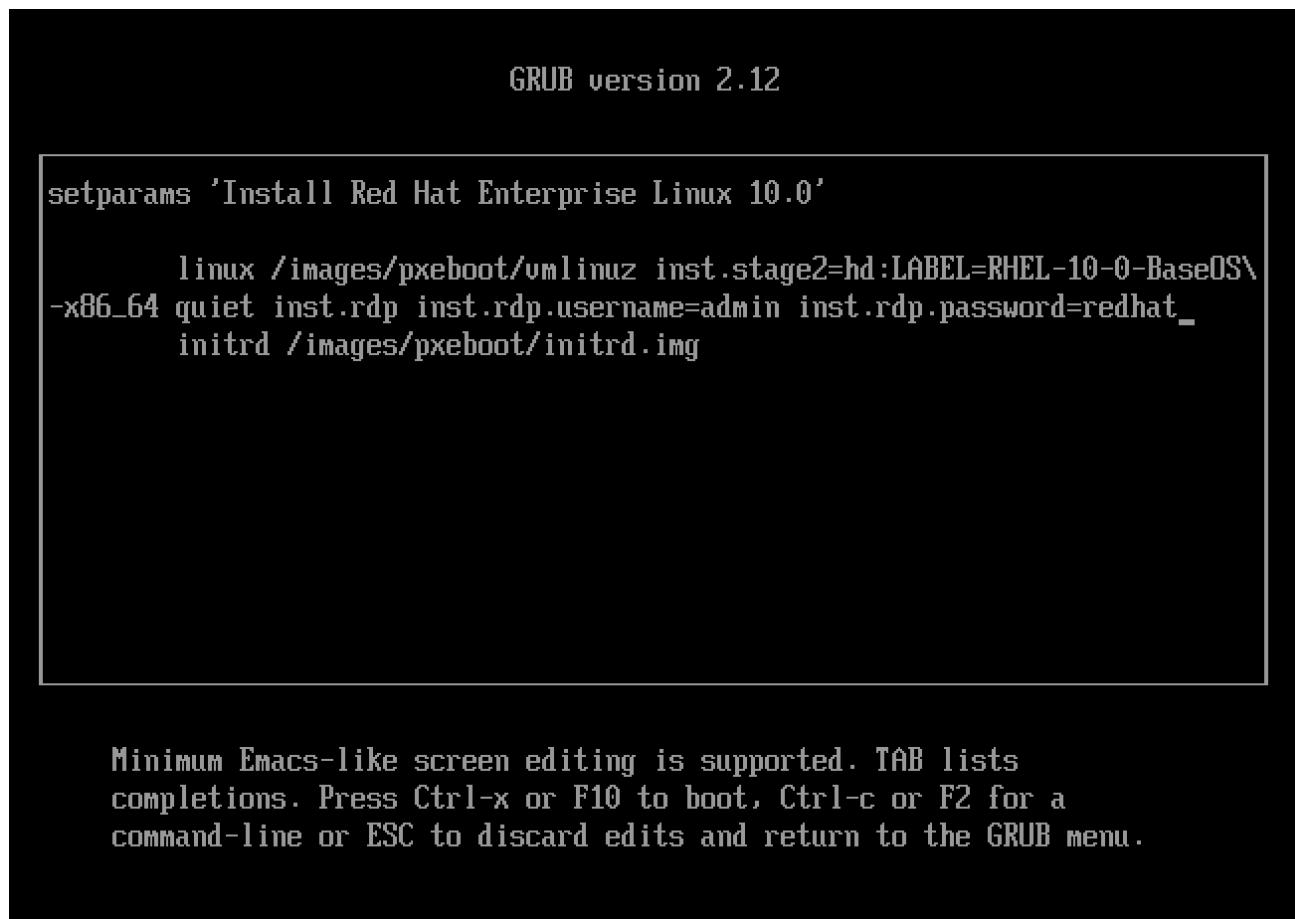
When the installation finishes, click **Reboot**. Anaconda displays the **Initial Setup** screen when you log in to a graphical desktop for the first time. Accept the license information and optionally register the system with the subscription manager. You might skip system registration until later.

## Install RHEL by Using Remote Desktop Protocol (RDP)

In Red Hat Enterprise Linux 10, the installer provides the option to use Remote Desktop Protocol (RDP) to control the installation. An RDP client application is required to interact remotely with the

Anaconda installer, which has performance benefits compared to the now deprecated VNC mode.

To start the installation in RDP mode, you must provide the `inst.rdp` option on the kernel command line in the boot-loader stage. Starting the RDP service requires authentication, which you can provide by using the `inst.rdp.username` and `inst.rdp.password` arguments on the command line. If no authentication details are provided on the command line, then Anaconda interactively prompts for an RDP username and password before starting the RDP service.



**Figure 23. RDP boot options**

When the installer successfully starts in RDP mode, it displays the IP address that the RDP client uses to start the installation. Although the network port for incoming client connections is not shown, the installer uses the default 5900 (TCP) RDP port. Multiple RDP clients can connect to the service for collaboration.

## Troubleshoot the Installation

During a Red Hat Enterprise Linux 10 installation, Anaconda provides two virtual consoles. The first virtual console is text-based and runs the tmux terminal multiplexer to provide six separate windows that show useful information. You can access that console by pressing `Ctrl + Alt + F1`. The second

virtual console, which is displayed by default, shows the Anaconda graphical interface. You can access it by pressing **Ctrl + Alt + F6**.

The tmux terminal provides a shell prompt in the second window in the first virtual console. You can use the terminal to enter commands to inspect and troubleshoot the system while the installation continues. The other windows provide diagnostic messages, logs, and additional information.

The following table lists the keystroke combinations to access the virtual consoles and the tmux terminal windows. In the tmux terminal, the keyboard shortcuts are performed in two actions: press and release **Ctrl + B**, and then press the number key of the window to access. In the tmux terminal, you can also press **Alt + Tab** to rotate the current focus between the windows.

Key sequence	Content
<b>Ctrl + Alt + F1</b>	Access the tmux terminal multiplexer.
<b>Ctrl + Alt + F6</b>	Access the Anaconda graphical interface.
<b>Ctrl + B 1</b>	In the tmux terminal, access the main information page for the installation process.
<b>Ctrl + B 2</b>	In the tmux terminal, provide a root shell. Anaconda stores the installation log files in the /tmp directory.
<b>Ctrl + B 3</b>	In the tmux terminal, display the contents of the /tmp/anaconda.log file.
<b>Ctrl + B 4</b>	In the tmux terminal, display the contents of the /tmp/storage.log file.
<b>Ctrl + B 5</b>	In the tmux terminal, display the contents of the /tmp/program.log file.
<b>Ctrl + B 6</b>	In the tmux terminal, display the contents of the /tmp/packaging.log file.

## Note

For compatibility with earlier Red Hat Enterprise Linux versions, the virtual consoles from **Ctrl** + **Alt** + **F2** through **Ctrl** + **Alt** + **F5** also present root shells during installation.

## References

For more information about installing Red Hat Enterprise Linux 10, refer to the *Interactively Installing RHEL from Installation Media* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/interactively\\_installing\\_rhel\\_from\\_installation\\_media/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/interactively_installing_rhel_from_installation_media/index)

For more information about feature updates for Red Hat Enterprise Linux 10, refer to the *Considerations in Adopting RHEL 10* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/considerations\\_in\\_adopting\\_rhel\\_10/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/considerations_in_adopting_rhel_10/index)

For more information about architecture changes for Red Hat Enterprise Linux 10, refer to the *Architectures* chapter in the *Considerations in Adopting RHEL 10* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/considerations\\_in\\_adopting\\_rhel\\_10/index#architectures](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/considerations_in_adopting_rhel_10/index#architectures)

For more information about installation media for Red Hat Enterprise Linux 10, refer to the *Creating a Bootable Installation Medium for RHEL* chapter in the *Interactively Installing RHEL from Installation Media* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/interactively\\_installing\\_rhel\\_from\\_installation\\_media/index#creating-a-bootable-installation-medium-for-rhel](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/interactively_installing_rhel_from_installation_media/index#creating-a-bootable-installation-medium-for-rhel)

## 16.2. Guided Exercise

### Install Red Hat Enterprise Linux Interactively

Install Red Hat Enterprise Linux interactively on a server.

#### Outcomes

- Manually install Red Hat Enterprise Linux 10.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start installing-install
```

#### Instructions

- To start a fresh installation of RHEL 10, access the `serverc` console and reboot the machine to launch PXE network booting.
  - Locate the `serverc` console icon in your classroom environment. Open the console.
  - To reboot, send `Ctrl + Alt + Del` to your machine by using the relevant keyboard, virtual, or menu entry.
  - When the PXE boot-loader menu appears, select `Install RHEL 10` and press `Enter`.
- Select your preferred language and click `Continue`. You will see the `INSTALLATION SUMMARY` screen.
- Configure date and time settings.
  - Click `Time & Date`.
  - Select Americas from the `Region` list.
  - Select New York from the `City` list.
  - Click `Configure NTP`.
  - Click `Remove` to delete the current entry.
  - Click `Add`. In the `Host Name` field, enter `classroom.example.com`, and then press `Enter`.
  - Click `Ok`, and then click `Done`.

**4.** Set up static networking.

**4.1.** Click **Network & Host Name** to configure the network.

**4.2.** Enter `serverc.lab.example.com` in the **Host Name** field, and then click **Apply**.

**4.3.** Click **Configure**, and then go to the **IPv4 Settings** tab. Select **Manual** from the **Method** list.

**4.4.** Click **Add**, and enter the following details:

Field	Value
<b>Address</b>	172.25.250.12
<b>Netmask</b>	24
<b>Gateway</b>	172.25.250.254
<b>DNS servers</b>	172.25.250.220

**Note**

During PXE network booting, the DHCP server on the classroom network leases a dynamic IP address from a pool to the target machine. In this exercise, the `serverc.lab.example.com` machine must be set up with a static IP address for production use. Note that the settings applied here take immediate effect and the new network details will be used during installation. These settings enable you to access the repository that is hosted on the installation network. These settings will also be configured permanently in NetworkManager on the newly installed machine.

**4.5.** Click **Save** to save the network configuration, and then click **Done**.

**5.** Verify that the content source is correctly configured.

**5.1.** Click **Installation Source**.

- 5.2.** Verify that the **On the network** option is selected and that `content.example.com/rhel10.0/x86_64/dvd` is in the address field.
- 5.3.** Click **Done**.
- 6.** Configure the storage settings for the installation.
  - 6.1.** Click **Installation Destination** to select the installation disk.
  - 6.2.** Select the sda disk.
  - 6.3.** In the **Storage Configuration** section, select **Custom**.
  - 6.4.** Click **Done**.
  - 6.5.** On the **MANUAL PARTITIONING** screen, select **Standard Partition** from the list of partitioning schemes.
  - 6.6.** Click **Click here to create them automatically** to create a default scheme based on the method that was selected in the previous step. Analyze the scheme that the installer creates. Note that the installer defines partition sizes automatically, depending on the disk capacity and amount of RAM that is installed.
  - 6.7.** Leave all settings unchanged. Click **Done**.
  - 6.8.** Click **Accept Changes** in the **SUMMARY OF CHANGES** window.
- 7.** Configure the selection of software to install on the machine.
  - 7.1.** Click **Software Selection**, and then select **Minimal Install** from the **Base Environment** list.
  - 7.2.** Click **Done**.
- 8.** Enable the root user in the machine.
  - 8.1.** Click **Root Account**, and select the **Enable root account** option.
  - 8.2.** Enter `redhat` in the **Root Password** field.
  - 8.3.** Enter `redhat` in the **Confirm** field.
  - 8.4.** Leave the **Allow root SSH login with password** checkbox unchecked.
  - 8.5.** Click **Done** twice because the password fails the dictionary check.
- 9.** Add the student user and set student as the password. Make sure that the student user has administrative privileges.
  - 9.1.** Click **User Creation**.
  - 9.2.** Enter `student` in the **Full Name** field. The student username automatically populates the **User name** field.

- 9.3.** Enter student in the **Password** field.
- 9.4.** Enter student in the **Confirm password** field.
- 9.5.** Leave the Add administrative privileges option checked. The student user requires sudo access and that option is now enabled by default.
- 9.6.** Click **Done** twice because the password fails the dictionary check.
- 10.** Begin the installation and monitor the process.
- 10.1.** Click **Begin Installation**.
- 10.2.** While the machine is being installed, switch to the tmux terminal with **Ctrl + Alt + F1** by using the relevant keyboard, virtual, or menu entry.
- 10.3.** Switch to terminal 2 by pressing **Ctrl + B**, and then pressing **2** to access the interactive shell. Go to the /tmp directory and review the contents of the anaconda.log, storage.log, program.log, and packaging.log files. You can use the tail, cat, or less commands to accomplish this task.
- 10.4.** Review terminals 3 to 6 by pressing **Ctrl + B**, and then pressing the number key of the window to access. Review the output from each terminal, and compare the output to the files that you reviewed in the previous step. The less command enables you to see previous log messages that are not displayed on these screens.
- 10.5.** Enter **Ctrl + Alt + F6** by using the relevant keyboard, virtual, or menu entry to return to the graphical installer. Wait for the process to complete.
- 11.** Click **Reboot System** when the installation is complete.
- 12.** Boot from local disk and log in to the machine.
- 12.1.** The PXE boot-loader menu appears again because this machine is intentionally configured to always boot from the network. Select Boot from local disk to start the newly installed operating system.
- 13.** Verify the system settings that were defined during installation.
- 13.1.** When the machine displays the login prompt, log in as the student user with student as the password.

```
serverc login: student
Password: student
...output omitted...
[student@serverc ~]$
```

- 13.2.** Run the `id` command as the `root` user. Use `redhat` as the password. If the command executes successfully, then the `root` account is enabled.

```
[student@serverc ~]$ su - -c id
Password: redhat
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 13.3.** Verify that the `student` user can run administrative commands by using the `sudo id` command. Use `student` as the password. A uid of `0(root)` shows that the `student` user has administrative privileges.

```
[student@serverc ~]$ sudo id
[sudo] password for student: student
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 13.4.** Use the `lsblk` command to verify the partitioning scheme that you accepted during installation.

```
[student@serverc ~]$ lsblk --output NAME,FSTYPE,SIZE,MOUNTPOINTS /dev/sda
NAME   FSTYPE SIZE MOUNTPOINTS
sda            10G
└─sda1        1M
└─sda2    xfs   1G /boot
└─sda3    swap   1G [SWAP]
└─sda4    xfs   8G /
```

- 14.** Verify the network configuration that was provided during installation.

- 14.1.** Use the `ip` command to verify the IP address configuration.

```
[student@serverc ~]$ ip addr show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:00:fa:0c brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname enx52540000fa0c
    inet 172.25.250.12/24 brd 172.25.250.255 scope global noprefixroute ens3
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe00:fa0c/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

**14.2.** Use the hostname command to verify the hostname of the machine.

```
[student@serverc ~]$ hostname
serverc.lab.example.com
```

**14.3.** Verify the DNS settings by displaying the contents of the /etc/resolv.conf file.

```
[student@serverc ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
search lab.example.com
nameserver 172.25.250.220
```

**14.4.** Verify the IP route settings.

```
[student@serverc ~]$ ip route show
default via 172.25.250.254 dev ens3 proto static metric 100
172.25.250.0/24 dev ens3 proto kernel scope link src 172.25.250.12 metric 100
```

**15.** Close the serverc console.

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish installing-install
```

## 16.3. Automating Red Hat Enterprise Linux Installation with Kickstart

### Objectives

- Describe and configure Kickstart to automatically install Red Hat Enterprise Linux.

### Introduction to Kickstart

The *Kickstart* feature of Red Hat Enterprise Linux automates a system installation that would typically require user interaction. You can use a single text file to provide answers to common installation questions, such as disk configuration, network settings, package selection, and users to create. The Kickstart feature is similar to an answer file that is used for an unattended installation of Microsoft Windows.

A Kickstart file begins with the command section, which defines several aspects of how to install the target system. The installer ignores comment lines, which start with the number sign (#). In addition to commands, you can also provide additional sections. Each of these sections must begin with a specific directive that starts with the percent sign (%). The %end directive marks the end of each section.

The %packages section specifies the software to include for installation. You can provide a detailed list of individual packages by name, without versions. The installer resolves any dependencies that the packages require. You can also use wildcards to match multiple packages. You can also define larger sets of packages in the form of groups or environments. The at sign (@) denotes package groups (either by group or ID), and the character combination of the at sign followed by a caret (@^) denotes environment groups (groups of package groups).

```
%packages
@^graphical-server-environment
vim*
%end
```

Groups have mandatory, default, and optional components. Typically, Kickstart installs mandatory and default components. To exclude a package or a package group from the installation, precede it with a hyphen (-). Excluded packages or package groups might still install if they are mandatory dependencies of other requested packages.

A Kickstart configuration file can include a %pre and a %post section to run automated scripts that further configure the system. Scripts can use any available interpreter on the system, including Bash or Python.

The %pre scripts execute before any disk partitioning is done. Typically, you use %pre scripts to initialize a storage or network device that the remainder of the installation requires. The %post scripts execute after the initial installation is complete.

Anaconda executes the preinstallation scripts outside the chroot environment that contains the file system of the target system. As a result, fewer commands might be available compared to postinstallation scripts, which run in the chroot environment and have access to all the tools on the newly installed system.

Administrators often use %post scripts to automate common changes during Kickstart. However, be aware that copying scripts or RPM packages from the installation media does not work in the chroot environment that is used in the final phase of the installation. Therefore, such tasks must be done in a %pre script.

Lastly, you can specify as many sections as you need, in any order. For example, you can have two %post sections, and they are interpreted in order of appearance.

## Installation Commands

The following Kickstart commands configure the installation source and method.

### url

Specifies the URL that points to the installation media that is hosted on a remote HTTP or FTP server.

```
url --url="http://classroom.example.com/rhel10.0/x86_64/dvd/"
```

### repo

Specifies where to find additional packages for installation. This option must point to a valid DNF repository.

```
repo --name="appstream" --
baseurl=http://classroom.example.com/rhel10.0/x86_64/dvd/AppStream/
```

#### text

Forces a text mode installation.

## Device and Partitioning Commands

The following Kickstart commands configure devices and partitioning schemes.

#### clearpart

Removes partitions from the system before creating partitions.

```
clearpart --all --drives=vda,vdb
```

#### part

Specifies the size, format, and name of a partition. Required unless the autopart or mount commands are present.

```
part /home --fstype=xfs --label=homes --size=4096 --maxsize=8192 --grow
```

#### autopart

Automatically creates a root partition, a swap partition, and an appropriate boot partition for the architecture. On large enough drives (50 GB+), this command also creates a /home partition.

#### ignoredisk

Prevents Anaconda from modifying disks, and is useful alongside the autopart command.

```
ignoredisk --drives=sdc
```

#### bootloader

Defines where to install the boot loader. Required.

```
bootloader --location=mbr --boot-drive=sda
```

#### volgroup, logvol

Create LVM volume groups and logical volumes.

```
part pv.01 --size=8192
volgroup myvg pv.01
logvol / --vgname=myvg --fstype=xfs --size=2048 --name=rootvol --grow
logvol /var --vgname=myvg --fstype=xfs --size=4096 --name=varvol
```

## zerombr

Initialize disks whose formatting is unrecognized.

## Network Commands

The following Kickstart commands configure networking-related features.

### network

Configures network information for the target system. Activates network devices in the installer environment.

```
network --device=enp1s0 --bootproto=dhcp
```

### firewall

Defines the firewall configuration for the target system.

```
firewall --enabled --service=ssh,http
```

## Location and Security Commands

The following Kickstart commands configure security, language, and region settings.

### lang

Sets the language to use on the installed system. Required.

```
lang en_US
```

### keyboard

Sets the system keyboard type. Required.

```
keyboard --vckeymap=us
```

### timezone

Defines the time zone and whether the hardware clock uses UTC. Required.

```
timezone --utc Europe/Amsterdam
```

## timesource

Enables or disables NTP. If you enable NTP, then you must specify NTP servers or pools.  
Optional.

```
timesource --ntp-server classroom.example.com
```

## authselect

Sets up authentication options. Options that the authselect command recognizes are valid for this command. Refer to the authselect(8) man page. Optional.

## rootpw

Defines the initial root user password. Starting in RHEL 10, this command is optional, because the root account is disabled by default.

```
rootpw --lock
```

```
rootpw --plaintext redhat
```

```
rootpw --iscrypted  
$6$KUnFfrTz08jv.PiH$YlBb0tXBkWzoMuRfb0.SpbQ....XDR1UuchoMG1
```

## selinux

Sets the SELinux mode for the installed system.

```
selinux --enforcing
```

## services

Modifies the default set of services to run under the default systemd target.

```
services --disabled=rsyslog --enabled=NetworkManager,firewalld
```

## group, user

Creates a local group or user on the system.

```
group --name=admins --gid=10001
user --name=jdoe --gecos="John Doe" --groups=admins
```

### Note

In RHEL 10, make sure to create a user that belongs to the `wheel` administrative group if you decide to keep the `root` account locked.

## Miscellaneous Commands

The following Kickstart commands configure logging the host power state on completion.

### logging

This command defines how Anaconda handles logging during the installation. Optional.

```
logging --host=loghost.example.com
```

### firstboot

Controls whether the GNOME Initial Setup application should start the first time that the system boots. This command requires the `initial-setup` package.

```
firstboot --disabled
```

### reboot, poweroff, halt

Specify the final action when the installation completes. The default setting is the `halt` option, which waits for the user to press a key before rebooting.

### Note

There are many more Kickstart commands available. Commands might also support complex options that are hard to remember without consulting references. For more information, refer to [Kickstart Commands and Options Reference](#).

## Example Kickstart File

In the following example, the command section of the Kickstart file is followed by package selection, an add-on section to configure the kdump feature of the Linux kernel, and a postinstallation script.

```
# version=RHEL10

# Installation mode
graphical

# Keyboard layouts
keyboard --vckeymap=us --xlayouts='us'

# System language
lang en_US.UTF-8

# System timezone
timezone America/New_York --utc

# Network information
network --bootproto=dhcp --device=enp1s0 --ipv6=auto --activate
network --hostname=serverb.lab.example.com

# Use network installation
url --url="http://content.example.com/rhel10.0/x86_64/dvd/"

# Storage commands
zerombr
clearpart --all --initlabel
autopart
ignoredisk --only-use=vda

# Root password
rootpw --iscrypted --allow-ssh
$y$j9T$jfY1BtZko.0LZxZLBCGeb7vM$7agy5.gxAgxgk9lGXCak1DP0Vb1j7uf8HPb7vWXc46B
user --groups=wheel --name=student --
password=$y$j9T$EQF0SpVuDKgpRnc2gmxnUnGV$6BqUu.HhnduZszglv6t1Vy6ptF8GQeQkk8maJX
WM/y5 --iscrypted --gecos="student"

%packages
@^minimal-environment
vim-enhanced
-hyperv*
%end

# Configure the kernel crash dumping mechanism
%addon com_redhat_kdump --enable --reserve-mb='auto'
%end
```

```
%post
echo "This system was deployed using Kickstart on $(date)" > /etc/motd
%end
```

 **Note**

In a Kickstart file, missing required values cause the installer to interactively prompt for an answer or to stop the installation entirely.

## Kickstart Installation Workflow

Use the following steps to automate the installation of Red Hat Enterprise Linux with the Kickstart feature:

1. Create a Kickstart file.
2. Publish the Kickstart file on removable media or a network server so that the Anaconda installer can access it.
3. Boot the Anaconda installer and point it to the Kickstart file.

## Create a Kickstart File

You can use either of these methods to create a Kickstart file:

- Use the Kickstart Generator website.
- Use a text editor to create your own version.

The Kickstart Generator site at <https://access.redhat.com/labs/kickstartconfig> presents dialog boxes for user inputs, and creates a Kickstart configuration file based on the user's choices. Each dialog box corresponds to the configurable items in the Anaconda installer.

The screenshot shows the Red Hat Customer Portal interface with the 'Kickstart Generator' section selected. On the left, a sidebar lists configuration categories: Installation, Partition, BootLoader, Packages, Authentication, Network, Security, Display, and Pre-Installation Script. The 'Basic Configuration' tab is active. In the main area, the 'Basic Configuration' panel is displayed with the following settings:

Default Language	English (USA)
Keyboard	English (US)
Time Zone	America/New York
<input checked="" type="checkbox"/> Use UTC clock	
Root Password *	<input type="text"/> Root Password is required.
Repeat Root Password *	<input type="text"/>
<input type="checkbox"/> Do not reboot system after installation	
<input type="checkbox"/> Perform installation in text mode (Graphical is default)	

At the top of the page, there are navigation links for Subscriptions, Downloads, Red Hat Console, and Get Support. The top right features a search bar, language selection (English), and a 'Ask Red Hat' button. The Red Hat logo is also present in the top right corner.

Figure 24. Red Hat Customer Portal Kickstart Generator

Creating a Kickstart file from scratch can be complex, so the recommended practice is to start with updating an existing sample file. Every installation creates a `/root/anaconda-ks.cfg` file that captures the Anaconda configuration that was used during the installation of the host system. This file is a good starting point to create a Kickstart file.

The `ksvalidator` utility checks for syntax errors in a Kickstart file. It ensures that keywords and options are correctly used, but it does not validate URL paths, individual packages, groups, nor any part of `%post` or `%pre` scripts. For example, if the `firewall --disabled` directive is misspelled, then the `ksvalidator` command might produce one of the following errors:

```
user@host:~$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:
Unknown command: firewall

user@host:~$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:
no such option: --dsabled
```

The ksverdiff utility displays syntax differences between different operating system versions. For example, the following command displays the Kickstart syntax changes between RHEL 8 and RHEL 9:

```
user@host:~$ ksverdiff -f RHEL9 -t RHEL10
The following commands were removed in RHEL10:
auth authconfig autostep btrfs method nvdimm

The following commands were deprecated in RHEL10:
vnc

The following commands were added in RHEL10:

The following options were added to the autopart command in RHEL10:
--hibernation
...output omitted...
```

The pykickstart package provides the ksvalidator and ksverdiff utilities.

## Publish the Kickstart File

The Kickstart file must be made available for the Anaconda installer to access it. The following list describes several options for storing the file:

- A network server that is available during installation by using the FTP, HTTP, or NFS protocols.
- A removable USB disk or CD-ROM.
- A local hard disk on the system.

The installer must access the Kickstart file to begin an automated installation. In enterprise environments, the file is usually made available via an FTP, HTTP, or NFS server.

A Kickstart file can also be published on removable media. A USB drive might be more convenient than burning the file to optical media such as a CD-ROM or DVD disc.

For a quick rebuild of a single system, you might choose to provide the Kickstart file on one of the local drives.

## Start the Installer and Point to the Kickstart File

When you boot the target system by using the installation media, you must inform the Anaconda installer where to locate the Kickstart file by passing the `inst.ks=LOCATION` parameter to the kernel.

Consider the following examples to specify the correct location of your published Kickstart file:

- `inst.ks=http://server/dir/file`
- `inst.ks=ftp://server/dir/file`
- `inst.ks=nfs:server:/dir/file`
- `inst.ks=hd:device:/dir/file`
- `inst.ks=cdrom:device`

The appropriate `inst.ks` argument can be added in the boot-loader stage. From the installation menu, select the `Install` menu item, and press `C` to edit the kernel arguments.

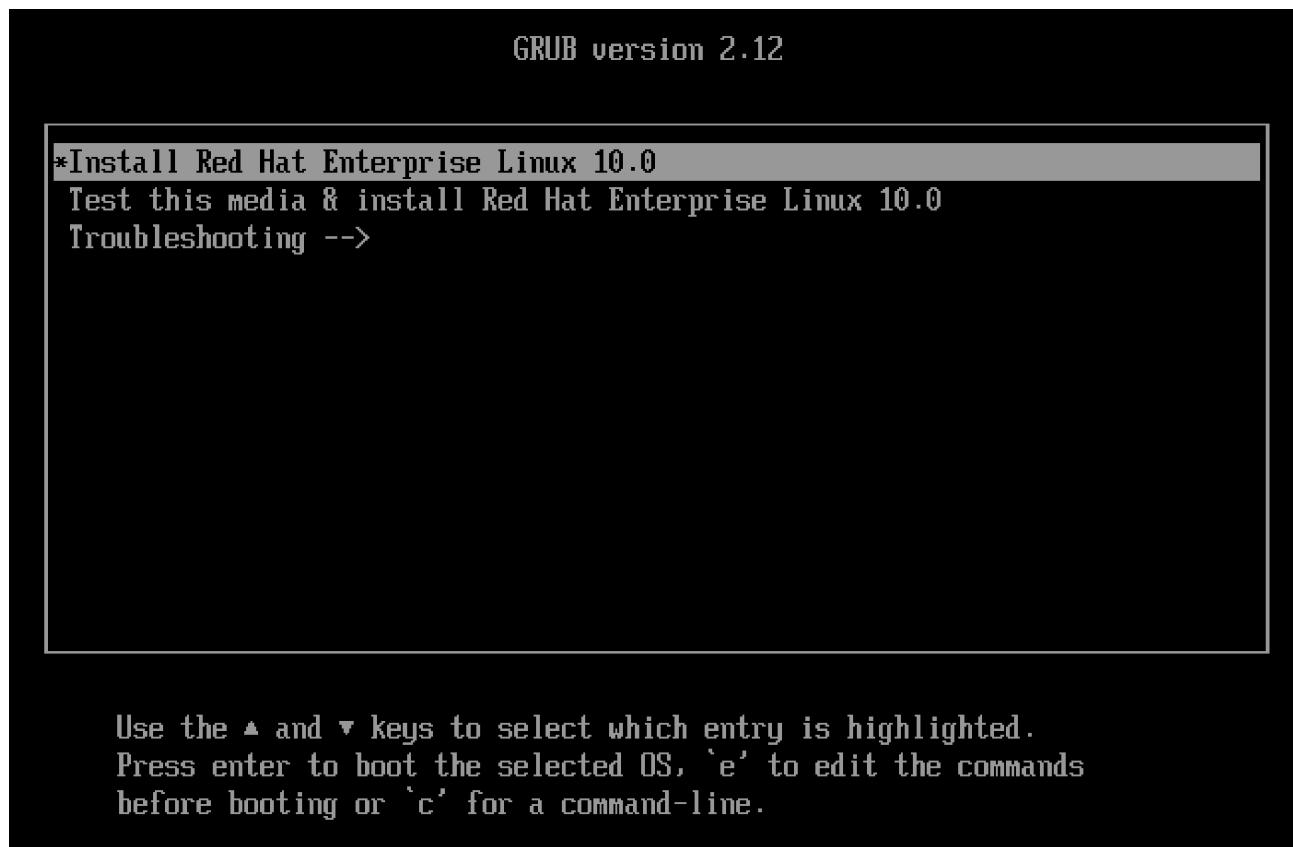


Figure 25. Selecting the boot-loader menu item to edit

Append the `inst.ks=` argument without changing anything else. Press `Ctrl + X` to start the boot process.

```
GRUB version 2.12

setparams 'Install Red Hat Enterprise Linux 10.0'

linux /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=RHEL-10-0-BaseOS\x
-x86_64 quiet inst.ks=http://classroom.example.com/kickstart.cfg_
initrd /images/pxeboot/initrd.img

Minimum Emacs-like screen editing is supported. TAB lists
completions. Press Ctrl-x or F10 to boot, Ctrl-c or F2 for a
command-line or ESC to discard edits and return to the GRUB menu.
```

Figure 26. Specifying the Kickstart file location

After specifying the Kickstart file location, the Anaconda installer will start. If the referenced Kickstart file includes all the required commands, then Anaconda does not prompt for further input.

During installation, you still have the option to use the standard Anaconda-related keyboard shortcuts from **Ctrl + Alt + F1** to **Ctrl + Alt + F6** to switch between screens and to get a root shell to issue troubleshooting commands. Using the shell directly to look at log files and issue commands is an invaluable resource to troubleshoot installation issues or failures. For example, incorrect syntax in Kickstart files or storage configuration that cannot be accomplished by using the available hard drives can cause the installation to fail. The installer also reports errors caused by malformed postinstallation scripts.

After the installation process has finished successfully, you might have to press **Reboot** to continue unless the Kickstart file specified the `reboot` command.

## References

For further information, refer to the *Automated Installation Workflow* chapter in the *Automatically Installing Red Hat Enterprise Linux* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/automatically\\_installing\\_rhel/index#automated-installation-workflow](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/automatically_installing_rhel/index#automated-installation-workflow)

For further information, refer to the *Kickstart Commands and Options Reference* chapter in the *Automatically Installing Red Hat Enterprise Linux* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/automatically\\_installing\\_rhel/index#kickstart-commands-and-options-reference](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/automatically_installing_rhel/index#kickstart-commands-and-options-reference)

## 16.4. Guided Exercise

# Automate Red Hat Enterprise Linux Installation with Kickstart

Install Red Hat Enterprise Linux automatically based on a predefined configuration by using Kickstart.

## Outcomes

- Create a valid Kickstart file.
- Make the file available by using the HTTP protocol.
- Install Red Hat Enterprise Linux 10 by using Kickstart.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start installing-kickstart
```

## Instructions

1. Update the Kickstart file on the servera machine.

- 1.1. On the workstation machine, log in to the servera machine as the student user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

2. Modify the ~/RH304/labs/installing-kickstart/kickstart.cfg Kickstart file with the following changes.

- 2.1. Change the installation mode from graphical to text mode. Replace the graphical command with the text command.

```
# Use text install mode
text
```

- 2.2. Update the packages section to include the guest-agents package group and the vim-enhanced package.

```
%packages  
@^minimal-environment  
@guest-agents  
vim-enhanced  
%end
```

**2.3.** Disable the GNOME Initial Setup application during the first login.

```
# Don't run the Setup Agent on first boot  
firstboot --disable
```

**2.4.** At the end of the file, add a postinstallation script that updates the index of all available manual pages on the machine. Update the login prompt message to show the date of the machine installation.

```
%post  
mandb  
echo "Kickstarted on $(date +%F)" >> /etc/issue  
%end
```

**2.5.** Save and close the Kickstart file.

**2.6.** Your updated Kickstart file must match the following content:

```
# version=RHEL10

# Use text install mode
text

# Keyboard layouts
keyboard --vckeymap=us --xlayouts='us'

# System language
lang en_US.UTF-8

# Use network installation
url --url="http://content.example.com/rhel10.0/x86_64/dvd/"
repo --name="AppStream" --
baseurl="http://content.example.com/rhel10.0/x86_64/dvd/AppStream"

%packages
@^minimal-environment
@guest-agents
vim-enhanced
%end

# Don't run the Setup Agent on first boot
firstboot --disable

# Storage commands
# Partition clearing information
zerombr
clearpart --all --initlabel
autopart
bootloader --location=mbr

# System timezone
timezone America/Chicago --utc

# Root password
rootpw --lock
user --groups=wheel --name=student --password=student --plaintext --
gecos="student"

%addon com_redhat_kdump --enable --reserve-mb='auto'
%end

%post
mandb
echo "Kickstarted on $(date +%F)" >> /etc/issue
%end
```

**3.** Validate the syntax in the Kickstart file.

**3.1.** Install the pykickstart package on the servera machine.

```
[student@servera ~]$ sudo dnf install pykickstart
[sudo] password for student: student
...output omitted...
=====
=====
  Package          Arch    Version      Repository
Size
=====
=====
Installing:
pykickstart        noarch   3.52.8-1.el10  rhel-10.0...rpms  35 k
Installing dependencies:
python3-kickstart  noarch   3.52.8-1.el10  rhel-10.0...rpms  620 k

Transaction Summary
=====
=====
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

**3.2.** Use the ksvalidator command to ensure that the syntax in your Kickstart file is correct. Fix any potential issues before proceeding.

```
[student@servera ~]$ ksvalidator ~/RH304/labs/installing-kickstart/kickstart.cfg

Checking kickstart file /home/student/RH304/labs/installing-
kickstart/kickstart.cfg
```

**4.** Publish the Kickstart file.

**4.1.** Use the Apache HTTP server that is running on the servera machine to publish the Kickstart file. Copy the completed Kickstart file to the /var/www/html directory.

```
[student@servera ~]$ sudo cp ~/RH304/labs/installing-kickstart/kickstart.cfg \
/var/www/html/
```

**4.2.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
student@workstation:~$
```

5. Access the `serverc` console and reboot the machine to initiate PXE network booting and to perform a fresh installation by using Kickstart.
  - 5.1. Locate the `serverc` console icon in your classroom environment. Open the console.
  - 5.2. To reboot, send `Ctrl + Alt + Del` to your machine by using the relevant keyboard, virtual, or menu entry.
  - 5.3. When the PXE boot-loader menu appears, select the `Install RHEL 10` menu entry and press `Tab` to edit the menu item.
  - 5.4. Add the `inst.ks=http://servera.lab.example.com/kickstart.cfg` boot option at the end of the line.
  - 5.5. Press `Enter` to start the installer.
  - 5.6. The installation begins automatically without prompting for any other information. As the installation proceeds, you can switch between the tmux windows by pressing `Ctrl + B` followed by the number of the window to access.
6. After the installation finishes, press `Enter` to reboot the `serverc` machine.
7. Boot from local disk and log in to the `serverc` machine.
  - 7.1. The PXE boot-loader menu appears again because this machine is intentionally configured to always boot from the network. Select the `Boot from local disk` menu entry to start the newly installed operating system.
  - 7.2. When the machine displays the login prompt, notice the installation date, which shows that the postinstallation script was successful.
  - 7.3. Log in to the `serverc` machine as the `student` user. Use `student` as the password.

```
localhost login: student  
Password: student  
[student@localhost ~]$
```

8. Verify system settings that were configured during the Kickstart-based installation.

- 8.1.** Verify that the serverc machine uses the Dynamic Host Configuration Protocol (DHCP) for network configuration because the Kickstart file did not add static addressing.

```
[student@localhost ~]$ nmcli con show ens3 | grep ipv4.method  
ipv4.method: auto
```

- 8.2.** Verify that the serverc machine uses the localhost hostname. The leased IP address does not have a corresponding hostname configured in DNS, so the machine uses the localhost transient hostname.

```
[student@localhost ~]$ hostnamectl | grep hostname  
  Static hostname: (unset)  
Transient hostname: localhost
```

- 8.3.** Use the lsblk command to inspect the storage configuration. The volumes use Logical Volume Management (LVM) by default because of the autopart Kickstart command.

```
[student@localhost ~]$ lsblk  
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS  
sda        8:0    0 10G  0 disk  
└─sda1     8:1    0   1M  0 part  
└─sda2     8:2    0   1G  0 part /boot  
└─sda3     8:3    0   9G  0 part  
  ├─rhel-root 253:0  0   8G  0 lvm  /  
  └─rhel-swap 253:1  0   1G  0 lvm  [SWAP]  
sr0       11:0   1   1G  0 rom
```

- 8.4.** To verify whether the postinstallation script ran as expected, use the apropos command to search for the fstab manual page. If the manual page is listed in the output, then the postinstallation script successfully updated the index of manual pages.

```
[student@localhost ~]$ apropos fstab  
fstab (5)           - static information about the filesystems  
systemd-fstab-generator (8) - Unit generator for /etc/fstab
```

- 9.** Close the serverc console.

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish installing-kickstart
```

## 16.5. Quiz

### Installing Red Hat Enterprise Linux

---

Choose the correct answers to the following questions:

- 1.** What are the recommended disk and memory requirements to perform a local installation of Red Hat Enterprise Linux 10 on x86-64 architecture?  
 **a.** 20 GiB of disk space and 3 GiB RAM  
 **b.** 10 GiB of disk space and 1.5 GiB RAM  
 **c.** 10 GiB of disk space and 3 GiB RAM  
 **d.** 20 GiB of disk space and 1 GiB RAM
- 2.** Which two features regarding root account creation changed in Red Hat Enterprise Linux 10? (Choose two.)  
 **a.** The root account is now disabled by default.  
 **b.** The default password is redhat and must be changed at first login.  
 **c.** SSH access for the root user is disabled by default.  
 **d.** The root account cannot be accessed directly, and the only way to execute administrative commands is with sudo privileges.
- 3.** What is a new feature of the Anaconda installer in RHEL 10?  
 **a.** Mouse support  
 **b.** Remote VNC access  
 **c.** Preinstallation scripts  
 **d.** Remote RDP access
- 4.** What is the primary purpose of the Kickstart feature in RHEL?  
 **a.** Manage user accounts  
 **b.** Enhance system security  
 **c.** Improve network performance  
 **d.** Fully automate system installation

**5.** Which directive marks the end of a section in a Kickstart file?

- a.** %stop
- b.** %finish
- c.** %end
- d.** %close

**6.** Which section of the Kickstart file should you use to specify the installation of software components?

- a.** %post
- b.** %end
- c.** %packages
- d.** %pre

**7.** The Anaconda installer can access a Kickstart file in which three locations? (Choose three.)

- a.** A network server that uses the FTP, HTTP, or NFS protocols.
- b.** A tmux terminal in which the /root/anaconda-ks.cfg file can be edited at installation time.
- c.** A removable USB disk or CD-ROM.
- d.** A local hard disk on the system.

## Solution

---

1. What are the recommended disk and memory requirements to perform a local installation of Red Hat Enterprise Linux 10 on x86-64 architecture?  
 a. 20 GiB of disk space and 3 GiB RAM  
 b. **10 GiB of disk space and 1.5 GiB RAM**  
 c. 10 GiB of disk space and 3 GiB RAM  
 d. 20 GiB of disk space and 1 GiB RAM
2. Which two features regarding root account creation changed in Red Hat Enterprise Linux 10? (Choose two.)  
 a. **The root account is now disabled by default.**  
 b. The default password is redhat and must be changed at first login.  
 c. **SSH access for the root user is disabled by default.**  
 d. The root account cannot be accessed directly, and the only way to execute administrative commands is with sudo privileges.
3. What is a new feature of the Anaconda installer in RHEL 10?  
 a. Mouse support  
 b. Remote VNC access  
 c. Preinstallation scripts  
 d. **Remote RDP access**
4. What is the primary purpose of the Kickstart feature in RHEL?  
 a. Manage user accounts  
 b. Enhance system security  
 c. Improve network performance  
 d. **Fully automate system installation**

5. Which directive marks the end of a section in a Kickstart file?

- a. %stop
- b. %finish
- c. %end
- d. %close

6. Which section of the Kickstart file should you use to specify the installation of software components?

- a. %post
- b. %end
- c. %packages
- d. %pre

7. The Anaconda installer can access a Kickstart file in which three locations? (Choose three.)

- a. **A network server that uses the FTP, HTTP, or NFS protocols.**
- b. A tmux terminal in which the /root/anaconda-ks.cfg file can be edited at installation time.
- c. **A removable USB disk or CD-ROM.**
- d. **A local hard disk on the system.**

## 16.6. Summary

---

In this lesson, you learned to install Red Hat Enterprise Linux 10 interactively and through Kickstart automation.

By using the Anaconda graphical installer, you can interactively set up and configure installation parameters before the process starts.

The Kickstart feature enables you to perform an unattended installation by providing the information that the installer requires in a text file.

## Chapter 17.

# Managing Containers with Podman

---

## Goal

Manage containers and container images with the latest version of container management tools.

## Sections

- Introduction to Containers (and Quiz)
- Running Containers with Podman (and Guided Exercise)
- Creating and Managing Container Images (and Guided Exercise)

## Lab

- Manage Containers with Podman

## 17.1. Introduction to Containers

### Objectives

- Explain container concepts and core technologies in Red Hat Enterprise Linux.

### Container Technology Overview

In computing, a *container* is an encapsulated process that includes the required runtime dependencies for a program to run. In a container, application-specific libraries are independent of the host operating system libraries. The host operating system and kernel provide libraries and functions that are not specific to the containerized application. The provided libraries and functions help to ensure that the container remains compact, and that it can quickly execute and stop as needed.

A container engine creates a union file system by merging container image layers. A *layer* is a component of the container image that represents a specific set of files or changes, such as system libraries, application binaries, or configuration files. Each layer builds on the previous one, forming a stacked structure that makes up the complete image. Because container image layers are immutable, a container engine adds a writable layer for runtime file modifications. Containers are *ephemeral* by default, which means that the container engine removes the writable layer when you remove the container.

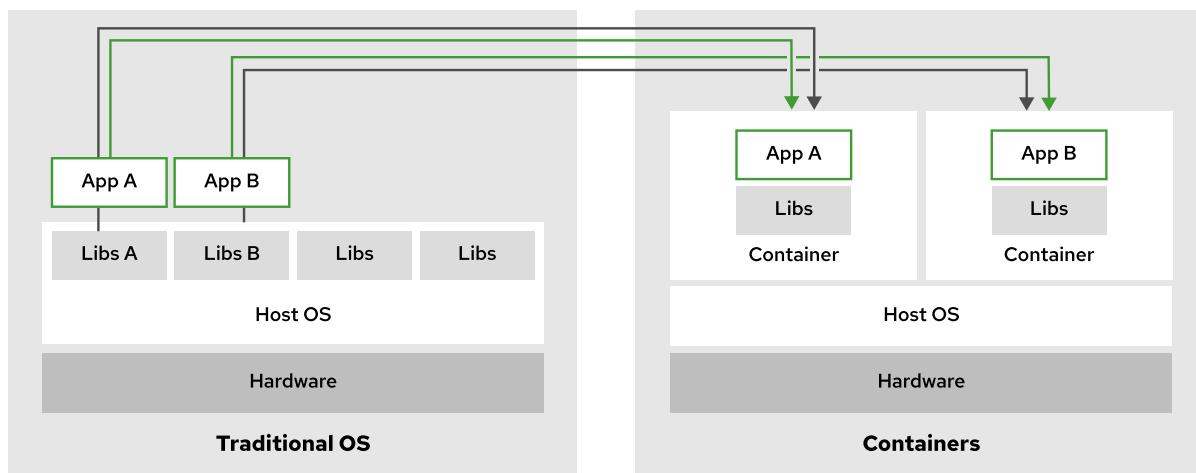


Figure 27. Applications in containers versus on host operating system

The environment within a container is Linux-based, regardless of the host operating system. Containers use Linux kernel features, such as namespaces, control groups (cgroups), SELinux, and secure computing mode (seccomp). For example, containers use cgroups for resource management,

such as CPU time allocation and system memory. Namespaces in particular provide the functionality to isolate processes within containers from each other and from the host system. When using containers on non-Linux operating systems, these Linux-specific features are often virtualized by the container engine implementation.

Containerization originated from technologies such as the `chroot` utility, a method to partially or fully isolate an environment, and evolved to the *Open Container Initiative (OCI)*, which is a governance organization that defines standards for creating and running containers. Most container engines conform to the OCI specifications, so developers can confidently build their deployable target artifacts to run as OCI containers.

Containers use SELinux and secure computing mode to enforce security boundaries and to restrict the features that are available in containers.

#### Note

For a more detailed discussion about container architecture and security, refer to the Red Hat white paper, [A Layered Approach to Container and Kubernetes Security](#).

## Benefits and Challenges of Using Containers

One of the main benefits is that containers can start in seconds, which makes them ideal for dynamic and scalable environments. They are also well suited for deploying modern applications, such as cloud-native applications. They are highly efficient, as they use fewer resources compared to virtual machines. Containers are portable, which means that they run consistently across development, testing, and production environments. Their isolated nature ensures that application environments remain separate and predictable.

However, containers also present some challenges. Data persistence requires additional configuration, such as using volumes to store data outside of the container's ephemeral storage. Networking becomes more complex in configurations that involve multiple containers or multiple hosts. Security management is another concern, as containers share the host's kernel and require careful access control. Although rootless containers improve security by allowing users to run containers without elevated privileges, rootless containers might also limit access to certain system functions.

## Images Versus Instances

*Container images* and *container instances* are commonly used terms when working with containers, and they have different meanings. A *container image* contains immutable data, instructions, and libraries that define an application. You can use container images to create *container instances*, which are

executable versions of the image that include references to networking, disks, and other runtime necessities.

You can use a single container image multiple times to create many distinct container instances. You can also run these instances across multiple hosts. The application within a container is independent of the host environment.

### Note

OCI container images are defined by the `image-spec` specification, whereas OCI container instances are defined by the `runtime-spec` specification.

Another way to think about *container images* versus *container instances* is that an instance relates to an image as an object relates to a class in object-oriented programming.

## Container Registries

A container image registry is a storage location that hosts container images and related artifacts for container-based applications. The `/etc/containers/registries.conf` file is a system-wide configuration file. This file lists the container registries that are available for tools like Podman, Buildah, and Skopeo.

Red Hat provides the following registries:

- `registry.redhat.io` : requires authentication
- `registry.access.redhat.com` : requires no authentication
- `registry.connect.redhat.com` : holds Red Hat Partner Connect program images

## Comparing Containers to Virtual Machines

Containers generally serve a similar role to *virtual machines* (VMs), where an application resides in a self-contained environment with virtualized networking for communication. Despite this initial similarity, containers have a smaller footprint, and start and stop faster than a virtual machine. For both memory and disk usage, VMs are often measured in gigabytes, whereas containers are measured in megabytes.

A VM is useful when an additional full computing environment is required, such as when an application requires specific, dedicated hardware. Additionally, a VM is preferable when an application requires a non-Linux operating system or a different kernel from the host.

## Containers Versus Virtual Machines

Virtual machines and containers use different software for management and functionality. Hypervisors, such as Kernel-based Virtual Machine (KVM), Xen, VMware, and Hyper-V, are applications that provide the virtualization functionality for VMs. The container equivalent of a hypervisor is a container engine, such as Podman.

	Virtual machines	Containers
<b>Machine-level functionality</b>	Hypervisor	Container engine
<b>Management</b>	VM management interface	Container engine or orchestration software
<b>Virtualization level</b>	Fully virtualized environment	Only relevant parts
<b>Size</b>	Measured in gigabytes	Measured in megabytes
<b>Portability</b>	Generally only same hypervisor	Any OCI-compliant engine

You can manage hypervisors with additional management software. The software can be included with the hypervisor, or it can be external, such as Virtual Machine Manager (`virt-manager`) with KVM. Alternatively, you can manage containers directly through the container engine itself. Additionally, you can use container orchestration tools, such as Red Hat OpenShift Container Platform (RHOCP) and Kubernetes, to run and manage containers at scale. RHOCP manages both containers and virtual machines from a common interface.

With VMs, interoperability is uncommon. A VM that runs on one hypervisor is usually not guaranteed to run on a different hypervisor. In contrast, containers that follow the OCI specification do not require a particular container engine to function. Many container engines can function as drop-in replacements for each other.

## References

For more information about containers, refer to the *Introduction to Containers* chapter in the *Building, Running, and Managing Containers* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/building\\_running\\_and\\_managing\\_containers/index#introduction-to-containers](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/building_running_and_managing_containers/index#introduction-to-containers)

[A Layered Approach to Container and Kubernetes Security](#)

## 17.2. Quiz

### Introduction to Containers

---

Choose the correct answers to the following questions:

- 1.** Which statement describes the difference between a container image and a container instance?  
 **a.** An image is a running environment, and an instance is used to build new images.  
 **b.** An image is a static definition, and an instance is the live, executable form.  
 **c.** An image contains system logs, and an instance is a user session.  
 **d.** An image is used to create a single instance, but instances can be reused.
  
- 2.** Which organization governs the standards for containerization?  
 **a.** Docker  
 **b.** Red Hat  
 **c.** Open Container Initiative  
 **d.** Linux Foundation
  
- 3.** Which Linux kernel feature helps to provide isolation and resource control in containers?  
 **a.** Snapshots and virtualization layers  
 **b.** Namespaces and control groups (cgroups)  
 **c.** BIOS and UEFI support  
 **d.** SSH and system services
  
- 4.** What is the purpose of a container image registry?  
 **a.** To virtualize system-level services  
 **b.** To store container logs and network data  
 **c.** To store and distribute container images  
 **d.** To manage root access to containers

**5.** Which item provides machine-level functionality for running containerized applications?

- a.** Container instance
- b.** Container image
- c.** Container engine
- d.** Hypervisor

**6.** What is an executable version of an application that includes references to networking, disks, and other runtime necessities?

- a.** Container instance
- b.** Container image
- c.** Container engine
- d.** Virtual machine

## Solution

1. Which statement describes the difference between a container image and a container instance?  
 a. An image is a running environment, and an instance is used to build new images.  
 b. **An image is a static definition, and an instance is the live, executable form.**  
 c. An image contains system logs, and an instance is a user session.  
 d. An image is used to create a single instance, but instances can be reused.
2. Which organization governs the standards for containerization?  
 a. Docker  
 b. Red Hat  
 c. **Open Container Initiative**  
 d. Linux Foundation
3. Which Linux kernel feature helps to provide isolation and resource control in containers?  
 a. Snapshots and virtualization layers  
 b. **Namespaces and control groups (cgroups)**  
 c. BIOS and UEFI support  
 d. SSH and system services
4. What is the purpose of a container image registry?  
 a. To virtualize system-level services  
 b. To store container logs and network data  
 c. **To store and distribute container images**  
 d. To manage root access to containers
5. Which item provides machine-level functionality for running containerized applications?  
 a. Container instance  
 b. Container image  
 c. **Container engine**  
 d. Hypervisor

**6.** What is an executable version of an application that includes references to networking, disks, and other runtime necessities?

- a. Container instance**
- b. Container image**
- c. Container engine**
- d. Virtual machine**

## 17.3. Running Containers with Podman

### Objectives

- Run an existing container image from a registry as a container by using Podman.

### The Podman Container Management Tool

Podman is an open source tool for managing your containers locally. With Podman, you can find, run, build, or deploy Open Container Initiative (OCI) containers and container images.

Some container tools use a daemon to proxy the requests, which creates a single point of failure for container management tasks. In addition, a daemon might require elevated privileges, which might become a security concern.

Podman is daemonless by design, which means that Podman interacts directly with containers, images, and registries without a daemon. This design makes Podman a suitable tool to use in a production environment.

In Red Hat Enterprise Linux 10, Podman is installed by default. You can verify the Podman version that is installed in your machine by using the `podman -v` command. The version number might differ in your system.

```
user@host:~$ podman -v
podman version 5.4.0
```

Podman provides three ways to interact with your containers: the Podman CLI, the RESTful API, and a desktop application named *Podman Desktop*.

This section focuses on running containers by using the Podman CLI.

### Container Image Registries

A container image is a packaged version of your application, with all the dependencies that are necessary for the application to run. You can download a container image from a container registry, and use this container image as a base to run your containers. Commonly used image registries include Red Hat Registry, Quay.io, Docker Hub, and Amazon Elastic Container Registry (Amazon ECR).

### Red Hat Container Registries

Red Hat distributes container images by using the following registries:

- `registry.redhat.io`: requires authentication
- `registry.access.redhat.com`: requires no authentication
- `registry.connect.redhat.com`: holds Red Hat Partner Connect program images

Red Hat Ecosystem Catalog provides centralized searching for these registries at <https://catalog.redhat.com>. You can use the Ecosystem Catalog to search for images and to get technical details about them. Go to <https://catalog.redhat.com/search> to search for container images.

The screenshot shows the Red Hat Ecosystem Catalog homepage. The top navigation bar includes links for Solutions, Certified products, Artifacts, Partners, and a search bar. Below the search bar are filters for Type (Partners, Cloud, Hardware, Software, Containers, Solutions) and Provider (Presidio, Inc., OI Communique Laboratory Inc., 10Duke, 128 Technology, 1Password). The main content area displays search results for container images. The first result is 'Ansible Operator Base Image' by Red Hat, published 1 hour ago, version v4.15.0-. It is a Container image for openshift4/ose-ansible-operator. The second result is 'The specialized must gather image for Shared Resource CSI Driver support.' by Red Hat, published 1 hour ago, version v4.15.0-. It is a Container image for openshift4/ose-csi-driver-shared-resource-mustgather-rhel8. The third result is 'OpenShift Egress Router' by Red Hat, published 1 hour ago, version v4.15.0-. It is a Container image for openshift4/ose-egress-router. A blue 'Feedback' button is visible on the right side of the page.

Figure 28. Red Hat Ecosystem Catalog

The container image details page provides relevant information about the container image, such as the Containerfile that is used to create the image, the packages that are installed within the image, or a security scanning. You can also change the image version by selecting a specific tag.

## The Quay.io Container Registry

The Red Hat Registry stores only images from Red Hat and certified providers. To store your custom images, you can use the Quay.io registry. Storing public images in Quay.io is free, and paying customers receive further benefits, such as private repositories. Developers can also deploy an on-premise Quay instance, which you can use to set up an image registry on your infrastructure.

You can use your Red Hat developer account to log in to the Quay.io registry.

## Logging in to Container Image Registries

Usually, you provide authentication credentials to interact with a container registry by using the `podman login` command. The `podman login` command accepts the registry URL as an argument and then it requests the username and password.

The following example shows a login attempt to the `registry.redhat.io` registry with a username and password.

```
user@host:~$ podman login registry.redhat.io
Username: provide_username
Password: provide_password
Login Succeeded!
```

### Note

For security reasons, the `podman login` command does not show your password in the interactive session. Although you do not see what you are typing, Podman registers every key stroke. Press `Enter` when you have typed your full password in the interactive session to initiate the login.

Some container registries might not require you to provide an authorized account, which means that you can download container images without providing credentials. In this case, you must provide the URL of the container image registry and leave the username and password information empty.

```
user@host:~$ podman login registry.access.redhat.com
Username: Enter
Password: Enter
Login Succeeded!
```

## Inspecting Configured Container Registries

To view your machine's default container image registry, you can use the `podman info` command. The `podman info` command shows OS and hardware characteristics, configured registries, installed plugins, and versioning information.

The following example shows an excerpt of the locally configured `my_org_registry.example.com` registry. Note that the output shows additional registries under the `search` field that are used to search for container images.

```
user@host:~$ podman info
...output omitted...
registries:
  my_org_registry.example.com:
    Blocked: false
    Insecure: true
    Location: my_org_registry.example.com
    MirrorByDigestOnly: false
    Mirrors: null
    Prefix: my_org_registry.example.com
    PullFromMirror: ""

  search:
    - registry.access.redhat.com
    - registry.redhat.io
    - docker.io
    - quay.io
...output omitted...
```

Podman primarily uses two files to store registry configuration:

- The `/etc/containers/registries.conf` file (or the `/etc/containers/registries.conf.d` drop-in directory). Podman uses the `registries.conf` file to store registry configuration.
- The `$(XDG_RUNTIME_DIR)/containers/auth.json` file. Podman stores credential information in the `$(XDG_RUNTIME_DIR)/containers/auth.json` file, where `$(XDG_RUNTIME_DIR)` refers to a directory that is specific to the current user. The credentials are encoded in the base64 format.

## Running and Managing Containers

After authenticating to a container registry, you can start running containers.

### Running Containers

You use the `podman run` command to run a container. The `podman run` command requires an image name as an argument, which can be stored locally or it can exist remotely in a registry. The `podman run` command downloads or *pulls* the container image and uses it to run a container.

An image might accept additional arguments, such as commands to be run in the container after it starts.

In the following example, you run a container that uses the ubi10 image from the registry.redhat.io registry to run the echo 'Hello World!' command. Podman searches for the image, pulls it, starts the container, and runs the requested command. Note that the command output shows the process of pulling the image from the registry.

```
user@host:~$ podman run registry.redhat.io/ubi10 echo 'Hello World!'
Trying to pull registry.redhat.io/ubi10:latest...
...output omitted...
Hello World!
```

When the container finishes executing the echo command, the container stops because no other process keeps it running.

By default, a container keeps running while it has tasks to run and it retains the terminal prompt until it finishes. You might run containers that run a quick task and then stop, such as the echo command example, or you might run a container that runs indefinitely. To avoid blocking the command prompt, you can use the --detach or -d options.

The following example shows running a detached container that executes the sleep infinity command, which means that the container runs indefinitely. This container uses the same ubi10 image, but this time Podman does not pull the image because it is already stored locally.

```
user@host:~$ podman run -d registry.redhat.io/ubi10 sleep infinity
8699...c98b
```

## Displaying Containers

You can list the running containers by using the podman ps command. By default, the podman ps command lists the following details for your containers:

- The container's ID
- The name of the image that the container uses
- The command that the container executes
- The time that the container was created
- The status of the container
- The exposed ports in the container
- The name of the container

Podman can identify the containers either by the *Universal Unique Identifier (UUID)* short identifier, which is composed of 12 alphanumeric characters, or by the UUID long identifier, which is composed of 64 alphanumeric characters. The following example shows the list of containers that are running in the environment. The first column shows the 8699...1cae string, which is the container's ID, or UUID short identifier.

```
user@host:~$ podman ps
CONTAINER ID IMAGE COMMAND ... NAMES
8699...1cae registry.redhat.io/ubi10:... sleep infinity ... bold_chebyshev
```

The list does not show the container that ran the echo command because the container stopped after running the command. However, *stopping* a container is not the same as *removing* a container.

Although the container is stopped, Podman does not remove it. You can list all containers (running and stopped) by adding the `--all` or `-a` option to the `podman ps` command:

```
user@host:~$ podman ps -a
CONTAINER ID ... COMMAND ... STATUS ... NAMES
fad3...00d3 ... echo Hello World! ... Exited (0) 7 minutes ago ...
nervous_nobel
8699...1cae ... sleep infinity ... Up 53 seconds ...
bold_chebyshev
```

If you do not provide a name for the container during its creation, then Podman generates a random string for the container name. Be sure to define a unique name to simplify the identification of your containers when managing their lifecycle. To assign a name to a container, use the `podman run` command with the `--name` option.

The following example shows running the `which python` command on the `my_python` container that is based on the `ubi9/python-312` image. The container prints `/opt/app-root/bin/python` as the result and then it stops.

```
user@host:~$ podman run --name my_python \
registry.redhat.io/ubi9/python-312 which python
Trying to pull registry.redhat.io/ubi9/python-312:latest...
Getting image source signatures
...output omitted...
Storing signatures
/opt/app-root/bin/python
```

The podman ps command now shows my\_python as the name for the container:

```
user@host:~$ podman ps -a
CONTAINER ID IMAGE                                     COMMAND      ... NAMES
...output omitted...
a57e...fa88 registry.redhat.io/ubi9/python-312:latest which python ...
my_python
```

## Restarting Containers

Execute the podman restart command to restart a running container. You can also use this command to start a stopped container.

The following command restarts a container named nginx:

```
user@host:~$ podman restart nginx
1b98a7c275dd
```

## Stopping and Removing Containers

You can stop a container gracefully by using the podman stop command. When you execute the podman stop command, Podman sends a SIGTERM signal to the container. Processes use the SIGTERM signal to implement clean-up procedures before stopping.

The following example stops a container with a container ID of 1b982aeb75dd:

```
user@host:~$ podman stop 1b982aeb75dd
1b982aeb75dd
```

You can stop all the running containers by using the --all or -a flag. In the following example, the command stops three containers:

```
user@host:~$ podman stop --all
4aea81cc0c2a
6b18d10f54ea
77632e81cbc0
```

If a container does not respond to the SIGTERM signal, then Podman sends a SIGKILL signal to forcefully stop the container. Podman waits 10 seconds by default before sending the SIGKILL signal.

You can send the SIGKILL signal to the container by using the `podman kill` command. In the following example, a container named `httpd` is stopped forcefully:

```
user@host:~$ podman kill httpd
httpd
```

Use the `podman rm` command to remove a stopped container. The following command removes a stopped container with the container ID of `c58cf4b90df`:

```
user@host:~$ podman rm c58cf4b90df
c58cf4b90df
```

By default, you cannot remove running containers; you must first stop the running container and then remove it. You can add the `--force` (or `-f`) flag to remove the container forcefully:

```
user@host:~$ podman rm c58cf4b90df --force
c58cf4b90df
```

You can also automatically remove a container when it exits by adding the `--rm` option to the `podman run` command:

```
user@host:~$ podman run --rm registry.redhat.io/ubi10 \
echo 'Hello World!'
Red Hat
```

After the container prints the string, the container exits and Podman removes the container. The following example shows that the container is not listed after running the `podman ps` command with the `--all` option:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

## Exposing Container Ports

Many containerized applications, such as web servers or databases, must run indefinitely to wait for connections. You must also ensure that these containers can expose their service ports externally through a network protocol.

You can use the `podman run` command with the `-p` option to map a port in your local machine to a port inside the container. This way, the traffic in your local port is forwarded to the port inside the container, which enables you to access the application from your computer.

The following example creates a container that runs an Apache HTTP server by mapping the 8080 port in the `mywebapp.example.com` machine to the 8080 port inside the container. The container is not running in detached mode so that you can view the output in the terminal.

```
user@host:~$ podman run -p 8080:8080 registry.redhat.io/rhel10/httpd-24:latest
...output omitted...
=> sourcing 10-set-mpm.sh ...
=> sourcing 20-copy-config.sh ...
=> sourcing 40-ssl-certs.sh ...
---> Generating SSL key pair for httpd...
[Mon Jun 09 05:30:25.657414 2025] [ssl:warn] [pid 1:tid 1] AH01909:
workstation.lab.example.com:8443:0 server certificate does NOT include an ID
which matches the server name
...output omitted...
[Mon Jun 09 05:30:25.734309 2025] [core:notice] [pid 1:tid 1] AH00094: Command
line: 'httpd -D FOREGROUND'
```

**Ctrl + C**

The container is now serving requests by using the 8080 port in the `mywebapp.example.com` machine. You can access the content of the HTTP server from a machine in the network at `mywebapp.example.com:8080` or directly from the `mywebapp` machine at `localhost:8080`.

```
user@host:~$ curl 127.0.0.1:8080
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
...output omitted...
```

To prevent the terminal from being blocked, run the container in detached mode by using the `podman run` command with the `-d` option:

```
user@host:~$ podman run -d -p 8080:8080 \
registry.redhat.io/rhel10/httpd-24:latest
b7eb...636a
```

## References

podman-login(1), podman-info(1), podman-run(1), podman-ps(1), podman-restart(1),  
podman-stop(1), podman-kill(1) and podman-rm(1) man pages

For more information about running containers, refer to the *Building, Running, and Managing Containers* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/building\\_running\\_and\\_managing\\_containers/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/building_running_and_managing_containers/index)

[Podman Documentation](#)

## 17.4. Guided Exercise

### Run Containers with Podman

Run an existing container image from a registry as a container by using Podman.

#### Outcomes

- Run a container that finishes its task and exits.
- Run a container in detached mode.
- Stop and remove containers.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start containers-podman
```

#### Instructions

1. Run a container that prints the Hello Red Hat! text.

Use the following information to run the container:

Option	Description
Registry	registry.lab.example.com:5000
Image	ubi10/ubi
Username	student
Password	redhat

- 1.1.** Log in to the `registry.lab.example.com:5000` remote registry. Use `student` as the username, and `redhat` as the password.

 **Note**

If you are already logged in to the registry, then you do not need to authenticate again. You can proceed with the next steps.

```
student@workstation:~$ podman login registry.lab.example.com:5000
Username: student
Password: redhat
Login Succeeded!
```

- 1.2.** Use the `registry.lab.example.com:5000/ubi10/ubi` image to create a container. Provide the echo 'Hello Red Hat!' command as the argument to be executed inside the container.

```
student@workstation:~$ podman run registry.lab.example.com:5000/ubi10/ubi \
echo 'Hello Red Hat!'
...output omitted...
Hello Red Hat!
```

- 2.** Verify that the container is no longer running, and then remove it.

- 2.1.** Verify that the container is not running after printing the message.

student@workstation:~\$ podman ps						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES

- 2.2.** Verify that the container is listed as stopped.

Note the container ID to use in a future step.

The container ID might vary in your environment.

```
student@workstation:~$ podman ps -a
CONTAINER ID IMAGE                                COMMAND          ... STATUS    ...
9556df127169 .../ubi10/ubi:latest              echo Hello Red Hat! ... Exited (0) ...
```

- 2.3.** Remove the stopped container by using its ID.

Use the container ID from your environment.

```
student@workstation:~$ podman rm 9556df127169
9556df127169
```

## 2.4. Verify that the container is removed.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

3. Create a container named `my_webserver` that hosts a web server on the 8080 port of the workstation machine. Use the `registry.lab.example.com:5000/rhel10/httpd-24` image. Verify that you can access the web server from a web browser.

- 3.1. Create the `my_webserver` container and expose the 8080 port of the container to the 8080 port of the workstation machine. Use the `registry.lab.example.com:5000/rhel10/httpd-24` image.

After the container finishes creating, the container keeps running and the logs are displayed in the terminal.

```
student@workstation:~$ podman run \
--name my_webserver \
-p 8080:8080 \
registry.lab.example.com:5000/rhel10/httpd-24
Trying to pull registry.redhat.io/rhel10/httpd-24:latest...
...output omitted...
=> sourcing 10-set-mpm.sh ...
=> sourcing 20-copy-config.sh ...
=> sourcing 40-ssl-certs.sh ...
---> Generating SSL key pair for httpd...
[Mon Jun 09 05:52:28.024389 2025] [ssl:warn] [pid 1:tid 1] AH01909:
workstation.lab.example.com:8443:0 server certificate does NOT include an ID
which matches the server name
...output omitted...
[Mon Jun 09 05:52:28.106333 2025] [core:notice] [pid 1:tid 1] AH00094: Command
line: 'httpd -D FOREGROUND'
```

- 3.2. In a web browser, go to the `http://localhost:8080` page. Verify that the HTTP server is running at the 8080 port.

The web page shows a boilerplate message, which confirms that the web server is running.

- 3.3.** In the command-line terminal, press **Ctrl + C** to stop the execution of the container.

```
...output omitted...
Ctrl + C
[Mon Jun 09 05:54:37.710852 2025] [mpm_event:notice] [pid 1:tid 1] AH00491:
caught SIGTERM, shutting down
```

- 3.4.** Delete the container by using the container name.

```
student@workstation:~$ podman rm my_webserver
my_webserver
```

- 4.** Create the `my_webserver` container again, and run the container in detached mode.

```
student@workstation:~$ podman run -d \
--name my_webserver \
-p 8080:8080 \
registry.lab.example.com:5000/rhel10/httpd-24
c5ca...e30d
```

- 5.** Verify that the container is running in the background (in detached mode) and that the web server is serving requests.

- 5.1.** Verify that the container is running in the background.

```
student@workstation:~$ podman ps
... COMMAND           ... STATUS          PORTS          NAMES
... /usr/bin/run-http... Up 3 minutes 0.0.0.0:8080->8080/tcp... my_webserver
```

- 5.2.** Verify that the web server is serving requests.

```
student@workstation:~$ curl 127.0.0.1:8080
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
...output omitted...
```

- 6.** Stop the `my_webserver` container, and then remove it.

- 6.1.** Stop the `my_webserver` container.

```
student@workstation:~$ podman stop my_webserver  
my_webserver
```

**6.2.** Remove the my\_webserver container.

```
student@workstation:~$ podman rm my_webserver  
my_webserver
```

**6.3.** Verify that no containers are running or stopped.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish containers-podman
```

## 17.5. Creating and Managing Container Images

### Objectives

- Manage downloaded container images and create a simple container image.

### Containers and Container Images

Recall that a *container* is an isolated runtime environment where applications are executed as isolated processes. This isolation of the runtime environment ensures that applications do not interrupt any other containers or system processes. A *container image* contains a packaged version of your application, along with all the dependencies that the application requires to run. Images can exist without containers; however, containers require container images to build a runtime environment and to execute applications.

### Working with Container Images

The Podman tool performs many operations on container images.

- Pull an image from a registry
- List images in a registry
- Inspect a local or remote image's contents
- Copy images or image layers
- Tag an image with an additional name
- Save or load an image to or from a container archive
- Share images through a registry

### Searching, Pulling, and Listing Images

You can run a container to deploy an application without explicitly pulling a container image. In some cases, you might not be sure which images are available in the container registry or you are not sure which image to use.

You can search for images in a registry by using the `podman search` command followed by the URL of the registry. Make sure that the URL that you define includes a trailing forward slash (/).

```
user@host:~$ podman search registry.redhat.io/rhel10
NAME                                     DESCRIPTION
registry.redhat.io/rhel10/buildah          rhcc_registry.access...
registry.redhat.io/rhel10/toolbox          rhcc_registry.access...
registry.redhat.io/rhel10/net-snmp        rhcc_registry.access...
...output omitted...
```

Red Hat offers a Universal Base Image (UBI) as a foundation for new container applications. Each Red Hat UBI is compliant with the Open Container Initiative (OCI) and is freely redistributable. To search for a Universal Base Image, add the `ubi` keyword to the `podman search` command:

```
user@host:~$ podman search registry.redhat.io/ubi
NAME                                     DESCRIPTION
registry.redhat.io/ubi8/ubi               Provides the latest release of th...
registry.redhat.io/ubi9/ubi               rhcc_registry.access.redhat.com_u...
registry.redhat.io/ubi10/ubi              rhcc_registry.access.redhat.com_u...
...output omitted...
```

You can fetch container images from image registries by using the `podman pull` command. The containerized RHEL 10 image is referenced in the form of `name:tag`, where `name` is the name of the image and `tag` is the tag (also known as the *version*) of the image.

Tags identify different versions of an image. Often, tags are specific version numbers, such as `10.0`. The `latest` tag refers to the latest version of an image. When no tag is provided, Podman uses the `latest` tag by default.

For example, the following command pulls a containerized version of the latest Red Hat Enterprise Linux 10. The containerized version is a bootable container (`bootc`) image.

```
user@host:~$ podman pull registry.redhat.io/rhel10/rhel-bootc
...output omitted...
a93f...d5b0
```

Note that the version is not specified, so the `latest` version is automatically pulled. To pull a specific version of the image, specify the exact version, for example:

```
user@host:~$ podman pull registry.redhat.io/rhel10/rhel-bootc:10.0
...output omitted...
a93f...d5b0
```

After you run the `pull` command, the image is stored locally. You can list all the images in your system by using the `podman images` command:

```
user@host:~$ podman images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
registry...rhel10/rhel-bootc  10.0    a93f5ef0baa4  3 weeks ago  1.43 GB
registry...rhel10/rhel-bootc  latest   a93f5ef0baa4  3 weeks ago  1.43 GB
```

## Inspecting Images

You can inspect the contents of a container image by using the `podman image inspect` command. The contents are rendered in JSON format.

```
user@host:~$ podman image inspect rhel-bootc:latest
[
  {
    "Id": "a93f...d5b0",
    "Digest": "sha256:510f...c913",
    "RepoTags": [
      "registry.redhat.io/rhel10/rhel-bootc:10.0",
      "registry.redhat.io/rhel10/rhel-bootc:latest"
    ],
    "RepoDigests": [
      "registry.redhat.io/rhel10/rhel-bootc@sha256:510f...c913",
      "registry.redhat.io/rhel10/rhel-bootc@sha256:8578...7556"
    ],
    ...output omitted...
  }
]
```

By using the `--format` option with a Go template, you can inspect the values of specific keys.

The following example returns the name of the container image:

```
user@host:~$ podman image inspect rhel-bootc:latest \
--format "{{.Config.Labels.name}}"
rhel10/rhel-bootc
```

The following example returns the creation date and time of the container image:

```
user@host:~$ podman image inspect rhel-bootc:latest \
--format "{{.Created}}"
2025-05-15 15:00:38.956441305 +0000 UTC
```

The following example returns the description of the container image:

```
user@host:~$ podman image inspect rhel-bootc:latest \
--format "{{.Config.Labels.description}}"
RHEL bootc container
```

To inspect a remote image, first pull the image from the remote repository, for example:

```
user@host:~$ podman image pull registry.redhat.io/rhel10/rhel-bootc:latest
...output omitted...
a93f...d5b0
```

Next, use the `podman image inspect` command to inspect the pulled image, for example:

```
user@host:~$ podman image inspect rhel-bootc:latest
[
  {
    "Id": "a93f...d5b0",
    "Digest": "sha256:510f...c913",
    "RepoTags": [
      "registry.redhat.io/rhel10/rhel-bootc:latest",
      "registry.redhat.io/rhel10/rhel-bootc:10.0"
    ],
    "RepoDigests": [
      "registry.redhat.io/rhel10/rhel-bootc@sha256:510f...c913",
      "registry.redhat.io/rhel10/rhel-bootc@sha256:8578...7556"
    ],
    ...
  ...
]
```

### Note

You can also omit the content type for the `podman inspect` command. If the type is not defined, then Podman inspects containers, images, volumes, network, and pods in that order.

## Tagging Images

You can assign a new name and tag to a container image by using the `podman tag` command, followed by the current image ID, and the new container image name and tag to use.

```
user@host:~$ podman tag c6222576494f fedora:latest
```

## Creating an Image from a Containerfile

You can create a container image by using a Containerfile. A Containerfile contains all the instructions that are required to assemble an image.

All Containerfile files start with a base image as their foundation. In this example, the Universal Base Image `ubi10` is pulled from the `registry.redhat.io` repository to local storage:

```
user@host:~$ podman image pull registry.redhat.io/ubi10/ubi
...output omitted...
da86...50d1
```

The additional libraries, binaries, and files that are required to run an application can then be layered on top of this base image. The following Containerfile uses the ubi10 base image, installs the httpd web server package, copies an index.html file to the web server root directory, exposes port 80 for the httpd process, and sets the entrypoint to run the httpd process in the foreground.

```
# Use the ubi10 base image
FROM ubi10/ubi

# Install httpd
RUN dnf install -y httpd

# Copy a temporary index.html file to the web server root
COPY index.html /var/www/html/index.html

# Expose port 80 for httpd
EXPOSE 80

# Set the entrypoint to run httpd in the foreground
ENTRYPOINT ["/usr/sbin/httpd", "-DFOREGROUND"]
```

Use the podman build command with the Containerfile file to create the actual container image.

```
user@host:~$ podman build -t my-httpd -f Containerfile
STEP 1/5: FROM ubi10/ubi
STEP 2/5: RUN dnf install -y httpd
--> Using cache 84ad...b195
--> 84ada70bc51a
STEP 3/5: COPY index.html /var/www/html/index.html
--> cd70d8bde15a
STEP 4/5: EXPOSE 80
--> 4df4dd4a9158
STEP 5/5: ENTRYPOINT ["/usr/sbin/httpd", "-DFOREGROUND"]
COMMIT my-httpd
--> 3067a8d97686
Successfully tagged localhost/my-httpd:latest
3067...1366
```

Use the podman image list command to list images that are in storage.

```
user@host:~$ podman image list
REPOSITORY                                     TAG      ...  SIZE
localhost/my-htt...                         latest   ...  273 MB
...output omitted...
```

Use the `podman run` command to create a running instance of a container image with an ephemeral, writable layer.

```
user@host:~$ podman run my-htt...
```

## Sharing Images Through a Registry

When an image is ready to be shared with others, use the `podman image push` command to push an image, a manifest list, or an image index to a registry.

The following command pushes the `my-htt`d image to the `registry.redhat.io` registry and places it at the root of the registry:

```
user@host:~$ podman image push my-htt... registry.redhat.io/my-htt...
```

## Deleting and Pruning Images

The `podman rmi` command removes an image, as well as any untagged and unreferenced parent images, from local storage.

The following command removes the image with an ID of `4e138cd2375b`:

```
user@host:~$ podman rmi 4e138cd2375b
```

The `podman image prune` command removes all images from local storage that are without tags and that are not referenced by other images. These images are also referred to as *dangling* images.

The following command prunes all dangling images from local storage:

```
user@host:~$ podman image prune
```

To delete all images that are not used by any container, add the `--all` option.

## References

`podman(1)`, `podman-build(1)`, `podman-image-inspect(1)`, `podman-image-prune(1)`, `podman-images(1)`, `podman-inspect(1)`, `podman-pull(1)`, `podman-push(1)`, `podman-rmi(1)`, `podman-run(1)`, `podman-search(1)`, and `podman-tag(1)` man pages

For more information about containers in RHEL 10, refer to the various *Containers* sections in the *Release Notes for Red Hat Enterprise Linux 10.0* at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/10.0\\_release\\_notes/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/10.0_release_notes/index)

For more information about container image types, refer to the *Types of Container Images* chapter in the *Building, Running, and Managing Containers* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/building\\_running\\_and\\_managing\\_containers/index#types-of-container-images](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/building_running_and_managing_containers/index#types-of-container-images)

For more information about working with container images, refer to the *Working with Container Images* chapter in the *Building, Running, and Managing Containers* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/building\\_running\\_and\\_managing\\_containers/index#working-with-container-images](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/building_running_and_managing_containers/index#working-with-container-images)

## 17.6. Guided Exercise

### Create and Manage Container Images

---

Manage downloaded container images and create a simple container image.

#### Outcomes

- Create a simple container image by using a Containerfile.
- Create and run a container by using the container image.
- Push a container image to a local registry.
- Edit a container image configuration.
- Inspect container images.
- Remove container images.

#### Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start containers-image
```

#### Instructions

1. Log in to the `registry.lab.example.com:5000` remote registry and search for available images.
  - 1.1. Log in to the `registry.lab.example.com:5000` registry. Use `student` as the username, and `redhat` as the password.

##### Note

If you are already logged in to the registry, then you do not need to authenticate again. You can proceed with the next steps.

```
student@workstation:~$ podman login registry.lab.example.com:5000
Username: student
Password: redhat
Login Succeeded!
```

**1.2.** Search the remote registry for available ubi images.

```
student@workstation:~$ podman search registry.lab.example.com:5000/ubi
NAME                                     DESCRIPTION
registry.lab.example.com:5000/ubi10/ubi
registry.lab.example.com:5000/ubi8/ubi
```

- 2.** Create a container image named `my_image` by using a `Containerfile`, and assign the `1.0` tag. Use the `ubi8/ubi` image as the base image, and configure the container to print the `This container uses the ubi8/ubi image` message.

**2.1.** Create the `~/my_image` directory.

```
student@workstation:~$ mkdir my_image
```

- 2.2.** Create the `~/my_image/Containerfile` file. In the first half of the file, add `registry.lab.example.com:5000/ubi8/ubi` as the base image.

```
# Set the base image
FROM registry.lab.example.com:5000/ubi8/ubi
```

- 2.3.** In the second half of the file, configure the container to print the `This container uses the ubi8/ubi image` message on standard output.

```
# Print a message to standard output
CMD echo "This container uses the ubi8/ubi image"
```

- 2.4.** The completed `Containerfile` must match the following content:

```
# Set the base image
FROM registry.lab.example.com:5000/ubi8/ubi

# Print a message to standard output
CMD echo "This container uses the ubi8/ubi image"
```

- 2.5.** Save and close the `Containerfile`.

- 2.6.** Create the container image from the Containerfile. Use `1.0` as the tag of the image.

```
student@workstation:~$ podman build -t my_image:1.0 /home/student/my_image/.  
STEP 1/2: FROM registry.lab.example.com:5000/ubi8/ubi  
...output omitted...  
STEP 2/2: CMD echo "This container uses the ubi8/ubi image"  
COMMIT my_image:1.0  
--> 2970cf11e4d0  
Successfully tagged localhost/my_image:1.0  
2970...c308
```

- 2.7.** Verify that the image with the `1.0` tag exists in local storage.

```
student@workstation:~$ podman images  
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE  
localhost/my_image  1.0      2970cf11e4d0  8 seconds ago  235 MB  
...output omitted...
```

- 3.** Use the local `my_image:1.0` image to create and run a container. Verify that the message `This container uses the ubi8/ubi image` prints on standard output.

- 3.1.** Use the `podman run` command to create a container from the local `my_image:1.0` image.

Verify that the message `This container uses the ubi8/ubi image` prints on standard output.

```
student@workstation:~$ podman run my_image:1.0  
This container uses the ubi8/ubi image
```

- 3.2.** Verify that the container exited. Use the `--format` option to show only essential information.

Note the container ID, which you use in a future step. The container ID might differ on your machine.

```
student@workstation:~$ podman ps -a \  
--format "table {{.ID}}\t{{.Image}}\t{{.Status}}"  
CONTAINER ID  IMAGE           STATUS  
36e9d348661c  localhost/my_image:1.0  Exited (0) 8 seconds ago
```

- 4.** Delete the container by using the container ID, and verify that the container is deleted.

- 4.1.** Remove the exited container. Use your container ID, which you retrieved in a previous step.

```
student@workstation:~$ podman rm 36e9d348661c
36e9d348661c
```

**4.2.** Verify that the container is removed from local storage.

```
student@workstation:~$ podman ps -a
CONTAINER ID  IMAGE      COMMAND     CREATED      STATUS      PORTS      NAMES
```

- 5.** Push the `my_image` container image to the `registry.lab.example.com:5000` remote registry and assign the `1.0` tag.

**5.1.** Push the `my_image` container image to the remote registry. Set the image tag to `1.0`.

```
student@workstation:~$ podman image push \
localhost/my_image:1.0 \
registry.lab.example.com:5000/my_image:1.0
...output omitted...
Writing manifest to image destination
```

**5.2.** Verify that the `my_image` container image is present in the remote registry.

```
student@workstation:~$ podman search registry.lab.example.com:5000/my_image
NAME                                     DESCRIPTION
registry.lab.example.com:5000/my_image
```

- 6.** Create a second container image named `my_image` by modifying the existing `~/my_image/Containerfile` file, and assign the `1.1` tag. Use the `ubi10/ubi` image as the base image. Configure the container to print the `This container uses the ubi10/ubi image` message. Verify that two versions of the `my_image` container image exist in local storage with different image tags.

**6.1.** Update the contents of the `~/my_image/Containerfile` file. In the first half of the file, add `registry.lab.example.com:5000/ubi10/ubi` as the base image.

```
# Set the base image
FROM registry.lab.example.com:5000/ubi10/ubi
```

**6.2.** In the second half of the file, configure the container to print the `This container uses the ubi10/ubi image` message on standard output.

```
# Print a message to stdout
CMD echo "This container uses the ubi10/ubi image"
```

**6.3.** The completed Containerfile must match the following content:

```
# Set the base image
FROM registry.lab.example.com:5000/ubi10/ubi

# Echo a message to stdout
CMD echo "This container uses the ubi10/ubi image"
```

**6.4.** Save and close the Containerfile.

**6.5.** Create the container image from the Containerfile. Define 1.1 as the tag of the image.

```
student@workstation:~$ podman build -t my_image:1.1 /home/student/my_image/.
STEP 1/2: FROM registry.lab.example.com:5000/ubi10/ubi
...output omitted...
STEP 2/2: CMD echo "This container uses the ubi10/ubi image"
COMMIT my_image:1.1
--> eaa23f9aa253
Successfully tagged localhost/my_image:1.1
eaa2...0a5f
```

**6.6.** Verify that two versions of the my\_image container image exist in local storage, one with the 1.0 tag, and one with the 1.1 tag.

```
student@workstation:~$ podman images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
localhost/my_image  1.1      eaa23f9aa253  42 seconds ago  216 MB
localhost/my_image  1.0      2970cf11e4d0  6 minutes ago   235 MB
...output omitted...
```

**7.** Use the local my\_image:1.1 image to create and run a container. Verify that the This container uses the ubi10/ubi image message prints on standard output.

**7.1.** Use the podman run command to create a container from the local my\_image:1.1 image. Verify that the message This container uses the ubi10/ubi image prints on standard output.

```
student@workstation:~$ podman run my_image:1.1
This container uses the ubi10/ubi image
```

**7.2.** Verify that the container exited. Use the `--format` option to show only essential information.

Note the container ID, which you use in a future step. The container ID might differ on your machine.

```
student@workstation:~$ podman ps -a \
--format "table {{.ID}}\t{{.Image}}\t{{.Status}}"
CONTAINER ID IMAGE STATUS
f1e565f3e550 localhost/my_image:1.1 Exited (0) 7 seconds ago
```

**8.** Delete the container and verify that the container is deleted.

**8.1.** Remove the exited container. Use your container ID, which you retrieved in a previous step.

```
student@workstation:~$ podman rm f1e565f3e550
f1e565f3e550
```

**8.2.** Verify that the container is removed from local storage.

```
student@workstation:~$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

**9.** Push the `my_image` container image to the `registry.lab.example.com:5000` remote registry, and assign the `1.1` tag. Verify that two versions of the `my_image` container image exist in the remote registry, one with the `1.0` tag, and one with the `1.1` tag.

**9.1.** Push the `my_image` container image to the remote registry. Set the image tag to `1.1`.

```
student@workstation:~$ podman image push \
localhost/my_image:1.1 \
registry.lab.example.com:5000/my_image:1.1
Getting image source signatures
...output omitted...
Writing manifest to image destination
```

**9.2.** Verify that two versions of the `my_image` container image exist in the remote registry, one with the `1.0` tag, and one with the `1.1` tag. Use the `--list-tags` option and specify the image name `my_image` to list all tag values.

```
student@workstation:~$ podman search --list-tags \
registry.lab.example.com:5000/my_image
NAME                                     TAG
registry.lab.example.com:5000/my_image  1.0
registry.lab.example.com:5000/my_image  1.1
```

- 10.** Inspect the two versions of the my\_image container image. Examine the Cmd section of each image.

- 10.1.** Use the podman image inspect command to inspect the local my\_image:1.0 container image. Examine the Cmd section.

```
student@workstation:~$ podman image inspect my_image:1.0
...output omitted...
    "Cmd": [
        "/bin/sh",
        "-c",
        "echo \"This container uses the ubi8/ubi image\""
    ],
...output omitted...
```

- 10.2.** Use the podman image inspect command to inspect the local my\_image:1.1 container image. Examine the Cmd section.

```
student@workstation:~$ podman image inspect my_image:1.1
...output omitted...
    "Cmd": [
        "/bin/sh",
        "-c",
        "echo \"This container uses the ubi10/ubi image\""
    ],
...output omitted...
```

- 11.** Remove both versions of the my\_image image from local storage.

- 11.1.** Use the podman rmi command to remove both versions of the my\_image image from local storage.

```
student@workstation:~$ podman rmi my_image:1.0 my_image:1.1
Untagged: localhost/my_image:1.0
Untagged: localhost/my_image:1.1
Deleted: 2970...c308
Deleted: eaa2...0a5f
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish containers-image
```

## 17.7. Lab

# Manage Containers with Podman

Create, run, and build containers and container images with the latest version of container management tools.

## Outcomes

- Build a container image from a Containerfile.
- Push a container image to a remote registry.
- Run a detached container and expose a port to your local machine.

## Prerequisites

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start containers-review
```

## Instructions

1. Log in to the `registry.lab.example.com:5000` remote registry. Use `student` as the username, and `redhat` as the password.
2. Search the remote registry for an HTTP image to run a web server, and then run a detached container by using the image.
3. Create the `custom_app` image with `1.0` as the tag by using a Containerfile present in the `~/custom_app` directory. Use the `rhel10/httpd-24` image as the base image, and add the `This is my containerized webserver.` text to the `/var/www/html/index.html` file.
4. Push the `custom_app:1.0` container image to the `registry.lab.example.com:5000` remote registry, and assign the `1.0` tag. Delete the image locally.
5. Run a detached container named `my_app` that uses the `custom_app` image. Expose the `8080` port of the container to the `8080` port in the workstation machine. Verify that the container runs the web server on the `8080` port of the workstation machine, and displays the `This is my containerized webserver.` text.

### Important

Do not exit the SSH session that you use to create the container. Containers do not persist after you exit the SSH session. Exiting the SSH session before you grade this activity will result in failed tasks. Configuring persistent containers is out of the scope of this lesson.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade containers-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish containers-review
```

## Solution

---

Create, run, and build containers and container images with the latest version of container management tools.

## Outcomes

- Build a container image from a Containerfile.
- Push a container image to a remote registry.
- Run a detached container and expose a port to your local machine.

## Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start containers-review
```

## Instructions

1. Log in to the `registry.lab.example.com:5000` remote registry. Use `student` as the username, and `redhat` as the password.
  - 1.1. Log in to the `registry.lab.example.com:5000` registry. Use `student` as the username and `redhat` as the password.



### Note

If you are already logged in to the registry, then you do not need to authenticate again. You can proceed with the next steps.

```
student@workstation:~$ podman login registry.lab.example.com:5000
Username: student
Password: redhat
Login Succeeded!
```

2. Search the remote registry for an HTTP image to run a web server, and then run a detached container by using the image.
  - 2.1. Search the `registry.lab.example.com:5000` registry for an HTTP image.

The classroom registry includes the `rhel10/httpd-24` image that contains the `httpd` service.

```
student@workstation:~$ podman search registry.lab.example.com:5000/
NAME                                     DESCRIPTION
...output omitted...
registry.lab.example.com:5000/rhel10/httpd-24
registry.lab.example.com:5000/rhel10/rhel-bootc
registry.lab.example.com:5000/ubi10/ubi
registry.lab.example.com:5000/ubi8/ubi
```

## 2.2. Pull the matching container image to your machine.

```
student@workstation:~$ podman pull registry.lab.example.com:5000/rhel10/httpd-24
...output omitted...
Writing manifest to image destination
53d5...720d
```

## 2.3. Verify that the HTTP image is listed in your machine.

```
student@workstation:~$ podman images
REPOSITORY                      TAG      IMAGE ID   CREATED    SIZE
registry...m:5000/rhel10/httpd-24  latest   53d5db9e1d42  2 months ago  326
MB
```

## 3. Create the `custom_app` image with `1.0` as the tag by using a `Containerfile` present in the `~/custom_app` directory. Use the `rhel10/httpd-24` image as the base image, and add the `This is my containerized webserver.` text to the `/var/www/html/index.html` file.

### 3.1. Create the `~/custom_app` directory.

```
student@workstation:~$ mkdir custom_app
```

### 3.2. Create the `~/custom_app/Containerfile` file with the following content.

The `Containerfile` adds a customized message to the web server main page.

```
FROM registry.lab.example.com:5000/rhel10/httpd-24
RUN echo "This is my containerized webserver." > /var/www/html/index.html
```

### 3.3. Create the container image from the container file. Define `1.0` as the tag of the image.

```
student@workstation:~$ podman build -t custom_app:1.0 /home/student/custom_app/.
STEP 1/2: FROM registry.lab.example.com:5000/rhel10/httpd-24
STEP 2/2: RUN echo "This is my containerized webserver." >
/var/www/html/index.html
COMMIT custom_app:1.0
--> f8f36fe8e2ed
Successfully tagged localhost/custom_app:1.0
f8f3...a324
```

### 3.4. Verify that the container image exists locally.

```
student@workstation:~$ podman images
REPOSITORY                      TAG      IMAGE ID      CREATED        SIZE
localhost/custom_app             1.0      f8f36fe8e2ed  17 seconds ago  326 MB
...output omitted...
```

- Push the `custom_app:1.0` container image to the `registry.lab.example.com:5000` remote registry, and assign the `1.0` tag. Delete the image locally.

#### 4.1. Push the `custom_app:1.0` container image to the remote registry.

```
student@workstation:~$ podman image push \
localhost/custom_app:1.0 \
registry.lab.example.com:5000/custom_app:1.0
...output omitted...
Writing manifest to image destination
```

#### 4.2. Verify that the image is present in the remote registry.

```
student@workstation:~$ podman search registry.lab.example.com:5000/
NAME                                     DESCRIPTION
registry.lab.example.com:5000/custom_app
...output omitted...
```

#### 4.3. Remove the `custom_app` image locally.

```
student@workstation:~$ podman rmi localhost/custom_app:1.0
Untagged: localhost/custom_app:1.0
Deleted: f8f3...a324
```

- Run a detached container named `my_app` that uses the `custom_app` image. Expose the `8080` port of the container to the `8080` port in the workstation machine. Verify that the container runs

the web server on the 8080 port of the workstation machine, and displays the This is my containerized webserver. text.

### Important

Do not exit the SSH session that you use to create the container. Containers do not persist after you exit the SSH session. Exiting the SSH session before you grade this activity will result in failed tasks. Configuring persistent containers is out of the scope of this lesson.

- 5.1.** Run a detached container named my\_app that uses the custom\_app image. Expose the 8080 port of the container to the 8080 port in the workstation machine.

```
student@workstation:~$ podman run -d \
-p 8080:8080 \
--name my_app \
registry.lab.example.com:5000/custom_app:1.0
Trying to pull registry.lab.example.com:5000/custom_app:1.0...
...output omitted...
Writing manifest to image destination
f033...632c
```

- 5.2.** Verify that the my\_app container is running.

```
student@workstation:~$ podman ps
... IMAGE ... STATUS ... NAMES
... registry...com:5000/custom_app:latest ... Up 11 seconds ... my_app
```

- 5.3.** Verify that the container displays the This is my containerized webserver. content on the 8080 port of the workstation machine.

```
student@workstation:~$ curl localhost:8080
This is my containerized webserver.
```

## Evaluation

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

```
student@workstation:~$ lab grade containers-review
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish containers-review
```

## 17.8. Summary

---

In this lesson, you learned to manage container images and to run containers.

You can obtain container images from trusted registries, such as the `registry.redhat.io` and `registry.access.redhat.com` registries that Red Hat provides.

By using the `podman` command, you can search for, create, and delete container images, and run detached containers with their ports exposed to the host.

## **Chapter 18.**

### **Working with Image-based Red Hat Enterprise Linux**

---

#### **Goal**

Create, use, install, and upgrade containers and servers that use image-based installation management.

#### **Sections**

- Image Mode for Red Hat Enterprise Linux (and Quiz)
- Creating Installable Images for Image Mode (and Guided Exercise)
- Installing Red Hat Enterprise Linux by Using Image Mode (and Guided Exercise)
- Managing Image Mode-based Systems (and Guided Exercise)

## 18.1. Image Mode for Red Hat Enterprise Linux

### Objectives

- Compare and contrast image mode for Red Hat Enterprise Linux with package mode installations.

### Introduction to Image Mode

Image mode for Red Hat Enterprise Linux 10 is a new deployment and management approach that uses container-native tools to build, deploy, and manage the operating system. With this method, you can deploy the operating system from a *bootable container* (bootc) image.

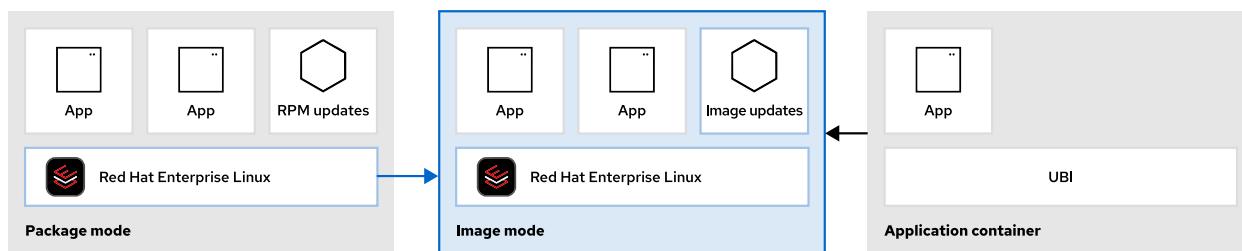


Figure 29. Using container images to update the operating system

Container technology has existed for many years and has proven successful at deploying applications. The same efficient technique of using layered images can be applied to entire hosts. However, the base image is more complex than the ones that application containers use. A bootable container image must include the kernel, a boot loader, and other operating system components that are typically excluded from application containers.

Red Hat provides bootc base images for AMD/Intel x86\_64 and the 64-bit ARM architectures. You can build your own derived image from the base image by adding your chosen application software and its dependencies.

### The Image Mode Workflow

Image mode introduces a container-native approach to system administration. Building a bootc container image is the foundation. This image is then deployed to various target systems. Management operations that follow the initial deployment involve updating the original container image and target systems fetching the new operating system updates from the registry.

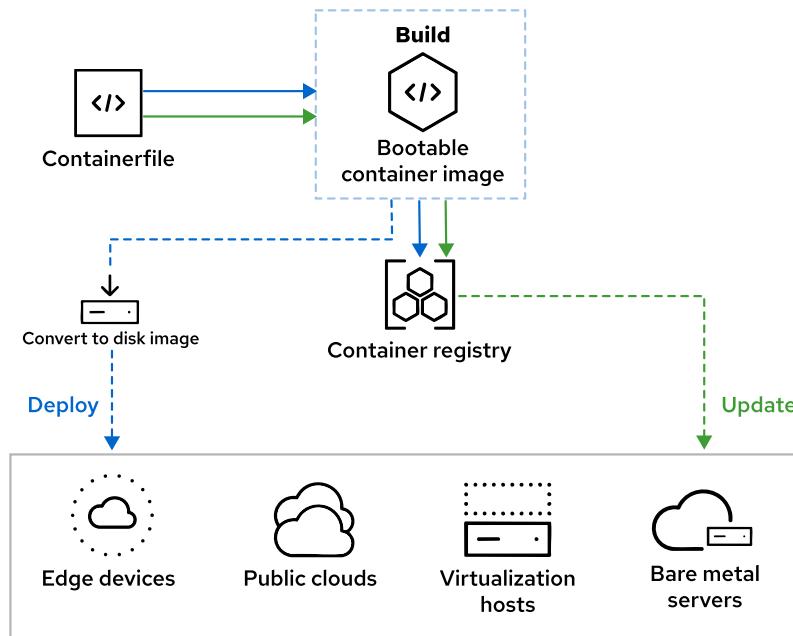


Figure 30. The image mode workflow

## Build

You can define your entire operating system by using a standard Containerfile. The file specifies the base bootc image to use as the starting point and provides additional build instructions for the image as needed. You can use the Podman tool to build and tag the container image.

## Deploy

When the build is complete, the bootable container image must be pushed to a registry that target systems can access. The registry is the source of truth during initial deployment as well as during continued management of your systems.

The Anaconda installer for Red Hat Enterprise Linux supports image mode to deploy to bare metal as well as to virtual machine targets by using Kickstart automation.

For cloud environments, Red Hat provides the `bootc-image-builder` containerized tool to help create disk images of various types from bootc images. Supported formats include: qcow2 (QEMU), ami (Amazon), vmdk (vSphere), and gce (Google).

## Manage

On systems that are deployed in image mode, the `bootc` command-line tool can manually fetch updates that are delivered in the form of container images and install them on the system. By default, target systems automatically update themselves when a later version of their bootc image is tagged in the container registry. Because the updates are transactional, a reboot is always required. The `bootc`

command also supports rollbacks by updating the boot-loader entry to point to a previous installed deployment.

## Benefits of Image Mode

Compared to traditional package mode, image mode offers several advantages.

### Limited infrastructure drift

Infrastructure drift is a challenge in today's cloud environments, when systems gradually deviate from the intended state due to manual configuration changes. Image mode limits infrastructure drift by using well-established container tools and techniques.

### Immutable operating system

In image mode, the root file system is immutable by default and application updates are atomic. The /etc and /var directories are designated to store mutable machine local data. The immutable design enhances stability and security.

### System definition committed to version control

A Containerfile is now a simplified blueprint for the operating system, to include system configuration and application files. Bootable containers are built by using existing tools such as Podman.

### Better scalability

Container images are fast to build and later updates are efficiently stored in additional image layers. An organization can maintain a handful of images and deploy them on many systems in large data center environments.

### Simplified troubleshooting

System updates are atomic and are delivered in the form of updated container images that can be rolled back if needed.

## References

For further information, refer to the *Introducing Image Mode for Red Hat Enterprise Linux* chapter in the *Using Image Mode for RHEL to Build, Deploy, and Manage Operating Systems* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_image\\_mode\\_for\\_rhel\\_to\\_build\\_deploy\\_and\\_manage\\_operating\\_systems/index#introducing-image-mode-for-rhel](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_image_mode_for_rhel_to_build_deploy_and_manage_operating_systems/index#introducing-image-mode-for-rhel)

For further information, refer to *Image Mode for Red Hat Enterprise Linux* at  
<https://developers.redhat.com/products/rhel-image-mode/overview#>

For further information, refer to *How Image Mode for RHEL Improves Security*  
<https://developers.redhat.com/articles/2025/02/25/how-image-mode-rhel-improves-security#>

## 18.2. Quiz

### Image Mode for Red Hat Enterprise Linux

---

Choose the correct answers to the following questions:

1. What is an advantage of using image mode over package mode when deploying Red Hat Enterprise Linux?  
 a. Image mode limits infrastructure drift.  
 b. Image mode requires no reboot after updates.  
 c. Image mode enables manual configuration changes without restrictions.  
 d. Image mode eliminates the need for a container registry.
  
2. Which tool builds and tags a container image for Red Hat Enterprise Linux in image mode?  
 a. Docker  
 b. OpenShift  
 c. Podman  
 d. Kubernetes
  
3. What is **not** a benefit of limiting infrastructure drift?  
 a. Software updates are provided in the form of container images.  
 b. Administrators can deploy local software packages as needed.  
 c. The root file system is read-only.  
 d. Rollbacks are supported when an upgrade goes wrong.
  
4. Which components must be included in a bootable container image for Red Hat Enterprise Linux?  
 a. Only the boot loader and application software.  
 b. Only application software and its dependencies.  
 c. The kernel and application software only.  
 d. The kernel, boot loader, and other operating system components that are typically excluded from application containers.

- For use by Wong Chia Cheong jason.wong76 jason.wong@trainocate.com Copyright © 2025 Red Hat, Inc. | For use by Wong Chia Cheong jason.wong76 jason.wong@trainocate.com Copyright © 2025 Red Hat, Inc.
- 5.** If the operating system is considered immutable when using image mode, how can administrators implement any changes?
- a.** The superuser can still make changes by using special local tools.
  - b.** Changes are possible only by updating the Containerfile and rebuilding the image that the server is associated with.
  - c.** The /etc and /var directories are designated to store local data.
  - d.** No changes are allowed in image mode.
- 6.** What function does the bootc command serve in image mode? Select all valid answers.
- a.** Checks for available updates.
  - b.** Downloads RPM packages.
  - c.** Installs image updates.
  - d.** Performs rollbacks.
- 7.** Which tool deploys Red Hat Enterprise Linux in image mode to a bare-metal target?
- a.** The bootc-image-builder tool
  - b.** Anaconda Kickstart
  - c.** The standard binary installation DVD
  - d.** A vmdk disk image that is built in image mode

## Solution

---

1. What is an advantage of using image mode over package mode when deploying Red Hat Enterprise Linux?  
 **a. Image mode limits infrastructure drift.**  
 **b. Image mode requires no reboot after updates.**  
 **c. Image mode enables manual configuration changes without restrictions.**  
 **d. Image mode eliminates the need for a container registry.**
2. Which tool builds and tags a container image for Red Hat Enterprise Linux in image mode?  
 **a. Docker**  
 **b. OpenShift**  
 **c. Podman**  
 **d. Kubernetes**
3. What is **not** a benefit of limiting infrastructure drift?  
 **a. Software updates are provided in the form of container images.**  
 **b. Administrators can deploy local software packages as needed.**  
 **c. The root file system is read-only.**  
 **d. Rollbacks are supported when an upgrade goes wrong.**
4. Which components must be included in a bootable container image for Red Hat Enterprise Linux?  
 **a. Only the boot loader and application software.**  
 **b. Only application software and its dependencies.**  
 **c. The kernel and application software only.**  
 **d. The kernel, boot loader, and other operating system components that are typically excluded from application containers.**

- For use by Wong Chia Cheong jason.wong76 jason.wong@trainocate.com Copyright © 2025 Red Hat, Inc. | For use by Wong Chia Cheong jason.wong76 jason.wong@trainocate.com
5. If the operating system is considered immutable when using image mode, how can administrators implement any changes?
- a. The superuser can still make changes by using special local tools.
  - b. Changes are possible only by updating the Containerfile and rebuilding the image that the server is associated with.
  - c. **The /etc and /var directories are designated to store local data.**
  - d. No changes are allowed in image mode.
6. What function does the bootc command serve in image mode? Select all valid answers.
- a. **Checks for available updates.**
  - b. Downloads RPM packages.
  - c. **Installs image updates.**
  - d. **Performs rollbacks.**
7. Which tool deploys Red Hat Enterprise Linux in image mode to a bare-metal target?
- a. The bootc-image-builder tool
  - b. **Anaconda Kickstart**
  - c. The standard binary installation DVD
  - d. A vmdk disk image that is built in image mode

## 18.3. Creating Installable Images for Image Mode

### Objectives

- Prepare a bootable container image that can be used by image mode for Red Hat Enterprise Linux.

### The Containerfile Format

When using image mode for Red Hat Enterprise Linux, you deploy software, associated dependencies, and configuration files to hosts by using container images that conform to the Open Container Initiative (OCI) standard. Container images deliver operating system updates that can also be rolled back if necessary.

You can use existing and familiar tools such as Podman to build container images. A key piece of the build process is the Containerfile, which is widely used to create application containers and has the benefit of a standardized format.

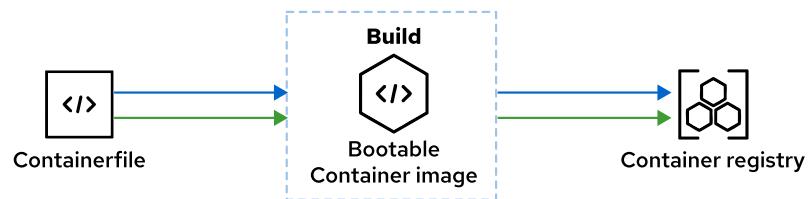


Figure 31. Building and publishing an image from a Containerfile

By adding a series of instructions to a Containerfile, you can customize an image to meet your needs. Each line in the Containerfile creates an image layer, which means that instructions are executed independently. During the build process, each instruction runs in an independent container by using an intermediate image built from the previous commands.

The following example Containerfile builds a typical application container:

```
# Use the ubi10 base image
FROM registry.redhat.io/ubi10 ①

# Install httpd
RUN dnf install -y httpd ②

# Copy a temporary index.html file to the web server root
COPY index.html /var/www/html/index.html ③

# Expose port 80 for httpd
EXPOSE 80 ④

# Set the entrypoint to run httpd in the foreground
ENTRYPOINT ["/usr/sbin/httpd", "-DFOREGROUND"] ⑤
```

- ① The Red Hat Universal Base Image is the foundation for this application container.
- ② The Apache HTTP service is layered on top of the base image.
- ③ The `index.html` file is copied from the project directory into the expected location in the container.
- ④ The `EXPOSE` instruction documents which port the service listens on at runtime. The HTTP port is TCP/80, which is defined in the standard Apache configuration file that ships in the `httpd` RPM package.
- ⑤ The `ENTRYPOINT` instruction defines the executable to run and any arguments to be passed to it when the container starts.

Although this sample Containerfile is sufficient for creating an application container, the approach to build a bootable container is somewhat different. Most importantly, the base image must be a specialized image that includes the required components for a full operating system, such as the kernel, a boot loader, and the `systemd` service manager. Red Hat provides the `rhel-bootc` base image, which incorporates tooling from the OSTree, composefs, and bootc upstream projects to enable image mode for RHEL.

### Important

Red Hat Enterprise Linux in image mode and the `rhel-bootc` image are subject to the Red Hat Enterprise Linux end user license agreement (EULA). Therefore, you **cannot** publicly redistribute the base or any derivative user-created images.

For example, the following Containerfile installs and enables the `httpd` package on top of the `rhel-bootc` image. Note the differences from the previous application container example.

```
# Use the rhel-bootc base image
FROM registry.redhat.io/rhel10/rhel-bootc:latest ①

# Install and enable httpd
RUN dnf -y install httpd mod_ssl && dnf clean all ②
RUN systemctl enable httpd && firewall-cmd --add-service=https ③

# Add httpd related files
ADD ./etc/ /etc ④
COPY ./index.html /var/www/html/index.html ⑤
```

- ① The FROM line directs Podman to use the rhel-bootc container image as a base.
- ② The RUN lines execute the command sequence within the container, creating new image layers in the process.
- ③ Because the image is for RHEL in image mode, this Containerfile enables the httpd service instead of configuring an ENTRYPOINT or CMD instruction. Additionally, this Containerfile opens the firewall for HTTPS instead of using an EXPOSE instruction.
- ④ The ADD line includes a file or directory at the destination within the container. The ADD command can resolve files over the network or unpack files within an archive in addition to local files.
- ⑤ The COPY command copies a file or directory at a relative path to the destination location. The COPY command does not unpack files within an archive, and instead copies archive files directly into the container.

#### Note

The Containerfile(5) manual page details the full specification for the Containerfile format. Many commands can take additional options and arguments, which you can use to significantly customize the resulting container image.

Some Containerfile commands have no effect for RHEL in image mode:

- The ENTRYPOINT and CMD instructions are replaced by starting services via the systemd service manager.
- The ENV instruction is replaced by configuring the systemd service.
- The EXPOSE instruction is only for documenting application ports. In image mode, you can control access to ports by using firewall configuration.
- The USER instruction is replaced by runtime user management.

Although these commands do not affect a RHEL system in image mode, you can still use commands such as the ENV command to facilitate intermediate steps in a Containerfile. For example, you might decide to define a list of packages by using the ENV command, and then install those packages later by using a RUN command.

## Building Bootable Container Images

You can use the podman build command to build a bootc image from the command line. Podman uses a Containerfile in the specified directory to build a container image. The --squash option merges all new layers into a single layer, which reduces the image's complexity and layer count.

```
user@host:~$ podman build --squash -t \
registry.example.com/user/webserver-bootc:latest .
...output omitted...
```

After building an image, you can push the image to an image repository by using the podman push command.

```
user@host:~$ podman push registry.example.com/user/webserver-bootc:latest
...output omitted...
```

## Testing a Bootable Image in Application Mode

To test a bootable container before deploying it, use the podman run command to launch the container locally. The following example starts the registry.example.com/user/webserver container image and forwards the 8080 port to port 80 in the container:

```
user@host:~$ podman run -d -p 8080:80 registry.example.com/user/webserver
9622...6a28
```

— You can then verify that the container functions as you expect. In this case, you can verify that the web server is accessible on port 8080 by using the curl command.

```
user@host:~$ curl http://localhost:8080
Hello image mode for RHEL!
```

You can also validate functionality by viewing the running processes in the container with the podman exec command.

```
user@host:~$ podman exec -l ps -ef
UID      PID ...      TIME CMD
root      1 ... 00:00:00 /sbin/init
root      34 ... 00:00:00 /usr/lib/systemd/systemd-journald
rpc       57 ... 00:00:00 /usr/bin/rpcbind -w -f
dbus      66 ... 00:00:00 /usr/bin/dbus-broker-launch --scope system --audit
root      69 ... 00:00:00 /usr/sbin/NetworkManager --no-daemon
root      75 ... 00:00:00 /usr/lib/systemd/systemd-logind
root      76 ... 00:00:00 /usr/libexec/udisks2/udisksd
root     133 ... 00:00:00 /usr/sbin/gssproxy -D
root     142 ... 00:00:00 /usr/sbin/httpd -DFOREGROUND
root     154 ... 00:00:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
apache   193 ... 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   194 ... 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   196 ... 00:00:00 /usr/sbin/httpd -DFOREGROUND
apache   207 ... 00:00:00 /usr/sbin/httpd -DFOREGROUND
root     378 ... 00:00:00 /bin/bash
```

Notice that PID 1 is the `init` process. The `init` process is the `systemd` service manager, which started the `httpd` service.

When running a `bootc`-based container in application mode, the included RHEL kernel does not start. Processes run in the context of the container platform, which is `podman` in this example.

## References

`podman(1)` and `Containerfile(5)` man pages

For further information, refer to the *Building and Testing RHEL bootc Images* chapter in the *Using Image Mode for RHEL to Build, Deploy, and Manage Operating Systems* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_image\\_mode\\_for\\_rhel\\_to\\_build\\_deploy\\_and\\_manage\\_operating\\_systems/index#building-and-testing-rhel-bootc-images](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_image_mode_for_rhel_to_build_deploy_and_manage_operating_systems/index#building-and-testing-rhel-bootc-images)

## 18.4. Guided Exercise

### Create Installable Images for Image Mode

Prepare a bootable container image that can be used by image mode for Red Hat Enterprise Linux.

#### Outcomes

- Build a bootable container image and push it to a container registry.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start image-bootable
```

#### Instructions

1. Log in to the `registry.lab.redhat.com:5000` container registry. List the registry's contents.

- 1.1. Log in to the `registry.lab.redhat.com:5000` registry with the `podman login` command. Use the student user and redhat as the password.

```
student@workstation:~$ podman login registry.lab.example.com:5000
Username: student
Password: redhat
Login Succeeded!
```

- 1.2. Use the `podman search` command to list the contents of the registry.

```
student@workstation:~$ podman search registry.lab.example.com:5000/
NAME                                     DESCRIPTION
registry.lab.example.com:5000/rhel10/httpd-24
registry.lab.example.com:5000/rhel10/rhel-bootc
registry.lab.example.com:5000/ubi10/ubi
registry.lab.example.com:5000/ubi8/ubi
```

2. Change to the `~/RH304/labs/image-bootable` directory. In this exercise, the `/etc` directory contains configuration for the services in the `webserver-bootc` container. You can inspect the files in the `/etc` directory to see examples of configurations that you might include in other environments.

```
student@workstation:$ cd ~/RH304/labs/image-bootable
```

- 3.** Modify the `~/RH304/labs/image-bootable/Containerfile` file with the following changes to start the `httpd` service.

- 3.1.** Use the `registry.lab.example.com:5000/rhel10/rhel-bootc` image as the base image from the registry.

```
# parent image
FROM registry.lab.example.com:5000/rhel10/rhel-bootc
```

- 3.2.** Add the local `/etc` directory to the container. The `httpd` configuration uses the `/etc` directory because the `/var` directory is not included in updates to the `bootc` layers.

```
# add local etc directory to container
ADD etc /etc
```

- 3.3.** Install and start the `httpd` service.

```
# install and enable httpd
RUN dnf install -y httpd && systemctl enable httpd
```

- 3.4.** Copy the `index.html` file to the `/var/www/html/` directory.

```
# copy local index.html file to /var/www/html/
COPY index.html /var/www/html/
```

- 3.5.** Your updated Containerfile must match the following content:

```
# parent image
FROM registry.lab.example.com:5000/rhel10/rhel-bootc

# add local etc directory to container
ADD etc /etc

# install and enable httpd
RUN dnf install -y httpd && systemctl enable httpd

# copy local index.html file to /var/www/html/
COPY index.html /var/www/html/
```

- 3.6.** Save and close the `Containerfile`.

**4.** Build the container image locally. Use

registry.lab.example.com:5000/student/webserver-bootc as the container image tag.

```
student@workstation:~/RH304/labs/image-bootable$ podman build --squash -t \
registry.lab.example.com:5000/student/webserver-bootc .
STEP 1/4: FROM registry.lab.example.com:5000/rhel10/rhel-bootc
Trying to pull registry.lab.example.com:5000/rhel10/rhel-bootc:latest...
Getting image source signatures
Copying blob e2ac657e37ce done    |
...output omitted...
Writing manifest to image destination
STEP 2/4: ADD etc /etc
STEP 3/4: RUN dnf install -y httpd && systemctl enable httpd
...output omitted...
STEP 4/4: COPY index.html /var/www/html/
COMMIT registry.lab.example.com:5000/student/webserver-bootc
...output omitted...
Copying blob 8fc931a47a47 done    |
Copying config 68c1bd7335 done    |
Writing manifest to image destination
--> 68c1bd73352a
Successfully tagged registry.lab.example.com:5000/student/webserver-
bootc:latest
68c1...db94
```

**5.** Test the image in a local container.

- 5.1.** Start the local container by using the podman run -d command. Add port forwarding for port 8080 on the workstation machine to port 80 in the container with the -p 8080:80 option.

```
student@workstation:~/RH304/labs/image-bootable$ podman run -p 8080:80 -d \
registry.lab.example.com:5000/student/webserver-bootc
852b...f40d
```

- 5.2.** Change to the /home/student directory.

```
student@workstation:~/RH304/labs/image-bootable$ cd /home/student
```

- 5.3.** Use the curl command to verify that you can access the web server.

```
student@workstation:$ curl localhost:8080
Hello image mode for RHEL!
```

- Push the container image to the container registry by using the podman push command.

```
student@workstation:~$ podman push \
registry.lab.example.com:5000/student/webserver-bootc
...output omitted...
Writing manifest to image destination
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish image-bootable
```

## 18.5. Installing Red Hat Enterprise Linux by Using Image Mode

### Objectives

- Install Red Hat Enterprise Linux on a new server by using image mode.

### Deploying a Bootable Container

A bootable container image includes the kernel, the boot loader, and other operating system components that are otherwise excluded from application containers. Although you can launch a container instance from the bootc image to test an application that is built into the image, ultimately a bootable container image must be deployed to a host that boots from it.

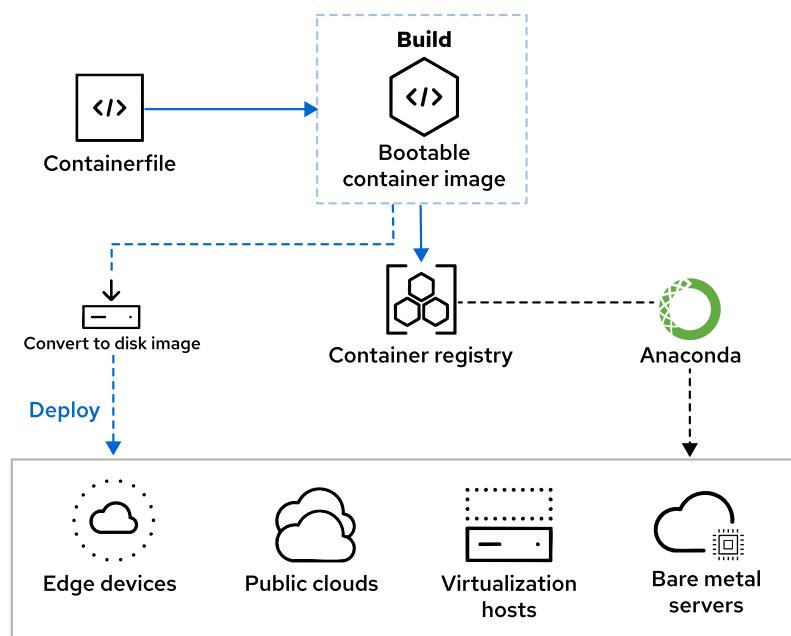


Figure 32. Using a bootable container to provision a host

When the contents of the image are installed on a physical or a virtual machine and the system boots up, the user space deployment itself does not run as a container. At this point, the container runtime that was used to build the image is no longer relevant. When the bootc tool is operating in deployed mode, the kernel that was installed from the image is executed during boot, and systemd runs as Process ID 1 to control all services, as is customary in Red Hat Enterprise Linux. The bootc tool is not a persistent daemon and does not impose any runtime overhead.

### Bare-metal Deployment by Using Anaconda Kickstart

To provision bare-metal servers, the recommended approach is to use the Anaconda installer's Kickstart feature, which supports image mode.

A Kickstart file helps to automate installation by providing answers to all the configuration options that the interactive installer would normally prompt the user for, such as storage configuration, user management, and package selection.

In general, all the standard commands and sections of a typical Kickstart file can be retained and used in image mode. The one notable exception is how software to be installed is specified. Traditionally, Anaconda is used in package mode and the %packages section in a Kickstart file is required to define software packages that you want to install:

```
%packages  
@^minimal-environment  
vim-enhanced  
%end
```

To use Kickstart in image mode, you must remove the %packages section and replace it with the `ostreecontainer` command, to indicate the registry and the bootable container to deploy on the target system. The container image must be derived from the official `rhel-bootc` base image.

```
ostreecontainer --url=registry.lab.example.com:5000/student/webserver-bootc
```

This command is a reference to the `libostree` project, which `bootc` currently depends on for storing the base container image. The use of OSTree is a design choice, and its role might decrease in the future.

#### Note

The direct use of the `rpm-ostree` command to make system changes or to install content is not supported.

The RHEL `bootc` image is generic and does not include any configuration such as a default user account. To create user accounts and set passwords or to inject SSH public keys, you can use standard Kickstart commands. The following example sets the root password and creates a user account with administrator privileges:

```
rootpw --iscrypted $6$KUnFfrTz08jv.PiH$YlBb0tXBkWzoMuRfb0.SpbQ....XDR1UuchoMG1  
user --name=student --groups=wheel --iscrypted --password=$6$JJlPUjtsQ....F2co
```

After you successfully adjust your Kickstart file for use in image mode, you can publish the file to make it available in your environment. The most popular choice is to copy the file to a web server that can be accessed on the local network.

To start the Anaconda installer on a target system, you can choose from several methods. You might set up PXE services for network booting as a more advanced solution, or copy the RHEL binary installation ISO to a USB drive to boot the new machine manually. In either case, when the Anaconda boot menu starts on the target system, the `inst.ks=URL` argument must be provided on the kernel command line to inform the installer where the Kickstart file is located.

If the Kickstart file is not referenced correctly, then Anaconda starts the normal graphical user interface, which does not support image mode currently.

#### Note

For more information about how to use Kickstart, refer to [Automatically Installing Red Hat Enterprise Linux](#).

## Generating Disk Images for Hybrid Cloud Environments

Portability is an important factor when using Red Hat Enterprise Linux in image mode. In addition to bare-metal systems, the `bootc` images can also be deployed to machines in virtualized, cloud, and edge environments. This process involves converting a `bootc` image to one of the native disk formats.

The `bootc-image-builder` tool is available to generate disk images from the `bootc` image for various platforms:

Image format	Target environment
qcow2 (default)	QEMU
vmdk	VMware vSphere
vhd	Microsoft Virtual PC

Image format	Target environment
ami	Amazon Machine Image
gcd	Google Compute Engine
raw	Unformatted raw disk

To get started with the `bootc-image-builder` tool, you must first download its container image from the Red Hat Ecosystem Catalog. This tool is intended for use as a container and it is not available as an RPM package in RHEL.

The following steps show how to use the `podman` command to get the necessary container image. The `sudo` command is used for privilege escalation because the `bootc-image-builder` tool must run as a root container when it is launched to generate disk images.

1. Authenticate to the registry:

```
user@host:~$ sudo podman login registry.redhat.io
Username: provide_username
Password: provide_password
Login Succeeded!
```

2. Download the `bootc-image-builder` container image:

```
user@host:~$ sudo podman pull registry.redhat.io/rhel10/bootc-image-builder
...output omitted...
8717...d405
```

## Creating a QCOW2 Disk Image

The default disk image format that the `bootc-image-builder` tool generates is QCOW2. This disk format can be used to deploy virtual machines that run on Kernel-based Virtual Machine (KVM) hypervisors.

Red Hat Enterprise Linux provides virtualization functions by using KVM. Virtualization capabilities are further expanded in the OpenShift Virtualization and Red Hat OpenStack Platform products.

The following example shows how to generate a QCOW2 disk image from a custom bootable container image that is derived from the official `rhel-bootc` base image.

The RHEL bootc base image is generic and contains no user accounts. The recommended approach to create users is to supply a customization file to the `bootc-image-builder` tool with the necessary user information. Create the `config.toml` file in your project directory to define users with content similar to the following example:

```
[[customizations.user]]
name = "user1"
password = "password1"
groups = ["wheel"]

[[customizations.user]]
name = "user2"
password = "password2"
groups = ["wheel"]
```

This example build configuration file is in the TOML data format; JSON is also supported.

During image build, the file must be mapped into the container by using the `/config.toml` path.

As another prerequisite, create a directory inside the project that binds to the output directory inside the container, where the `bootc-image-builder` tool writes the qcows2 disk image.

```
user@host:~$ mkdir output
```

To start the build process, run the `bootc-image-builder` container by using the following arguments:

```
user@host:~$ sudo podman run \
--rm \
-it \
--privileged \ ①
--security-opt label=type:unconfined_t \
-v ./config.toml:/config.toml \ ②
-v ./output:/output \ ③
registry.redhat.io/rhel10/bootc-image-builder \
--type qcow2 \ ④
registry.lab.example.com/user/bootc-httd:latest ⑤
...output omitted...
```

- ① This container must run with root privileges.
- ② The customization file to create user accounts is mounted into the bootc-image-builder container by using the expected /config.toml path.
- ③ The output directory in the current working directory is mounted into the container by using the expected /output path.
- ④ Specify the disk image type.
- ⑤ Refer to the customized bootable container image to convert to a disk image.

## Deploying a New Virtual Machine on KVM

To test the new disk image, copy it to the file-system location where RHEL hypervisors expect to store images by default:

```
user@host:~$ sudo cp output/qcow2/disk.qcow2 /var/lib/libvirt/images/
```

Then, create a virtual machine by using the `virt-install` command. The following example defines a minimal set of settings to create the machine and to import the disk that the bootc-image-builder container generated.

```
user@host:~$ sudo virt-install \
--name bootc-webserver \
--memory 4096 \
--import --disk /var/lib/libvirt/images/disk.qcow2 \
--graphics none \
--osinfo rhel10.0 \
--noautoconsole \
--noreboot
...output omitted...
```

After successfully defining the new virtual machine, you can start it by using the `virsh` command.

```
user@host:~$ sudo virsh start bootc-webserver  
...output omitted...
```

## References

`virsh(1)`, and `virt-install(1)` man pages

For more information about image mode, refer to the *Using Image Mode for RHEL to Build, Deploy, and Manage Operating Systems* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_image\\_mode\\_for\\_rhel\\_to\\_build\\_deploy\\_and\\_manage\\_operating\\_systems/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_image_mode_for_rhel_to_build_deploy_and_manage_operating_systems/index)

For more information about how to use Kickstart, refer to the *Automatically Installing Red Hat Enterprise Linux* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/automatically\\_installing\\_rhel/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/automatically_installing_rhel/index)

For more information about virtualization, refer to the *Configuring and Managing Linux Virtual Machines* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/configuring\\_and\\_managing\\_windows\\_virtual\\_machines/index](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/configuring_and_managing_windows_virtual_machines/index)

## 18.6. Guided Exercise

### Install Red Hat Enterprise Linux by Using Image Mode

Install Red Hat Enterprise Linux on a new server by using image mode.

#### Outcomes

- Adjust a Kickstart file to use image mode.
- Install Red Hat Enterprise Linux 10 in image mode by using Kickstart.
- Explore the differences between image mode and package mode.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

##### Important

You must complete the [Guided Exercise Create Installable Images for Image Mode](#) before starting this section. In the previous activity, you created a bootable container image and uploaded it to the local registry. In this activity, you use this container image to install RHEL 10 in image mode.

```
student@workstation:~$ lab start image-server
```

#### Instructions

1. Modify the `~/RH304/labs/image-server/ks.cfg` Kickstart file to use image mode instead of the traditional package mode for installation.

- 1.1. On the workstation machine, log in to the `servera` machine as the `student` user.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 1.2. Modify the `~/RH304/labs/image-server/ks.cfg` Kickstart file. Locate the `packages` section and delete it, because you are not using traditional package mode during this installation.

```
%packages  
@^minimal-environment  
vim-enhanced  
%end
```

- 1.3.** To set up image mode, add the `ostreecontainer` command and reference the registry URL to access the container image that you built in a previous exercise.

```
ostreecontainer --url=registry.lab.example.com:5000/student/webserver-bootc
```

- 1.4.** Your updated Kickstart file must match the following content:

```
# Use text display mode
text

# Storage configuration
clearpart --all --initlabel --disklabel=gpt
reqpart --add-boot
part / --grow --fstype xfs

# Network configuration
network --bootproto=dhcp --device=link --activate
firewall --disabled

# User management
user --name="student" --groups="wheel" --plaintext --password="student"
rootpw --plaintext redhat

ostreecontainer --url=registry.lab.example.com:5000/student/webserver-bootc

%pre
cat <<EOF > /etc/containers/registries.conf.d/registry-lab.conf
[[registry]]
location = "registry.lab.example.com"
insecure = true
blocked = false
EOF
skopeo login -u student -p redhat registry.lab.example.com:5000 --
authfile=/run/ostree/auth.json
%end

# Disable the kernel crash dumping mechanism
%addon com_redhat_kdump --disable
%end
```

**1.5.** Without making further changes, observe the preinstallation script in the Kickstart file. This script is necessary because the local registry in the classroom environment uses a self-signed certificate and requires authentication. The script only applies changes to the Anaconda environment and does not affect the target machine.

**1.6.** Save and close the Kickstart file.

**2.** Publish the Kickstart file to the Apache HTTP server running on the `servera` machine. This Kickstart file must be present in the `/var/www/html` directory. Use `student` as the password.

**2.1.** Use the Apache HTTP server that is running on the `servera` machine to publish the Kickstart file. Copy the completed Kickstart file to the `/var/www/html` directory. Use `student` as the

password.

```
[student@servera ~]$ sudo cp ~/RH304/labs/image-server/ks.cfg \
/var/www/html/
[sudo] password for student: student
```

**2.2.** Return to the workstation machine as the student user.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

**3.** Access the serverc console and reboot the system to initiate PXE network booting and to install RHEL 10 in image mode by using the Kickstart file.

**3.1.** Locate the serverc console icon in your classroom environment. Open the console.

**3.2.** To reboot, send **Ctrl** + **Alt** + **Del** to your system by using the relevant keyboard, virtual, or menu entry.

**3.3.** When the PXE boot-loader menu appears, select the **Install RHEL 10** menu entry and press **Tab** to edit the menu item.

**3.4.** Add the **inst.ks=http://servera.lab.example.com/ks.cfg** boot option at the end of the line.

**3.5.** Press **Enter** to start the installer.

**3.6.** The installation begins automatically without prompting for any other information. As the installation proceeds, you can switch between the tmux windows by pressing **Ctrl** + **B** followed by the number of the window to access.

**3.7.** After the installation finishes, press **Enter** to reboot the serverc machine.

**4.** Boot from local disk and log in to the serverc machine.

**4.1.** The PXE boot-loader menu appears again because this machine is intentionally configured to always boot from the network. Select the **Boot from local disk** menu entry to start the newly installed operating system.

**4.2.** Log in to the serverc machine as the student user. Use student as the password.

```
serverc login: student
Password: student
[student@serverc ~]$
```

5. Explore the new system that was deployed in image mode. Note important differences in configuration compared to traditional package mode.

**5.1.** List the contents of the root file system.

Notice that the /home, /root, /srv, and /mnt directories are stored under the /var directory, which is considered mutable machine local state.

```
[student@serverc ~]$ ls -l /
total 68
lrwxrwxrwx.  1 root root    7 Dec 31 1969 bin -> usr/bin
drwxr-xr-x.  5 root root  101 Jun 23 15:02 boot
drwxr-xr-x.  2 root root   27 Dec 31 1969 cachi2
drwxr-xr-x. 19 root root 3260 Jun 23 2025 dev
drwxr-xr-x. 91 root root 8192 Jun 23 2025 etc
lrwxrwxrwx.  1 root root    8 Dec 31 1969 home -> var/home
lrwxrwxrwx.  1 root root    7 Dec 31 1969 lib -> usr/lib
lrwxrwxrwx.  1 root root    9 Dec 31 1969 lib64 -> usr/lib64
lrwxrwxrwx.  1 root root    9 Dec 31 1969 media -> run/media
lrwxrwxrwx.  1 root root    7 Dec 31 1969 mnt -> var/mnt
drwxr-xr-x.  2 root root   27 Dec 31 1969 opt
lrwxrwxrwx.  1 root root   14 Dec 31 1969 ostree -> sysroot/ostree
dr-xr-xr-x. 197 root root    0 Jun 23 15:02 proc
lrwxrwxrwx.  1 root root   12 Dec 31 1969 root -> var/roothome
drwxr-xr-x.  41 root root 1120 Jun 23 2025 run
lrwxrwxrwx.  1 root root    8 Dec 31 1969 sbin -> usr/sbin
lrwxrwxrwx.  1 root root    7 Dec 31 1969 srv -> var/srv
dr-xr-xr-x.  13 root root    0 Jun 23 2025 sys
drwxr-xr-x.  12 root root   123 Jun 23 14:59 sysroot
drwxrwxrwt. 12 root root   240 Jun 23 15:03 tmp
drwxr-xr-x.  13 root root   224 Dec 31 1969 usr
drwxr-xr-x.  25 root root 4096 Jun 23 2025 var
```

**5.2.** Verify that the /var directory is mutable by creating the testfile file in the /var/tmp directory.

```
[student@serverc ~]$ echo testing > /var/tmp/testfile
```

**5.3.** To confirm that the /etc directory is also mutable and persistent, update the motd file with the This is image mode. text. Use student as the password.

```
[student@serverc ~]$ sudo bash -c 'echo "This is image mode." > /etc/motd'  
...output omitted...  
[sudo] password for student: student
```

- 5.4.** An important difference from package mode is that the root file system is immutable. Try to create a test file in the / directory to confirm that the file system is read-only.

```
[student@serverc ~]$ sudo touch /test.root  
touch: cannot touch '/test.root': Read-only file system
```

- 5.5.** The /usr directory where programs are stored is also immutable. Try to create test file in the /usr directory to confirm that the file system is read-only.

By default, you cannot install packages by using the dnf command. New software must be deployed by updating the container image that the system is associated with.

```
[student@serverc ~]$ sudo touch /usr/test.usr  
touch: cannot touch '/usr/test.usr': Read-only file system
```

- 5.6.** Using the bootc status command, verify the container image that is currently in use.

```
[student@serverc ~]$ sudo bootc status  
• Booted image: registry.lab.example.com:5000/student/webserver-bootc  
    Digest: sha256:1270...15ed  
    Version: 10.0 (2025-06-23 18:49:52.323586597 UTC)
```

- 6.** Close the serverc console.
- 7.** Use the curl command on the workstation machine to verify the website content that you added to the container image in the previous exercise.

```
student@workstation:~$ curl http://serverc.lab.example.com  
Hello image mode for RHEL!
```

## Finish

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish image-server
```

## 18.7. Managing Image Mode-based Systems

### Objectives

- Manage, inspect, modify, and update a server that uses image mode for Red Hat Enterprise Linux.

### Image Mode Day 2 Operations

Day 2 operations refers to the ongoing management tasks on a system after it is deployed. These tasks cover the bulk of the work to do on a system during its lifetime.

After you build and deploy the bootc image to a target system, you can perform Day 2 operations by fetching updates and applying them. These operations consist of transactional updates from a container registry; they are layered, are installed in-place, and can be rolled back if necessary.

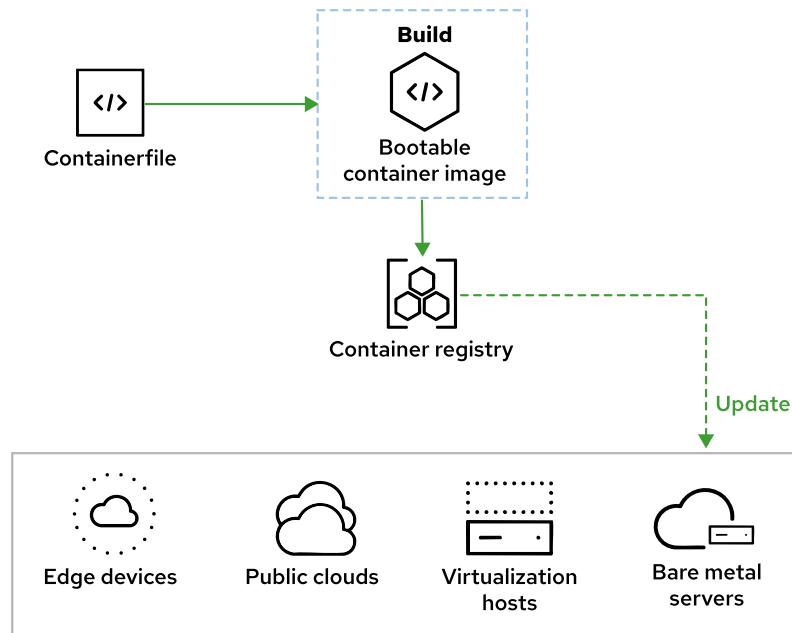


Figure 33. Managing an image mode-based operating system

### Verifying the Current System State

You can check the current status of the system by using the `bootc status` command. It displays whether a new image is staged for update, the current booted image in the system, and the rollback image that is set, if any. A newly installed system does not show any staged or rollback images.

```
root@host:~# bootc status
apiVersion: org.containers.bootc/v1
kind: BootcHost
metadata:
  name: host
spec:
  image:
    image: registry.lab.example.com/user/example-bootc
    transport: registry
  bootOrder: default
status:
  staged: null
  booted:
    image:
      image:
        image: registry.lab.example.com/user/example-bootc
        transport: registry
      version: 10.20250116.0
      timestamp: null
      imageDigest: sha256:87b02be441cda3d66c9c...71b092ce7b2311e2defbb93f
    cachedUpdate: null
    incompatible: false
    pinned: false
    store: ostreeContainer
    ostree:
      checksum: 9981913741c37dbb6400...3b68479d2c33b6f5434aa338
      deploySerial: 0
  rollback: null
  rollbackQueued: false
  type: bootcHost
```

## Upgrading the System

Red Hat Enterprise Linux needs to be installed in image mode only once on a system. After the system is installed, you must make all subsequent changes and updates by performing the following steps:

1. Edit the Containerfile in the bootc image project directory.
2. Rebuild the container image.
3. Push the new container image to the registry with the same name tag.
4. Update the image on the system by using the bootc command.
5. Reboot the system to apply the changes.

When using image mode for RHEL, automatic updates are enabled by default. The bootc-fetch-apply-updates systemd service and the timer unit files check the container registry for updates

regularly and apply them to the system.

```
root@host:~# systemctl status bootc-fetch-apply-updates.timer
● bootc-fetch-apply-updates.timer - Apply bootc updates
  Loaded: loaded (/usr/lib/systemd/system/bootc-fetch-apply-updates.timer);
  disabled; preset: disabled
  Active: active (waiting) since Wed 2025-04-09 13:22:18 EDT; 5min ago
    ...output omitted...
Apr 09 13:22:18 localhost systemd[1]: Started bootc-fetch-apply-updates.timer - Apply bootc updates.
```

However, you can choose to update your systems manually. You might update manually when applying nontrivial updates. Turning off the bootc automatic updates might also be necessary if you use automated processes for updating, for example by using Ansible.

To turn off automatic updates permanently, you can disable the timer unit that checks for image updates.

```
root@host:~# systemctl mask bootc-fetch-apply-updates.timer
Created symlink '/etc/systemd/system/bootc-fetch-apply-updates.timer' →
'/dev/null'.
```

To manually fetch updates from a registry and to boot into the new deployment, use the bootc upgrade command. This command fetches the transactional in-place updates from the container image registry to the installed operating system. The command queries the registry, downloads the updated image layers, and configures the new deployment for use during the next boot. When the updated base image is staged this way, the running system remains unaffected. A reboot is always required to apply the changes.

```
root@host:~# bootc upgrade
layers already present: 69; layers needed: 2 (34.5 MB)
Fetching layers [██████████] 2/2
  L Fetching [██████████] 242 B/242 B (0 B/s) layer
sha256:e0f434edc580ca
Fetched layers: 32.95 MiB in 36 seconds (945.46 KiB/s)
Pruned images: 0 (layers: 0, objsize: 40.0 MB)
Queued for next boot: registry.lab.example.com/user/example-bootc
  Version: 10.20250116.0
  Digest: sha256:dc6a...27bc
Total new layers: 71    Size: 739.4 MB
Removed layers:   2    Size: 26.9 MB
Added layers:    2    Size: 34.5 MB
```

You can use the `--apply` option to automatically reboot the system when a staged image is present. Use the `--check` option to see image updates without applying them.

#### Note

The `bootc upgrade` and `bootc update` commands are aliases.

## File-system Analysis in Image Mode

In image mode systems, the file-system layout is similar to a traditional installation in that directories such as `/home`, `/etc`, `/var`, and `/usr` still store the types of data that you would expect. However, these locations are used differently in image mode. They use a combination of immutable and mutable volumes to keep consistency, avoid drift, and still enable data persistence.

## The Root File System

Image mode for RHEL uses `composefs` for the root file system by default. `Composefs` is an overlay file system that enables a truly read-only root file system. The data comes from one of the OSTree deployments in the `/ostree/repo` directory. OSTree can store multiple file systems in parallel in its repository, which enables deploying a new file-system image without deleting the earlier version. This model is similar to source code version control, but for entire file-system objects instead of individual files. Combining `Composefs` with an OSTree-based system improves both performance and file-system integrity.

```
root@host:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        9.0G  1.9G  7.2G  21% /sysroot
composefs     8.4M  8.4M   0  100% /
devtmpfs        4.0M   0    4.0M   0% /dev
tmpfs           983M   0   983M   0% /dev/shm
tmpfs           394M  488K  393M   1% /run
tmpfs           983M   0   983M   0% /tmp
tmpfs           1.0M   0   1.0M   0% /run/credentials/systemd-journald.service
/dev/sda2        960M  153M  808M  16% /boot
tmpfs           1.0M   0   1.0M   0% /run/credentials/getty@tty1.service
tmpfs           197M  4.0K  197M   1% /run/user/1000
```

## The /etc and /var Directories

The /etc and /var system directories are mutable and persistent by default, and they work differently.

For the /etc directory, the bootc image performs a 3-way merge during upgrades.

Each file-system tree that you deploy from the bootc image has its own copy of the /etc directory. Your local changes in the /etc directory of the currently booted deployment are retained and propagated across all future upgrades. On the other hand, unmodified files in the /etc directory get updated if a new bootc image provides later versions of the same files.

The ostree-finalize-staged.service unit executes these tasks during shutdown, and creates a boot-loader entry for the new deployment.

The /var directory is shared between the various file-system deployments that are installed on the system. This directory is not merged during upgrades.

By default, the contents of the /var directory are copied from the container image during the initial installation but are not updated afterwards. During later upgrades, existing files are not changed or deleted, even if the Containerfile includes explicit instructions to do so. When you roll back by using the bootc command, the contents of the /var directory are not rolled back.

### Note

You cannot push changes to content in the /var directory by using image upgrades or rollbacks.

## Version Selection and Bootup

Image mode for RHEL uses GRUB2 (GRand Unified Bootloader version 2) by default except for the s390x architectures. Each OSTree deployment that is currently available on a system has its own menu entry.

The GRUB2 boot loader uses these entries to present an interactive menu during system bootup. Within menu entries, kernel arguments determine which OSTree deployment to use for the root file system. This mechanism enables booting into previous deployments if a particular upgrade caused problems.



Figure 34. Choosing a deployment version during boot

## Rolling Back a System

You can use the `bootc rollback` command to revert the system to a previous available image version. When the command is executed, the deployment under the `rollback` option is queued for the next boot, and the current version becomes the new rollback. The GRUB2 menu gets updated to show the previous deployment as the first entry. Finally, if a staged entry exists, it is discarded.

```
root@host:~# bootc rollback
bootfs is sufficient for calculated new size: 0 bytes
Next boot: rollback deployment
```

## References

For more information about managing systems in image mode, refer to the *Managing RHEL bootc Images* chapter in the *Using Image Mode for RHEL to Build, Deploy, and Manage Operating Systems* guide at [https://docs.redhat.com/en/documentation/red\\_hat\\_enterprise\\_linux/10/html-single/using\\_image\\_mode\\_for\\_rhel\\_to\\_build\\_deploy\\_and\\_manage\\_operating\\_systems/index#managing-rhel-bootc-images](https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html-single/using_image_mode_for_rhel_to_build_deploy_and_manage_operating_systems/index#managing-rhel-bootc-images)

## 18.8. Guided Exercise

### Manage Image Mode-based Systems

Manage, inspect, modify, and update a server that uses image mode for Red Hat Enterprise Linux.

#### Outcomes

- Update an image-mode based server to a new image to add new tools and features.
- Revert the server to a previous image to undo the changes.

#### Prerequisites

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

##### Important

You must complete the [Guided Exercise Install Red Hat Enterprise Linux by Using Image Mode](#) before starting this section. In the previous activity, you installed RHEL in image mode on the `serverc` machine. In this activity, you manage this system by making configuration changes in image mode.

```
student@workstation:~$ lab start image-manage
```

#### Instructions

1. In a previous exercise, you deployed a web server on the `serverc` machine. Verify that this web server is running, and add the Local content edited on `serverc.content` to the end of the `index.html` file.

- 1.1. On the workstation machine, use the `curl` command to confirm that the web server on the `serverc` machine from the previous activity is running.

```
student@workstation:~$ curl http://serverc.lab.example.com/index.html
Hello image mode for RHEL!
```

- 1.2. Log in to the `serverc` machine as the student user and switch to the root user. Use `student` as the password.

```
student@workstation:~$ ssh serverc
student@serverc's password: student
[student@serverc ~]$ sudo -i
[sudo] password for student: student
[root@serverc ~]#
```

At this point, if you receive an error message about the `bootc-fetch-apply-updates.service` failing, you can safely ignore it. The error is due to the registry's authentication requirement, which is addressed in a later step.

- 1.3.** Modify the `index.html` file on the `serverc` machine. Add the Local content edited on `serverc`. line to the end of the `/var/www/html/index.html` file.

```
[root@serverc ~]# echo "Local content edited on serverc." \
>> /var/www/html/index.html
```

- 2.** Open a new terminal window on the workstation machine, and verify the updated content of the web server running on the `serverc` machine.

The contents of the `/var` directory can be changed directly on the server that is running in image mode, because this directory is considered to be local machine state.

```
student@workstation:~$ curl http://serverc.lab.example.com/index.html
Hello image mode for RHEL!
Local content edited on serverc.
```

- 3.** On the `serverc` machine, try to install the `mod_ssl` package that enables the HTTPS web server, and the `vim_enhanced` package to provide the `vim` editor.

- 3.1.** On the `serverc` machine, try to install the `mod_ssl` and `vim_enhanced` packages by using the DNF package manager. The `dnf` command recognizes that it is running on a `bootc` system, and stops the operation by default.

```
[root@serverc ~]# dnf install mod_ssl vim-enhanced
...output omitted...
Dependencies resolved.
=====
 Package      Architecture Version       Repository      Size
=====
 Installing:
  mod_ssl      x86_64        1:2.4.63-1.el10  rhel-10.0-...-appstream-rpms 113 k
 Installing dependencies:
  sscg         x86_64        3.0.5-9.el10   rhel-10.0-...-appstream-rpms 49 k

Transaction Summary
=====
Install 2 Packages

Total download size: 162 k
Installed size: 366 k
This bootc system is configured to be read-only. Pass --transient to perform
this transaction in a transient overlay which will reset when the system
reboots.
Operation aborted.
```

- 3.2.** Check the available space on the server to confirm the root cause for the issue. Make note of the / file system.

The / file system is a read-only overlay, so you must take a different approach.

```
[root@serverc ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        9.0G  1.6G  7.5G  17% /sysroot
composefs        7.9M  7.9M    0 100% /
devtmpfs        4.0M    0  4.0M   0% /dev
tmpfs           983M    0  983M   0% /dev/shm
tmpfs           394M  496K  393M   1% /run
tmpfs           1.0M    0  1.0M   0% /run/credentials/systemd-journald.service
tmpfs           983M    0  983M   0% /tmp
/dev/sda2        960M  153M  808M  16% /boot
tmpfs           1.0M    0  1.0M   0% /run/credentials/getty@tty1.service
tmpfs           197M  4.0K  197M   1% /run/user/1000
```

- 4.** On the workstation machine, build an updated container image that includes the mod\_ssl and vim-enhanced packages. Also add the Hello from a modified image-mode based installation! text to the /var/www/html/index.html file in the container image. Push the new version of the container image to the registry with the registry.lab.example.com:5000/student/webserver-bootc image tag.

**4.1.** On the workstation machine, edit the `~/RH304/labs/image-manage/Containerfile` file and add the following lines to the end of the file.

**4.2.** Install the `mod_ssl` and `vim-enhanced` packages.

These commands could have been added to the existing RUN instruction to reduce the number of image layers and to improve efficiency in terms of image size. However, this Containerfile uses an additional RUN command for clarity.

```
# install additional packages
RUN dnf -y install mod_ssl vim-enhanced && \
dnf clean all
```

**4.3.** Add the Hello from a modified image-mode based installation! text to the `/var/www/html/index.html` file.

```
# Create an index.html file with different content
RUN <<EOF
echo "Hello from a modified image-mode based installation!" \
> /var/www/html/index.html
EOF
```

**4.4.** Your updated file must match the following content:

```
# parent image
FROM registry.lab.example.com:5000/rhel10/rhel-bootc

# add local etc directory to container
ADD etc /etc

# install and enable httpd
RUN dnf install -y httpd && systemctl enable httpd

# copy local index.html file to /var/www/html/
COPY index.html /var/www/html/

# install additional packages
RUN dnf -y install mod_ssl vim-enhanced && \
    dnf clean all

# Create an index.html file with different content
RUN <<EOF
    echo "Hello from a modified image-mode based installation!" \
        > /var/www/html/index.html
EOF
```

**4.5.** Save and close the Containerfile.

**4.6.** Build a new version of the bootable container image by using the updated Containerfile. Use the `registry.lab.example.com:5000/student/webserver-bootc` image tag.

The alphanumeric identifiers might differ on your system.

```
student@workstation:~$ podman build --squash -t \
registry.lab.example.com:5000/student/webserver-bootc RH304/labs/image-manage/
STEP 1/6: FROM registry.lab.example.com:5000/rhel10/rhel-bootc
Trying to pull registry.lab.example.com:5000/rhel10/rhel-bootc:latest...
Getting image source signatures
Copying blob 4d3d79f44f05 done    |
...output omitted...
STEP 2/6: ADD etc /etc
STEP 3/6: RUN dnf install -y httpd && systemctl enable httpd
...output omitted...
STEP 4/6: COPY index.html /var/www/html/
STEP 5/6: RUN dnf -y install mod_ssl vim-enhanced && dnf clean all
...output omitted...
STEP 6/6: RUN <<EOF (echo "Hello from a modified image-mode based installation!"
\...)
COMMIT registry.lab.example.com:5000/student/webserver-bootc
Getting image source signatures
Copying blob 31f9a8e96430 skipped: already exists
...output omitted...
Writing manifest to image destination
--> c3e98f94beba
Successfully tagged registry.lab.example.com:5000/student/webserver-bootc:latest
c3e9...7b16
```

5. On the workstation machine, push the created image to the `registry.lab.example.com:5000` registry.

#### 5.1. Log in to the registry with the student username and use redhat as the password.



##### Note

If you are already logged in to the registry, then you do not need to authenticate again.  
You can proceed with the next steps.

```
student@workstation:~$ podman login registry.lab.example.com:5000
Username: student
Password: redhat
Login Succeeded!
```

- 5.2. Push the `registry.lab.example.com:5000/student/webserver-bootc` local image to the registry.

The alphanumeric identifiers might differ on your system.

Most of the image layers are skipped during the push. Only the modified layers are uploaded.

```
student@workstation:~$ podman push \
registry.lab.example.com:5000/student/webserver-bootc
...output omitted...
Writing manifest to image destination
```

6. On the serverc machine, deploy the new container image and reboot the system to apply the upgrade.
  - 6.1. Use the `podman login` command to copy the registry credentials to the `/etc/ostree` directory. Use the `student` username and `redhat` as the password. This operation ensures that the `bootc` command does not prompt for authentication details in the future.

```
[root@serverc ~]# podman login registry.lab.example.com:5000 \
--authfile=/etc/ostree/auth.json
Username: student
Password: redhat
Login Succeeded!
```

- 6.2. Use the `bootc status` command to check the current status of the staged, booted, and rollback properties.

There is no image staged for update, and the booted image is the previous version. The rollback image is also not set.

```
[root@serverc ~]# bootc status
• Booted image: registry.lab.example.com:5000/student/webserver-bootc
  Digest: sha256:1270...15ed
  Version: 10.0 (2025-06-23 18:49:52.323586597 UTC)
```

- 6.3. Use the `bootc upgrade` command to stage the new image for update.

```
[root@serverc ~]# bootc upgrade
layers already present: 66; layers needed: 1 (33.0 MB)
Fetched layers: 31.43 MiB in 4 seconds (7.00 MiB/s)
  Deploying: done (6 seconds)
Queued for next boot: registry.lab.example.com:5000/student/webserver-bootc
  Version: 10.0
  Digest: sha256:52e3...051a
Total new layers: 67    Size: 782.0 MB
Removed layers:   1    Size: 22.7 MB
Added layers:     1    Size: 33.0 MB
```

- 6.4.** Use the `bootc status` command to check the new status of the staged, booted, and rollback properties.

The `registry.lab.example.com:5000/student/webserver-bootc` container image with the new SHA256 digest is staged for update, but it will not take effect until the next reboot. The booted image is still the previous version, and the rollback image is not set yet.

```
[root@serverc ~]# bootc status
  Staged image: registry.lab.example.com:5000/student/webserver-bootc
    Digest: sha256:52e3...051a
    Version: 10.0 (2025-06-23 19:14:17.135093334 UTC)

  • Booted image: registry.lab.example.com:5000/student/webserver-bootc
    Digest: sha256:1270...15ed
    Version: 10.0 (2025-06-23 18:49:52.323586597 UTC)
```

- 6.5.** Reboot the `serverc` machine to apply the changes.

```
[root@serverc ~]$ reboot
...output omitted...
Connection to serverc closed.
student@workstation:~$
```

- 7.** Log in to `serverc` machine again and verify that the image version has changed.

- 7.1.** Log in to the `serverc` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh serverc
student@serverc's password: student
...output omitted...
[student@serverc ~]$ sudo -i
[sudo] password for student: student
[root@serverc ~]#
```

- 7.2.** Use the `bootc status` command to check the new state of the booted system. The output indicates that no image is staged, the booted image changed to the new SHA256 digest, and the rollback image is the previous SHA256 digest.

```
[root@serverc ~]# bootc status
• Booted image: registry.lab.example.com:5000/student/webserver-bootc
  Digest: sha256:52e3...051a
  Version: 10.0 (2025-06-23 19:14:17.135093334 UTC)

  Rollback image: registry.lab.example.com:5000/student/webserver-bootc
  Digest: sha256:1270...15ed
  Version: 10.0 (2025-06-23 18:49:52.323586597 UTC)
```

- 8.** On the workstation machine, verify that the web server supports HTTPS.

- 8.1.** Use the `curl --insecure` command to test the website. Ignore any errors about the self-signed SSL certificate.

Despite the image upgrade, the contents of the `/var` directory remain unchanged. As expected, the Hello from a modified image-mode based installation! content does not display.

```
student@workstation:~$ curl --insecure https://serverc.lab.example.com
Hello image mode for RHEL!
Local content edited on serverc.
```

- 9.** On the `serverc` machine, verify that the `mod_ssl` and `vim-enhanced` packages are installed

```
[root@serverc ~]# rpm -q mod_ssl vim-enhanced
mod_ssl-2.4.62-4.el10.x86_64
vim-enhanced-9.1.083-2.el10.x86_64
```

- 10.** On the `serverc` machine, perform a rollback to revert the image mode-based server to the previous version.

- 10.1.** Use the `bootc rollback` command to stage a rollback to the previous image.

```
[root@serverc ~]# bootc rollback  
Next boot: rollback deployment
```

- 10.2.** Reboot the `serverc` machine to apply the rollback.

```
[root@serverc ~]# reboot  
...output omitted...  
Connection to serverc closed.  
student@workstation:~$
```

- 11.** On the `workstation` machine, verify that the rollback was successful.

- 11.1.** Use the `curl` command to confirm that the `mod_ssl` package is no longer present in the server.

```
student@workstation:~$ curl --insecure https://serverc.lab.example.com  
curl: (7) Failed to connect to serverc.lab.example.com port 443 after 2 ms:  
Could not connect to server
```

- 12.** On the `serverc` machine, verify that the `mod_ssl` and `vim-enhanced` packages are not present.

- 12.1.** Log in to the `serverc` machine as the `student` user. Use `student` as the password.

```
student@workstation:~$ ssh serverc  
student@serverc's password: student  
...output omitted...  
[student@serverc ~]$
```

- 12.2.** Verify that the `mod_ssl` and `vim-enhanced` packages are not installed.

```
[student@serverc ~]$ rpm -q mod_ssl vim-enhanced  
package mod_ssl is not installed  
package vim-enhanced is not installed
```

- 13.** Close all other terminal sessions and return to the `workstation` machine as the `student` user.

```
[student@serverc ~]$ exit  
logout  
Connection to serverc closed.  
student@workstation:~$
```

## Finish

On the workstation machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish image-manage
```

## 18.9. Summary

---

In this lesson, you learned to install and configure Red Hat Enterprise Linux 10 in image mode.

Image mode is a new approach to system administration that uses bootable container (`bootc`) images to deploy and manage the operating system.

You can use container-native tools, such as Podman, to build the container image. The build process relies on a standard `Containerfile` that includes the official `rhel-bootc` base image and additional build instructions as needed.

To deploy the container image to the target system, you must first push the image to a container registry. Afterward, you can boot the Anaconda installer to kickstart your systems in image mode, or use the `bootc-image-builder` tool to create disk images for cloud or virtualization environments.

After the target system is installed, you can manage it with the `bootc` tool. The command enables you to query the current status, and to perform image upgrades or rollbacks as needed.

## Chapter 19.

### Comprehensive Review

---

#### Goal

Review tasks from *Red Hat System Administration II*.

#### Sections

- Comprehensive Review
- Fix Boot Issues and Maintain Servers (Lab)
- Configure and Manage File Systems and Storage (Lab)
- Configure and Manage Server Security (Lab)
- Run Containers (Lab)

#### Lab

- Fix Boot Issues and Maintain Servers
- Configure and Manage File Systems and Storage
- Configure and Manage Server Security
- Run Containers

## 19.1. Comprehensive Review

---

### Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills that you learned in *Red Hat System Administration II*.

### Reviewing Red Hat System Administration II

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter. Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

#### **Chapter 1, Shell Scripting and the Command Line**

Write and run simple shell scripts, and use shell scripting features to efficiently run commands at the shell prompt.

#### **Chapter 2, Using Regular Expressions for Practical Applications**

Efficiently complete system administration tasks by using regular expressions to match text.

#### **Chapter 3, Scheduling User Tasks**

Schedule programs to run in the future, either at a specific time and date or on a recurring basis, as a regular user.

#### **Chapter 4, Scheduling System Tasks**

Schedule system programs that must run on a recurring basis to support daemons or operating system functions.

#### **Chapter 5, Analyzing and Storing Logs**

Locate and interpret system logs for troubleshooting purposes, and ensure accurate timestamps for log events.

#### **Chapter 6, Managing Security with SELinux**

Protect systems and manage security by using SELinux.

#### **Chapter 7, Archiving Files**

Create compressed archives of files so that they can be backed up and transferred to other systems.

#### **Chapter 8, Transferring Files**

Securely transfer files from one system to another.

## **Chapter 9, Tuning System Performance**

Improve system performance by setting a tuning profile and by adjusting the scheduling priority of specific processes.

## **Chapter 10, Managing Basic Storage**

Manage storage devices by creating partitions, file systems, and swap spaces from the command line.

## **Chapter 11, Managing Storage with Logical Volume Manager**

Use Logical Volume Manager (LVM) to manage logical volumes that can contain file systems and swap spaces.

## **Chapter 12, Controlling and Troubleshooting the Boot Process**

Manage how the system boots to control which services start and to troubleshoot and repair boot-time problems.

## **Chapter 13, Recovering Superuser Access**

Gain administrative access to a system when the superuser password is unknown or is locked.

## **Chapter 14, Managing Network Security**

Control network connections to services by using the system firewall, and network services that can bind to particular ports by using SELinux.

## **Chapter 15, Accessing Network-attached Storage**

Access network-attached storage that the Network File System (NFS) protocol provides, either manually or by using the automounter.

## **Chapter 16, Installing Red Hat Enterprise Linux**

Install Red Hat Enterprise Linux in package mode, either interactively or by using Kickstart.

## **Chapter 17, Managing Containers with Podman**

Manage containers and container images with the latest version of container management tools.

## **Chapter 18, Working with Image-based Red Hat Enterprise Linux**

Create, use, install, and upgrade containers and servers that use image-based installation management.

## 19.2. Lab

# Fix Boot Issues and Maintain Servers

Troubleshoot and repair boot problems, update the system's default boot target, and schedule tasks to run on a recurring schedule as a nonprivileged user.

### Outcomes

- Diagnose boot-related issues.
- Change the default boot target.
- Schedule recurring jobs to run as a nonprivileged user.

#### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

### Prerequisites

#### Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start comprevew-review1
```

### Specifications

You are a Linux system administrator at Quasar Technologies, an IT services company that provides managed hosting and cloud infrastructure to clients. Your duties include maintaining server stability, troubleshooting issues, and implementing automation to support the operations of clients.

- On the workstation machine, run the /tmp/rhcsa-break1 script. This script creates an issue with the boot process on the serverb machine and then reboots the serverb machine.

```
student@workstation:~$ bash /tmp/rhcsa-break1
```

- A client opened a case reporting that the serverb machine is down after rebooting the machine during maintenance tasks. The case description mentions that the server is stuck at boot with errors related to mounting a partition. You have been asked to investigate and to fix the issue.
  - Access the console for the serverb machine and boot into emergency mode for maintenance. Look for any error messages on the console that relate to a problem with mounting a partition. Use redhat as the password.
  - Remount the root file system in read/write mode.
  - Try to mount all the other file systems and identify the file system that fails to mount.
  - Edit the /etc/fstab file to fix the issue. Remove or comment out the incorrect line.
  - Reboot the serverb machine and confirm that the system now boots without errors.

- On the workstation machine, run the /tmp/rhcsa-break2 script. This script changes the default target from the multi-user target to the graphical target on the serverb machine and then reboots the serverb machine.

```
student@workstation:~$ bash /tmp/rhcsa-break2
```

- To reduce the unnecessary consumption of CPU and memory resources on the serverb machine, you have been asked to ensure that the multi-user target is the default target.
  - Determine the default target on the serverb machine, and change to the multi-user target if it is not the default target.
  - Ensure that the configuration persists after reboot.
- On the serverb machine, the student user stores some important documents in their home directory. To prevent data loss, the student user's home directory must be backed up. You have been assigned the task of scheduling a recurring job that meets the following criteria:
  - A backup script is available at the /home/student/RH134/labs/comprevie-review1/backup-home.sh file. The backup-home.sh script backs up the /home/student directory from the serverb machine to the severa machine in the /home/student/serverb-backup directory.

- Verify the permissions of the backup-home.sh script and ensure that the script is executable.
  - On the serverb machine, schedule a recurring job as the student user that executes the backup-home.sh script hourly between 7 PM and 9 PM every day, except on Saturday and Sunday.
- Reboot the serverb machine and wait for the boot to complete before grading.

## Evaluation

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

### Note

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade comprevew-review1
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish comprevew-review1
```

## Solution

Troubleshoot and repair boot problems, update the system's default boot target, and schedule tasks to run on a recurring schedule as a nonprivileged user.

## Outcomes

- Diagnose boot-related issues.
- Change the default boot target.
- Schedule recurring jobs to run as a nonprivileged user.

### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

## Prerequisites

### Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start compreview-review1
```

## Specifications

You are a Linux system administrator at Quasar Technologies, an IT services company that provides managed hosting and cloud infrastructure to clients. Your duties include maintaining server stability, troubleshooting issues, and implementing automation to support the operations of clients.

- On the workstation machine, run the /tmp/rhcsa-break1 script. This script creates an issue with the boot process on the serverb machine and then reboots the serverb machine.

```
student@workstation:~$ bash /tmp/rhcsa-break1
```

- A client opened a case reporting that the serverb machine is down after rebooting the machine during maintenance tasks. The case description mentions that the server is stuck at boot with errors related to mounting a partition. You have been asked to investigate and to fix the issue.
  - Access the console for the serverb machine and boot into emergency mode for maintenance. Look for any error messages on the console that relate to a problem with mounting a partition. Use redhat as the password.
  - Remount the root file system in read/write mode.
  - Try to mount all the other file systems and identify the file system that fails to mount.
  - Edit the /etc/fstab file to fix the issue. Remove or comment out the incorrect line.
  - Reboot the serverb machine and confirm that the system now boots without errors.

- On the workstation machine, run the /tmp/rhcsa-break2 script. This script changes the default target from the multi-user target to the graphical target on the serverb machine and then reboots the serverb machine.

```
student@workstation:~$ bash /tmp/rhcsa-break2
```

- To reduce the unnecessary consumption of CPU and memory resources on the serverb machine, you have been asked to ensure that the multi-user target is the default target.
  - Determine the default target on the serverb machine, and change to the multi-user target if it is not the default target.
  - Ensure that the configuration persists after reboot.
- On the serverb machine, the student user stores some important documents in their home directory. To prevent data loss, the student user's home directory must be backed up. You have been assigned the task of scheduling a recurring job that meets the following criteria:
  - A backup script is available at the /home/student/RH134/labs/comprevie-review1/backup-home.sh file. The backup-home.sh script backs up the /home/student directory from the serverb machine to the severa machine in the /home/student/serverb-backup directory.

- Verify the permissions of the backup-home.sh script and ensure that the script is executable.
  - On the serverb machine, schedule a recurring job as the student user that executes the backup-home.sh script hourly between 7 PM and 9 PM every day, except on Saturday and Sunday.
- Reboot the serverb machine and wait for the boot to complete before grading.

## Solution

One possible solution to this exercise is provided. You might use a different method to achieve the specified outcome.

To better prepare you for real-world challenges, the solution that is provided might not include commands, code blocks, or steps for all tasks.

If the grading script fails, then the system displays a hint about what content to review.

1. On the workstation machine, run the /tmp/rhcsa-break1 script, and wait for the serverb machine to boot.

```
student@workstation:~$ bash /tmp/rhcsa-break1  
...output omitted...
```

2. Open the serverb console, where the machine boots into the emergency mode, and then resolve the boot issue.
  - 2.1. Use the mount command to remount the root file system in read/write mode.
  - 2.2. Use the mount command to mount the other file systems to find the file system that the system cannot mount.
  - 2.3. Edit the /etc/fstab file to remove the incorrect line, and use the systemctl daemon-reload command to update the systemd daemon.
  - 2.4. Reboot the serverb machine, and wait for the boot process to complete.
3. On the workstation machine, run the /tmp/rhcsa-break2 script and wait for the serverb machine to reboot before proceeding.
4. Set the multi-user.target as the default target on the serverb machine.
5. Reboot the serverb machine.

- For use by Wong Chia Cheong jason.wong76 jason.wong@trainocate.com Copyright © 2025 Red Hat, Inc.
- 6. Wait until the serverb machine boots, and then log in as the student user.
  - 7. Make the backup-home.sh script executable.

```
[student@serverb ~]$ chmod +x ~/RH134/labs/comprevew-review1/backup-home.sh
```

- 8. Edit the student user crontab file by using the crontab command.
- 9. Add the following content to the crontab file:

```
0 19-21 * * Mon-Fri /home/student/RH134/labs/comprevew-review1/backup-home.sh
```

- 10. Reboot the serverb machine and wait for the boot process to complete.

## Evaluation

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

### Note

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade comprevew-review1
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

- On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish comprevew-review1
```

## 19.3. Lab

# Configure and Manage File Systems and Storage

Manage users and groups, use Logical Volume Manager (LVM) to create logical volumes and swap space, persistently mount the Network File System (NFS) share, and schedule a task to clear temporary files on the system.

## Outcomes

- Create a logical volume.
- Mount a network file system.
- Create a swap partition that is automatically activated at boot.
- Configure a directory to store temporary files.

### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

## Prerequisites

### Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start comprevew-review2
```

## Specifications

You are a Linux system administrator at Quasar Technologies, an IT services company that provides managed hosting and cloud infrastructure to clients. You have been tasked with implementing the following infrastructure changes to improve storage management, access to network resources, and user access controls.

- The company requires a dedicated, scalable, and efficient storage space on the `serverb` machine for new game assets. Configure storage to meet the following specifications.
  - Use the available `/dev/sdb` disk to create the partition for backing up the physical volume. Choose an appropriate size for the partition to meet the other requirements.
  - Create a 1 GiB `vol_home` logical volume in a new 2 GiB volume group named `extra_storage`. The volume group should use a partition created on the `/dev/sdb` disk as its physical volume.
  - The `vol_home` logical volume should be formatted with an XFS file system and mounted persistently on the `/user-homes` directory.
- The `servera` machine exports the `/share` NFS export, which contains files that all development workstations require. Make the `/share` NFS export from the `servera` machine available on the `serverb` machine automatically at boot.
  - Use the `/local-share` directory as the mount point on the `serverb` machine.
  - Configure the `serverb` machine to persistently mount the `servera:/share` NFS export on the `/local-share` directory.
- Due to increased memory demands from the game engineers, provide swap space to prevent the `serverb` machine from crashing during peak loads.
  - On the `serverb` machine, create a 512 MiB swap partition on the `/dev/sdc` disk.
  - Persistently activate the swap partition.
- The production team needs access to game assets on the `serverb` machine. Use a shared group for permission management.
  - Create the `production` user group.
  - Create the `production1`, `production2`, `production3`, and `production4` users. All four users should be members of the `production` supplementary group and must use `redhat` as the password.
- Continuous testing is generating many temporary build artifacts that must be cleaned up automatically to free up disk space. On the `serverb` machine, configure a new directory for temporary storage that meets the following specifications:

- Configure the `/run/volatile` directory to store temporary files.
- Files in the `/run/volatile` directory that are not accessed for more than 30 seconds are automatically deleted by the system.
- Set `0700` as the octal permissions for the `/run/volatile` directory.
- Use the `/etc/tmpfiles.d/volatile.conf` file to configure the time-based deletion of the files in the `/run/volatile` directory.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

### Note

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade compreview-review2
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

On the workstation machine, change to the student user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish compreview-review2
```

## Solution

Manage users and groups, use Logical Volume Manager (LVM) to create logical volumes and swap space, persistently mount the Network File System (NFS) share, and schedule a task to clear temporary files on the system.

## Outcomes

- Create a logical volume.
- Mount a network file system.
- Create a swap partition that is automatically activated at boot.
- Configure a directory to store temporary files.

### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

## Prerequisites

### Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start comprevew-review2
```

## Specifications

You are a Linux system administrator at Quasar Technologies, an IT services company that provides managed hosting and cloud infrastructure to clients. You have been tasked with implementing the

following infrastructure changes to improve storage management, access to network resources, and user access controls.

- The company requires a dedicated, scalable, and efficient storage space on the `serverb` machine for new game assets. Configure storage to meet the following specifications:
  - Use the available `/dev/sdb` disk to create the partition for backing up the physical volume. Choose an appropriate size for the partition to meet the other requirements.
  - Create a 1 GiB `vol_home` logical volume in a new 2 GiB volume group named `extra_storage`. The volume group should use a partition created on the `/dev/sdb` disk as its physical volume.
  - The `vol_home` logical volume should be formatted with an XFS file system and mounted persistently on the `/user-homes` directory.
- The `servera` machine exports the `/share` NFS export, which contains files that all development workstations require. Make the `/share` NFS export from the `servera` machine available on the `serverb` machine automatically at boot.
  - Use the `/local-share` directory as the mount point on the `serverb` machine.
  - Configure the `serverb` machine to persistently mount the `servera:/share` NFS export on the `/local-share` directory.
- Due to increased memory demands from the game engineers, provide swap space to prevent the `serverb` machine from crashing during peak loads.
  - On the `serverb` machine, create a 512 MiB swap partition on the `/dev/sdc` disk.
  - Persistently activate the swap partition.
- The production team needs access to game assets on the `serverb` machine. Use a shared group for permission management.
  - Create the `production` user group.
  - Create the `production1`, `production2`, `production3`, and `production4` users. All four users should be members of the `production` supplementary group and must use `redhat` as the password.
- Continuous testing is generating many temporary build artifacts that must be cleaned up automatically to free up disk space. On the `serverb` machine, configure a new directory for temporary storage that meets the following specifications:
  - Configure the `/run/volatile` directory to store temporary files.

- Files in the /run/volatile directory that are not accessed for more than 30 seconds are automatically deleted by the system.
- Set 0700 as the octal permissions for the /run/volatile directory.
- Use the /etc/tmpfiles.d/volatile.conf file to configure the time-based deletion of the files in the /run/volatile directory.

## Solution

One possible solution to this exercise is provided. You might use a different method to achieve the specified outcome.

To better prepare you for real-world challenges, the solution that is provided might not include commands, code blocks, or steps for all tasks.

If the grading script fails, then the system displays a hint about what content to review.

1. Log in to the serverb machine as the student user by using the ssh student@serverb command, and use the sudo -i command to switch to the root user.
2. Use the following command to configure the vol\_home logical volume, and format it with the XFS file-system type.

```
[root@serverb ~]# parted /dev/sdb mklabel msdos && \
parted /dev/sdb mkpart primary 1MiB 2049MiB && \
parted /dev/sdb set 1 lvm on && \
pvcreate /dev/sdb1 && \
vgcreate extra_storage /dev/sdb1 && \
lvcreate -L 1GiB -n vol_home extra_storage && \
mkdir /user-homes && \
mkfs -t xfs /dev/extra_storage/vol_home
...output omitted...
```

3. Persistently mount the /dev/extra\_storage/vol\_home logical volume on the /user-homes directory.

```
[root@serverb ~]# echo "/dev/extra_storage/vol_home /user-homes xfs defaults 0 0" \
>> /etc/fstab && \
systemctl daemon-reload && \
mount /user-homes
...output omitted...
```

4. Use the following command to mount the /share network file system on the /local-share directory.

```
[root@serverb ~]# mkdir /local-share && \
echo "servera:/share /local-share nfs rw,sync 0 0" >> /etc/fstab && \
systemctl daemon-reload && \
mount /local-share
```

5. Create, activate, and persistently mount the swap partition.

```
[root@serverb ~]# parted /dev/sdc mklabel msdos && \
parted /dev/sdc mkpart primary linux-swap 1MiB 513MiB && \
mkswap /dev/sdc1
[root@serverb ~]# lsblk -o UUID /dev/sdc1
UUID
cc18ccb6-bd29-48a5-8554-546bf3471b69
[root@serverb ~]# echo "UUID=cc18...1b69 swap swap defaults 0 0" >>
/etc/fstab
[root@serverb ~]# swapon -a
```

6. Use the following commands to create the production user group, to create and add the users that must have the production supplementary group, and to set the required user passwords.

```
[root@serverb ~]# groupadd production
[root@serverb ~]# for i in 1 2 3 4; do useradd -G production production$i;
done
[root@serverb ~]# for i in 1 2 3 4; do echo redhat | passwd --stdin
production$i; done
```

7. Use the following command to configure the /run/volatile directory to store temporary files.

```
[root@serverb ~]# echo "d /run/volatile 0700 root root 30s" > \
/etc/tmpfiles.d/volatile.conf
[root@serverb ~]# systemd-tmpfiles --create /etc/tmpfiles.d/volatile.conf
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

### Note

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade compreview-review2
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

On the workstation machine, change to the student user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish compreview-review2
```

## 19.4. Lab

# Configure and Manage Server Security

Configure secure access to remote servers, adjust firewall settings, and troubleshoot SELinux.

## Outcomes

- Configure SSH key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

## Prerequisites

### Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start comprevew-review3
```

## Specifications

You are a Linux system administrator at Quasar Technologies, an IT services company that provides managed hosting and cloud infrastructure to clients. You are responsible for implementing critical

security and configuration changes to ensure secure remote access, proper file system mounting, and web-based reporting.

- The company requires secure, passwordless access from the `serverb` machine to the `servera` machine for automated data transfers. The `student` user on the `servera` machine must be able to log in to the `servera` machine via SSH without entering a password.
  - On the `serverb` machine, generate an SSH key pair for the `student` user. Do not protect the private key with a passphrase.
  - Send the public key of the newly generated SSH key pair to the `student` user on the `servera` machine.
  - Verify that the `student` user can log in to the `servera` machine from the `serverb` machine without entering a password.
- You are investigating potential permission issues with the `/user-homes/production5` directory on the `servera` machine. You have decided to relax SELinux for debugging.
  - Verify the `/user-homes/production5` directory permissions.
  - Configure SELinux to run in permissive mode by default.
  - Reboot the `servera` machine and confirm that the SELinux mode is permissive.
- The `servera` machine exports the `production5` user's home directory as the `servera:/user-homes/production5` NFS export. Configure the `autofs` service to mount the network share on the `/localhome/production5` directory on the `serverb` machine.
  - Install the required package to set up the `autofs` service.
  - Configure the `autofs` service to mount the `servera:/user-homes/production5` NFS export on the `/localhome/production5` directory on the `serverb` machine.
- On the `serverb` machine, adjust the appropriate SELinux Boolean so that the `production5` user can use the NFS-mounted home directory after authenticating with an SSH key. Use `redhat` as the password of the `production5` user.
  - On the `servera` machine, as the `production5` user, generate a new SSH key pair for authentication. Use `redhat` as the password for the `production5` user.
  - Transfer the public key of the SSH key pair to the `production5` user on the `serverb` machine. Use `redhat` as the password for the `production5` user.
  - Use SSH public key-based authentication instead of password-based authentication to log in to the `serverb` machine as the `production5` user. Verify that the command fails.

- Adjust the `use_nfs_home_dirs` SELinux Boolean on the `serverb` machine to allow NFS to use home directories. Use `redhat` as the password.
- Verify and confirm that you are now able to use SSH public key-based authentication instead of password-based authentication to log in to the `serverb` machine as the `production5` user.
- For security reasons, you need to block all connection requests from the `servera` machine to the `serverb` machine.
  - On the `serverb` machine, adjust the firewall settings to block all connection requests that originate from the `servera` machine. Use the `servera` IPv4 address (`172.25.250.10`) to configure the firewall rule.
- A junior administrator configured the Apache HTTP Server on the `serverb` machine to listen on the `30080/TCP` port for connections. However, the service fails to start. You have been asked to investigate and fix this issue. You are also asked to adjust the firewall settings appropriately so that the `30080/TCP` port is open for incoming connections.
  - Attempt to restart the web service. Inspect appropriate log files to determine the reason for service failure, including possible SELinux policy violations. When identified, fix the issue and ensure that the service starts.
  - Add the `30080/TCP` port to the default public zone to allow incoming connections.

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

### Note

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade compreview-review3
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

On the workstation machine, change to the student user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous

exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish compreview-review3
```

## Solution

Configure secure access to remote servers, adjust firewall settings, and troubleshoot SELinux.

## Outcomes

- Configure SSH key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

## Prerequisites

### ⚠ Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

- As the student user on the workstation machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start comprevew-review3
```

## Specifications

You are a Linux system administrator at Quasar Technologies, an IT services company that provides managed hosting and cloud infrastructure to clients. You are responsible for implementing critical security and configuration changes to ensure secure remote access, proper file system mounting, and web-based reporting.

- The company requires secure, passwordless access from the serverb machine to the servera machine for automated data transfers. The student user on the serverb machine must be able to log in to the servera machine via SSH without entering a password.
  - On the serverb machine, generate an SSH key pair for the student user. Do not protect the private key with a passphrase.
  - Send the public key of the newly generated SSH key pair to the student user on the servera machine.
  - Verify that the student user can log in to the servera machine from the serverb machine without entering a password.
- You are investigating potential permission issues with the /user-homes/production5 directory on the servera machine. You have decided to relax SELinux for debugging.
  - Verify the /user-homes/production5 directory permissions.
  - Configure SELinux to run in permissive mode by default.
  - Reboot the servera machine and confirm that the SELinux mode is permissive.
- The servera machine exports the production5 user's home directory as the servera:/user-homes/production5 NFS export. Configure the autofs service to mount the network share on the /localhome/production5 directory on the serverb machine.
  - Install the required package to set up the autofs service.
  - Configure the autofs service to mount the servera:/user-homes/production5 NFS export on the /localhome/production5 directory on the serverb machine.
- On the serverb machine, adjust the appropriate SELinux Boolean so that the production5 user can use the NFS-mounted home directory after authenticating with an SSH key. Use redhat as the password of the production5 user.
  - On the servera machine, as the production5 user, generate a new SSH key pair for authentication. Use redhat as the password for the production5 user.
  - Transfer the public key of the SSH key pair to the production5 user on the serverb machine. Use redhat as the password for the production5 user.
  - Use SSH public key-based authentication instead of password-based authentication to log in to the serverb machine as the production5 user. Verify that the command fails.
  - Adjust the use\_nfs\_home\_dirs SELinux Boolean on the serverb machine to allow NFS to use home directories. Use redhat as the password.

- For use by Wong Chia Cheong jason.wong76.jason.wong@trainoate.com Copyright © 2025 Red Hat, Inc.
- Verify and confirm that you are now able to use SSH public key-based authentication instead of password-based authentication to log in to the serverb machine as the production5 user.
  - For security reasons, you need to block all connection requests from the servera machine to the serverb machine.
    - On the serverb machine, adjust the firewall settings to block all connection requests that originate from the servera machine. Use the servera IPv4 address (172.25.250.10) to configure the firewall rule.
  - A junior administrator configured the Apache HTTP Server on the serverb machine to listen on the 30080/TCP port for connections. However, the service fails to start. You have been asked to investigate and fix this issue. You are also asked to adjust the firewall settings appropriately so that the 30080/TCP port is open for incoming connections.
    - Attempt to restart the web service. Inspect appropriate log files to determine the reason for service failure, including possible SELinux policy violations. When identified, fix the issue and ensure that the service starts.
    - Add the 30080/TCP port to the default public zone to allow incoming connections.

## Solution

One possible solution to this exercise is provided. You might use a different method to achieve the specified outcome.

To better prepare you for real-world challenges, the solution that is provided might not include commands, code blocks, or steps for all tasks.

If the grading script fails, then the system displays a hint about what content to review.

1. Log in to the serverb machine as the student user.
2. Generate an SSH key pair. Do not protect the private key with a passphrase.

```
[student@serverb ~]$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/student/.ssh/id_ed25519):
Created directory '/home/student/.ssh'.
Enter passphrase for "/home/student/.ssh/id_ed25519" (empty for no
passphrase):
Enter same passphrase again:
...output omitted...
```

3. Send the public key of the newly generated SSH key pair to the student user on the servera machine.

```
[student@serverb ~]$ ssh-copy-id -i ~/.ssh/id_ed25519.pub student@servera
...output omitted...
```

4. Use the newly created SSH key to log in to the servera machine as the student user.

```
[student@serverb ~]$ ssh student@servera
...output omitted...
Last login: Mon Jul 21 19:39:08 2025 from 172.25.250.11
[student@servera ~]$
```

5. On the servera machine, set the SELINUX parameter to the permissive value in the /etc/selinux/config file. Then reboot the system by using the sudo systemctl reboot command.

```
...output omitted...
SELINUX=permissive
...output omitted...
```

6. As the root user on the serverb machine, install the autofs package, create the autoofs map files, and restart the autoofs service.

```
[root@serverb ~]# dnf install -y autofs && \
echo "/- /etc/auto.production5" > /etc/auto.master.d/production5.autofs
...output omitted...
Complete!
[root@serverb ~]# getent passwd production5
production5:x:5001:5001::/localhome/production5:/bin/bash
[root@serverb ~]# echo \
"/localhome/production5 -rw servera.lab.example.com:/user-homes/production5"
\
> /etc/auto.production5 && \
systemctl restart autofs
```

7. Open a new terminal window and log in to the servera machine as the student user. Switch to the production5 user, generate a SSH key pair without a passphrase, and send the SSH key pair to the production5 user on the serverb machine.

```
[student@servera ~]$ su - production5
Password: redhat
[production5@servera ~]$ ssh-keygen
...output omitted...
The key's randomart image is:
+--[ED25519 256]--+
|       ..E=+=.. . |
|       +.=o+ X   |
|       o..Oo=o * o |
|       o.=o+oo o . |
|       . .o*S. o o |
|       .... o     |
|       .. . .      |
|       o. . .      |
|       ...          |
+---[SHA256]---+
[production5@servera ~]$ ssh-copy-id production5@serverb
...output omitted...
Number of key(s) added: 1
```

Now try logging into the machine, with: "ssh 'production5@serverb'" and check to make sure that only the key(s) you wanted were added.

Exit and close the terminal that is connected to the servera machine as the production5 user. Keep open the terminal that is connected to the serverb machine as the root user.

8. As the root user on the serverb machine, set the use\_nfs\_home\_dirs SELinux Boolean to true.

```
[root@serverb ~]# setsebool -P use_nfs_home_dirs true
```

9. On the serverb machine, adjust and reload the firewall settings to block all connection requests that originate from the servera machine.

```
[root@serverb ~]# firewall-cmd --add-source=172.25.250.10/32 \
--zone=block --permanent && \
firewall-cmd --reload
success
success
```

10. On the serverb machine, use the systemctl restart and systemctl status commands on the httpd.service unit to investigate the problem with the Apache HTTP server. Then use the sealert -a /var/log/audit/audit.log command to determine whether a SELinux policy is preventing the httpd service from binding to the 30080/TCP port.

```
[root@serverb ~]# systemctl restart httpd.service
Job for httpd.service failed because the control process exited with error
code.
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for
details.
[root@serverb ~]# systemctl status httpd.service
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset:
disabled)
     Active: failed (Result: exit-code) since Mon 2025-07-21 19:49:23 UTC;
25s ago
...output omitted...
Jul 21 19:49:23 serverb systemd[1]: Starting httpd.service - The Apache HTTP
Server...
Jul 21 19:49:23 serverb (httpd)[3077]: httpd.service: Referenced but unset
environment variable evaluates >
Jul 21 19:49:23 serverb httpd[3077]: (13)Permission denied: AH00072:
make_sock: could not bind to address >
Jul 21 19:49:23 serverb httpd[3077]: (13)Permission denied: AH00072:
make_sock: could not bind to address >
Jul 21 19:49:23 serverb httpd[3077]: no listening sockets available, shutting
down
Jul 21 19:49:23 serverb httpd[3077]: AH00015: Unable to open logs
Jul 21 19:49:23 serverb systemd[1]: httpd.service: Main process exited,
code=exited, status=1/FAILURE
Jul 21 19:49:23 serverb systemd[1]: httpd.service: Failed with result 'exit-
code'.
Jul 21 19:49:23 serverb systemd[1]: Failed to start httpd.service - The
Apache HTTP Server.
```

11. Use the following commands to set the appropriate SELinux context, restart the httpd service, and update the firewall rules.

```
[root@serverb ~]# semanage port -a -t http_port_t -p tcp 30080
[root@serverb ~]# systemctl restart httpd
[root@serverb ~]# firewall-cmd --add-port=30080/tcp --permanent
success
[root@serverb ~]# firewall-cmd --reload
success
```

## Evaluation

As the student user on the workstation machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

 **Note**

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade comprevew-review3
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

On the workstation machine, change to the student user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish comprevew-review3
```

## 19.5. Lab

### Run Containers

---

Create a container image, push the container image to a registry, and create a detached container.

#### Outcomes

- Create a container image from a custom Containerfile.
- Push the custom container image to a remote registry.
- Create a detached container with port mapping.

#### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

#### Prerequisites

##### Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

- As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start compreview-review4
```

#### Specifications

You work for Supernova, which uses Podman for container management. As the Cloud Infrastructure Engineer, you are tasked with configuring a development environment on the serverb machine for a new application. The application requires a web server container that is created from a customized container image.

- You are asked to create the `podmgr` dedicated user on the `serverb` machine for container management. The user requires access to the company's internal container registry at `registry.lab.example.com:5000`.
  - Create the `podmgr` user and set `redhat` as the password for the user.
  - Log in to the `registry.lab.example.com:5000` registry to verify the local registry configuration. Use the `student` user and `redhat` as the password.
- You are asked to deploy a detached web container as the `podmgr` user on the `serverb` machine.

### Important

Open a new terminal tab to log in as the `podmgr` user and keep the SSH session open until you successfully grade all tasks of this activity. Containers do not persist after you exit the SSH session. Exiting the SSH session before you grade this activity will result in failed tasks. Configuring persistent containers is out of the scope of this lesson.

- In the `~/http-dev` directory, create the `http-server:9.0` container image from a custom Containerfile that uses the `rhel10/httpd-24` base image.
- The custom Containerfile must create an `index.html` file in the `/var/www/html` directory in the container. The message inside the `index.html` file must be `Welcome to the Supernova containerized webserver..`
- The custom image must be available in the remote registry.
- Create the `http-srv01` detached container. Use the `http-server:9.0` image from the remote registry.
- Map the `30000` port from the local machine to the `8080` port in the container.
  - You must configure the firewall to make the container accessible from outside the server.
- From the `workstation` machine, verify that the `http-srv01` web container serves the custom message that is defined in the container image.

## Evaluation

As the student user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

### Note

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade compreview-review4
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

On the workstation machine, change to the student user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish compreview-review4
```

## Solution

Create a container image, push the container image to a registry, and create a detached container.

## Outcomes

- Create a container image from a custom Containerfile.
- Push the custom container image to a remote registry.
- Create a detached container with port mapping.

### Note

The allocated time for this activity is 30 minutes.

If you need additional time to complete the task, then you must revisit the course content again, or practice more.

## Prerequisites

### Warning

If you previously used this lab environment to work on other labs, then save any previous lab work that you want to keep, and you must then delete the existing lab environment and create a new one.

If you already reset the lab environment, then you are ready to continue.

As the student user on the workstation machine, run the following lab command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start compreview-review4
```

## Specifications

You work for Supernova, which uses Podman for container management. As the Cloud Infrastructure Engineer, you are tasked with configuring a development environment on the `serverb` machine for a new application. The application requires a web server container that is created from a customized container image.

- You are asked to create the `podmgr` dedicated user on the `serverb` machine for container management. The user requires access to the company's internal container registry at `registry.lab.example.com:5000`.
  - Create the `podmgr` user and set `redhat` as the password for the user.
  - Log in to the `registry.lab.example.com:5000` registry to verify the local registry configuration. Use the `student` user and `redhat` as the password.
- You are asked to deploy a detached web container as the `podmgr` user on the `serverb` machine.

### Important

Open a new terminal tab to log in as the `podmgr` user and keep the SSH session open until you successfully grade all tasks of this activity. Containers do not persist after you exit the SSH session. Exiting the SSH session before you grade this activity will result in failed tasks. Configuring persistent containers is out of the scope of this lesson.

- In the `~/http-dev` directory, create the `http-server:9.0` container image from a custom Containerfile that uses the `rhel10/httpd-24` base image.
- The custom Containerfile must create an `index.html` file in the `/var/www/html` directory in the container. The message inside the `index.html` file must be `Welcome to the Supernova containerized webserver..`
- The custom image must be available in the remote registry.
- Create the `http-srv01` detached container. Use the `http-server:9.0` image from the remote registry.
- Map the `30000` port from the local machine to the `8080` port in the container.
  - You must configure the firewall to make the container accessible from outside the server.
- From the workstation machine, verify that the `http-srv01` web container serves the custom message that is defined in the container image.

## Solution

One possible solution to this exercise is provided. You might use a different method to achieve the specified outcome.

To better prepare you for real-world challenges, the solution that is provided might not include commands, code blocks, or steps for all tasks.

If the grading script fails, then the system displays a hint about what content to review.

1. Log in to the serverb machine as the student user. Create the podmgr user.

```
[student@serverb ~]$ sudo useradd podmgr
[sudo] password for student: student
[student@serverb ~]$ sudo passwd podmgr
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: password updated successfully
```

2. Configure the firewall on the serverb machine to open TCP port 30000. Make the change persistent.

```
[student@serverb ~]$ sudo firewall-cmd --add-port=30000/tcp
success
[student@serverb ~]$ sudo firewall-cmd --add-port=30000/tcp --permanent
success
```

3. Exit the student user session. Log in to the serverb machine as the podmgr user. Then, log in to the container registry as the student user and use redhat as the password.

```
[podmgr@serverb ~]$ podman login registry.lab.example.com:5000
Username: student
Password: redhat
Login Succeeded!
```

4. Create the ~/http-dev directory. In the directory, create a Containerfile that matches the following content:

```
FROM registry.lab.example.com:5000/rhel10/httpd-24
RUN echo "Welcome to the Supernova containerized webserver." \
    > /var/www/html/index.html
```

5. Build the http-server:9.0 custom container image and push it to the registry.

```
[podmgr@serverb ~]$ podman build -t http-server:9.0 ~/http-dev/  
...output omitted...
```

```
[podmgr@serverb ~]$ podman image push localhost/http-server:9.0 \  
registry.lab.example.com:5000/http-server:9.0  
...output omitted...
```

6. Create the http-srv01 detached container. Set up port forwarding from port 30000 on the host to container port 8080.

```
[podmgr@serverb ~]$ podman run -d --name http-srv01 \  
-p 30000:8080 registry.lab.example.com:5000/http-server:9.0
```

7. From another terminal on the workstation machine, verify that the http-srv01 container serves the custom message that is defined in the container image.

```
student@workstation:~$ curl http://serverb.lab.example.com:30000  
Welcome to the Supernova containerized webserver.
```

## Evaluation

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

### Note

If the task fails the grading script, then you receive hints about what content to review.

```
student@workstation:~$ lab grade comprevievw-review4
```

Evaluate the time allocated for this activity against the time it took you to complete it. If you used more time than allocated, then you must revisit the course content again, or practice more.

## Finish

On the workstation machine, change to the student user home directory and use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish comprevievw-review4
```