



# Red Hat Training and Certification

## Student Workbook (ROLE)

Red Hat Enterprise Linux 9.0 RH199

### RHCSA Rapid Track

Edition 4







## Join a community dedicated to learning open source

The Red Hat® Learning Community is a collaborative platform for users to accelerate open source skill adoption while working with Red Hat products and experts.



**Network** with tens of thousands of community members



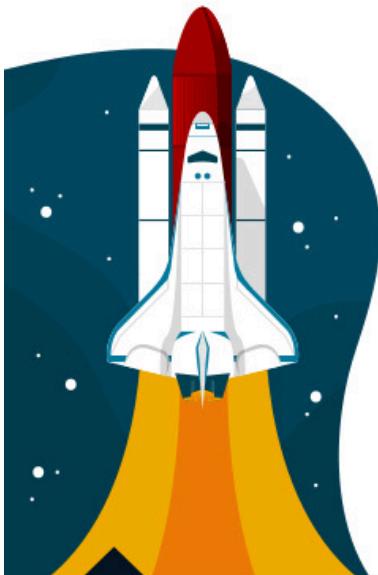
**Engage** in thousands of active conversations and posts



**Join and interact** with hundreds of certified training instructors



**Unlock** badges as you participate and accomplish new goals



This knowledge-sharing platform creates a space where learners can connect, ask questions, and collaborate with other open source practitioners.

**Access** free Red Hat training videos

**Discover** the latest Red Hat Training and Certification news

**Connect** with your instructor - and your classmates - before, after, and during your training course.

**Join** peers as you explore Red Hat products

Join the conversation [learn.redhat.com](https://learn.redhat.com)



Copyright © 2020 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Red Hat logo, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.





# RHCSA Rapid Track

**Red Hat Enterprise Linux 9.0 RH199**  
**RHCSA Rapid Track**  
**Edition 4 20221003**  
**Publication date 20221003**

Authors: Ashish Lingayat, Bernardo Gargallo, Ed Parenti, Jacob Pelchat,  
Mike Kelly, Morgan Weetman, Patrick Gomez  
Course Architect: Philip Sweany  
DevOps Engineer: Artur Glogowski  
Editor: Julian Cable

Copyright © 2022 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are  
Copyright © 2022 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, CloudForms, 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.

Contributors: Adarsh Krishnan, David Sacco, Hemant Chauhan, Roberto Velazquez, Sajith Eyamkuzhy, Samik Sanyal, Yuvaraj Balaraju

<b>Document Conventions</b>	<b>xi</b>
	xi
<b>Introduction</b>	<b>xiii</b>
RHCSA Rapid Track .....	xiii
Orientation to the Classroom Environment .....	xiv
Performing Lab Exercises .....	xviii
<b>1. Access Systems and Obtaining Support</b>	<b>1</b>
Edit Text Files from the Shell Prompt .....	2
Guided Exercise: Edit Text Files from the Shell Prompt .....	6
Configure SSH Key-based Authentication .....	8
Guided Exercise: Configure SSH Key-based Authentication .....	14
Get Help From Red Hat Customer Portal .....	20
Guided Exercise: Get Help From Red Hat Customer Portal .....	25
Detect and Resolve Issues with Red Hat Insights .....	27
Quiz: Detect and Resolve Issues with Red Hat Insights .....	34
Summary .....	36
<b>2. Manage Files from the Command Line</b>	<b>37</b>
Describe Linux File System Hierarchy Concepts .....	38
Quiz: Describe Linux File System Hierarchy Concepts .....	40
Make Links Between Files .....	44
Guided Exercise: Make Links Between Files .....	48
Match File Names with Shell Expansions .....	50
Quiz: Match File Names with Shell Expansions .....	55
Lab: Manage Files from the Command Line .....	59
Summary .....	69
<b>3. Manage Local Users and Groups</b>	<b>71</b>
Describe User and Group Concepts .....	72
Quiz: Describe User and Group Concepts .....	75
Gain Superuser Access .....	79
Guided Exercise: Gain Superuser Access .....	84
Manage Local User Accounts .....	89
Guided Exercise: Manage Local User Accounts .....	92
Manage Local Group Accounts .....	95
Guided Exercise: Manage Local Group Accounts .....	98
Manage User Passwords .....	101
Guided Exercise: Manage User Passwords .....	105
Lab: Manage Local Users and Groups .....	109
Summary .....	115
<b>4. Control Access to Files</b>	<b>117</b>
Manage File System Permissions from the Command Line .....	118
Guided Exercise: Manage File System Permissions from the Command Line .....	122
Manage Default Permissions and File Access .....	125
Guided Exercise: Manage Default Permissions and File Access .....	130
Lab: Control Access to Files .....	134
Summary .....	140
<b>5. Manage SELinux Security</b>	<b>141</b>
Change the SELinux Enforcement Mode .....	142
Guided Exercise: Change the SELinux Enforcement Mode .....	147
Control SELinux File Contexts .....	150
Guided Exercise: Control SELinux File Contexts .....	155
Adjust SELinux Policy with Booleans .....	158
Guided Exercise: Adjust SELinux Policy with Booleans .....	160

Investigate and Resolve SELinux Issues .....	163
Guided Exercise: Investigate and Resolve SELinux Issues .....	167
Lab: Manage SELinux Security .....	171
Summary .....	177
<b>6. Tune System Performance</b>	<b>179</b>
Kill Processes .....	180
Guided Exercise: Kill Processes .....	186
Monitor Process Activity .....	190
Guided Exercise: Monitor Process Activity .....	194
Adjust Tuning Profiles .....	199
Guided Exercise: Adjust Tuning Profiles .....	206
Influence Process Scheduling .....	211
Guided Exercise: Influence Process Scheduling .....	216
Lab: Tune System Performance .....	220
Summary .....	226
<b>7. Schedule Future Tasks</b>	<b>227</b>
Schedule Recurring User Jobs .....	228
Guided Exercise: Schedule Recurring User Jobs .....	231
Schedule Recurring System Jobs .....	234
Guided Exercise: Schedule Recurring System Jobs .....	237
Manage Temporary Files .....	240
Guided Exercise: Manage Temporary Files .....	243
Quiz: Schedule Future Tasks .....	246
Summary .....	250
<b>8. Install and Update Software Packages</b>	<b>251</b>
Register Systems for Red Hat Support .....	252
Quiz: Register Systems for Red Hat Support .....	255
Install and Update Software Packages with DNF .....	257
Guided Exercise: Install and Update Software Packages with DNF .....	266
Enable DNF Software Repositories .....	271
Guided Exercise: Enable DNF Software Repositories .....	274
Lab: Install and Update Software Packages .....	278
Summary .....	284
<b>9. Manage Basic Storage</b>	<b>285</b>
Mount and Unmount File Systems .....	286
Guided Exercise: Mount and Unmount File Systems .....	289
Add Partitions, File Systems, and Persistent Mounts .....	292
Guided Exercise: Add Partitions, File Systems, and Persistent Mounts .....	301
Manage Swap Space .....	305
Guided Exercise: Manage Swap Space .....	309
Lab: Manage Basic Storage .....	313
Summary .....	321
<b>10. Manage Storage Stack</b>	<b>323</b>
Create and Extend Logical Volumes .....	324
Guided Exercise: Create and Extend Logical Volumes .....	335
Manage Layered Storage .....	341
Guided Exercise: Manage Layered Storage .....	347
Lab: Manage Storage Stack .....	352
Summary .....	358
<b>11. Control Services and Boot Process</b>	<b>359</b>
Identify Automatically Started System Processes .....	360
Guided Exercise: Identify Automatically Started System Processes .....	365

Control System Services .....	369
Guided Exercise: Control System Services .....	373
Select the Boot Target .....	377
Guided Exercise: Select the Boot Target .....	383
Reset the Root Password .....	386
Guided Exercise: Reset the Root Password .....	390
Repair File System Issues at Boot .....	392
Guided Exercise: Repair File System Issues at Boot .....	395
Lab: Control the Boot Process .....	398
Summary .....	404
<b>12. Analyze and Store Logs</b>	<b>405</b>
Describe System Log Architecture .....	406
Quiz: Describe System Log Architecture .....	408
Review Syslog Files .....	412
Guided Exercise: Review Syslog Files .....	417
Review System Journal Entries .....	419
Guided Exercise: Review System Journal Entries .....	424
Preserve the System Journal .....	427
Guided Exercise: Preserve the System Journal .....	430
Maintain Accurate Time .....	433
Guided Exercise: Maintain Accurate Time .....	437
Lab: Analyze and Store Logs .....	441
Summary .....	446
<b>13. Manage Networking</b>	<b>447</b>
Validate Network Configuration .....	448
Guided Exercise: Validate Network Configuration .....	454
Configure Networking from the Command Line .....	457
Guided Exercise: Configure Networking from the Command Line .....	464
Edit Network Configuration Files .....	470
Guided Exercise: Edit Network Configuration Files .....	474
Configure Hostnames and Name Resolution .....	478
Guided Exercise: Configure Hostnames and Name Resolution .....	481
Lab: Manage Networking .....	485
Summary .....	490
<b>14. Access Network-Attached Storage</b>	<b>491</b>
Manage Network-Attached Storage with NFS .....	492
Guided Exercise: Manage Network-Attached Storage with NFS .....	495
Automount Network-Attached Storage .....	498
Guided Exercise: Automount Network-Attached Storage .....	502
Lab: Access Network-Attached Storage .....	508
Summary .....	515
<b>15. Manage Network Security</b>	<b>517</b>
Manage Server Firewalls .....	518
Guided Exercise: Manage Server Firewalls .....	526
Lab: Manage Network Security .....	529
Summary .....	537
<b>16. Run Containers</b>	<b>539</b>
Container Concepts .....	540
Quiz: Container Concepts .....	547
Deploy Containers .....	549
Guided Exercise: Deploy Containers .....	559
Manage Container Storage and Network Resources .....	565

Guided Exercise: Manage Container Storage and Network Resources .....	575
Manage Containers as System Services .....	581
Guided Exercise: Manage Containers as System Services .....	587
Lab: Run Containers .....	593
Summary .....	600

<b>17. Comprehensive Review</b>	<b>601</b>
Comprehensive Review .....	602
Lab: Fix Boot Issues and Maintain Servers .....	606
Lab: Configure and Manage File Systems and Storage .....	612
Lab: Configure and Manage Server Security .....	618
Lab: Run Containers .....	628

# 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

They 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

Warnings should not be ignored. Ignoring these admonitions will most likely cause data loss.

## Inclusive Language

Red Hat Training is currently reviewing its use of language in various areas to help remove any potentially offensive terms. This is an ongoing process 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

## RHCSA Rapid Track

The *Red Hat Certified System Administrator Rapid Track (RH199)* course is designed as a rapid training course on system administration of Red Hat Enterprise Linux for IT professionals with significant exposure to Linux. Students with a foundational understanding of the Linux command line will learn key tasks to administer a single Red Hat Enterprise Linux system. This course is an accelerated course. When delivered as an instructor-led course, this course covers training content in 5 days that is normally covered during 10 days in *Red Hat System Administration I* and *Red Hat System Administration II*. Some concepts from those other two courses are covered only briefly in this course or are not covered.

### Course Objectives

- Expand on and extend skills that you gained during the *Red Hat System Administration I (RH124)* course.
- Build the needed skills of an RHCSA-certified Red Hat Enterprise Linux system administrator.

### Audience

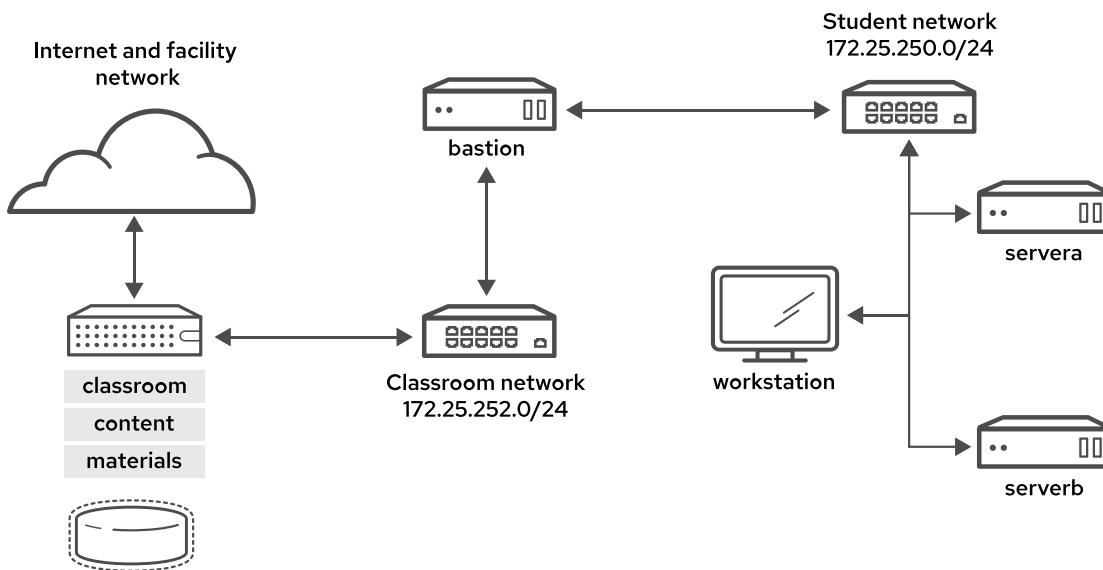
- *RHCSA Rapid Track (RH199)* is designed as a rapid training course on the system administration of Red Hat Enterprise Linux for IT professionals with significant exposure to Linux. Students should be comfortable with running common Linux commands from the shell prompt. Students who lack this knowledge are strongly encouraged to take *Red Hat System Administration I (RH124)* instead.

### Prerequisites

You are expected to be comfortable with the following list of activities:

- Create files and directories with relative and absolute paths.
- Copy and move files.
- Create and restore archive files.
- Interpret file system permissions.
- Locate files in the system.
- Write simple Bash scripts.

# Orientation to the Classroom Environment



**Figure 0.1: Classroom environment**

In this course, the main computer system for hands-on learning activities is **workstation**. Students also use two other machines for these activities: **servera** and **serverb**. All three systems are in the `lab.example.com` DNS domain.

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

## Classroom Machines

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

The primary function of **bastion** is to act as a router between the network that connects the student machines and the classroom network. If **bastion** is down, then other student machines can access only systems on the individual student network.

Several systems in the classroom provide supporting services. Two servers, **content.example.com** and **materials.example.com**, are sources for software and lab

## Introduction

materials in hands-on activities. Information about how to use these servers is provided in the instructions for those activities. These activities are provided by the **workstation** virtual machine. Both **classroom** and **bastion** must always be running for proper use of the lab environment.

**Note**

When logging on to **servera** or **serverb**, you might see a message about activating **cockpit**. You can ignore the message.

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

[student@serverb ~]$
```

## Controlling the Virtual Machines

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 [rol.redhat.com](http://rol.redhat.com) [<http://rol.redhat.com>]. 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.

VM Name	Status	Action	Open Console
bastion	active	ACTION -	OPEN CONSOLE
classroom	active	ACTION -	OPEN CONSOLE
servera	building	ACTION -	OPEN CONSOLE
serverb	building	ACTION -	OPEN CONSOLE
workstation	active	ACTION -	OPEN CONSOLE

**Figure 0.2: An example course Lab Environment management page**

## Machine States

Virtual machine state	Description
building	The virtual machine is being created.
active	The virtual machine is running and available. If it just started, it still might be starting services.
stopped	The virtual machine is completely shut down. On starting, the virtual machine boots into the same state it was in before shutdown. The disk state is preserved.

## Classroom Actions

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

## Machine Actions

Button or action	Description
OPEN CONSOLE	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.
ACTION > Start	Start (power on) the virtual machine.
ACTION > Shutdown	Gracefully shut down the virtual machine, preserving disk contents.
ACTION > Power Off	Forcefully shut down the virtual machine, while still preserving disk contents. This is equivalent to removing the power from a physical machine.
ACTION > Reset	Forcefully shut down the virtual machine and reset 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 instructed to reset a single virtual machine node, click **ACTION > Reset** for only that specific virtual machine.

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

If you want to return the classroom environment to its original state at the start of the course, then click **DELETE** to remove the entire classroom environment. After the lab has been deleted, then 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 set allotment of computer time. To help 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 + buttons at the bottom of the course management page. Click the auto-stop + button to add another hour to the auto-stop timer. Click the auto-destroy + button to add another day to the auto-destroy timer. Auto-stop has a maximum of 11 hours, and auto-destroy has a maximum of 14 days. Be careful to keep the timers set while 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.
- A *comprehensive review lab* is used at the end of the course. It is also a gradable hands-on activity, and might cover content from the entire course. You work through a specification of what to accomplish in the activity, without receiving the specific steps to do so. Again, a solution is provided with a step-by-step walk-through that meets the specification.

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 `start`, `grade`, or `finish`. All exercises support `start` and `finish`. Only end-of-chapter labs and comprehensive review labs support `grade`.

## **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. You can take an exercise at any time, even without taking preceding exercises.

## **grade**

For gradable activities, the `grade` action directs the `lab` command to evaluate your work, and shows a list of grading criteria with a `PASS` or `FAIL` status for each. To achieve a `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 take 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 the Tab key twice.

## Chapter 1

# Access Systems and Obtaining Support

### Goal

Edit text files, log in to local and remote Linux system, and investigate problem resolution methods provided through Red Hat Support and Red Hat Insights.

### Objectives

- Create and edit text files from the command line with the vim editor.
- Configure a user account to use key-based authentication to log in to remote systems securely without a password.
- Describe and use Red Hat Customer Portal key resources to find information from Red Hat documentation and the Knowledgebase.
- Use Red Hat Insights to analyze servers for issues, remediate or resolve them, and confirm that the solution worked.

### Sections

- Edit Text Files from the Shell Prompt (and Guided Exercise)
- Configure SSH Key-based Authentication (and Guided Exercise)
- Get Help From Red Hat Customer Portal (and Guided Exercise)
- Detect and Resolve Issues with Red Hat Insights (and Quiz)

# Edit Text Files from the Shell Prompt

## Objectives

Create and edit text files from the command line with the `vim` editor.

## Edit Files with Vim

The fundamental design principle of Linux is that it supports storage of the information and configuration settings in text-based files. These files follow various structures such as lists of settings, INI-like formats, structured XML or YAML, and others. The advantage of storing files in a text-based structure is that they are easily edited with any simple text editor.

Vim is an improved version of the `vi` editor, which is distributed with Linux and UNIX systems. Vim is highly configurable and efficient for practiced users, including split-screen editing, color formatting, and highlighting for editing text.

## Benefits of the Vim Editor

When a system uses a text-only shell prompt, you should know how to use at least one text editor for editing files. You can then edit text-based configuration files from a terminal window or remote logins through the `ssh` command or the Web Console. You also do not need access to a graphical desktop to edit files on a server, and that server might not need to run a graphical desktop environment.

The key reason to learn Vim is that it is almost always installed by default on a server for editing text-based files. The *Portable Operating System Interface* or *POSIX* standard specified the `vi` editor on Linux, and many other UNIX-like operating systems largely do likewise.

Vim is also used often as the `vi` implementation on other standard operating systems or distributions. For example, macOS currently includes a lightweight installation of Vim by default. So, Vim skills that are learned for Linux might also prove useful elsewhere.

## Get Started with Vim

You can install the Vim editor in Red Hat Enterprise Linux by using either of two packages. These two packages provide different features and Vim commands for editing text-based files.

With the `vim-minimal` package, you might install the `vi` editor with core features. This lightweight installation includes only the core features and the basic `vi` command. You can open a file for editing by using the `vi` command.

```
[user@host ~]$ vi filename
```

Alternatively, you can use the `vim-enhanced` package to install the Vim editor. This package provides a more comprehensive set of features, an online help system, and a tutorial program. Use the `vim` command to start Vim in this enhanced mode.

```
[user@host ~]$ vim filename
```

The core features of the Vim editor are available in both commands.

If `vim-enhanced` is installed, then a shell alias is set so that if regular users run the `vi` command, then they automatically get the `vim` command instead. This alias does not apply to the `root` user and other users with UIDs below 200 (which system services use).

If `vim-enhanced` is installed and a regular user wants to use the `vi` command, then they might have to use the `\vi` command to override the alias temporarily. You can use `\vi --version` and `vim --version` to compare the feature sets of the two commands.

## Vim Operating Modes

The Vim editor offers various modes of operation such as *command mode*, *extended command mode*, *edit mode*, and *visual mode*. You should always check the current mode as a Vim user, because keystrokes have different effects in different modes.

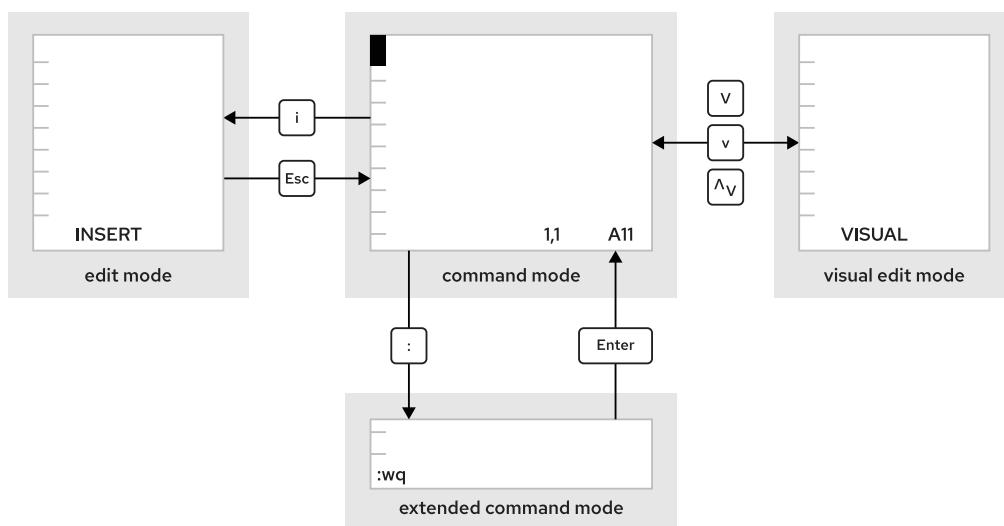


Figure 1.1: Moving between Vim modes

When you first open Vim, it starts in *command mode*, which is used for navigation, cut and paste, and other text modification. Pressing the required keystroke accesses specific editing functions.

- An `i` keystroke enters *insert mode*, where all typed text becomes file content. Pressing `Esc` returns to command mode.
- A `v` keystroke enters *visual mode*, where multiple characters might be selected for text manipulation. Use `Shift+V` for multiline and `Ctrl+V` for block selection. To exit the visual mode, use the `v`, `Shift+V`, or `Ctrl+V` keystrokes.
- The `:` keystroke begins *extended command mode* for tasks such as writing the file (to save it) and quitting the Vim editor.



### Note

If you are unsure which mode Vim is using, then press `Esc` a few times to get back into command mode. It is safe to press the `Esc` key in command mode repeatedly.

## The Minimum, Basic Vim Workflow

Vim has efficient, coordinated keystrokes for advanced editing tasks. Although considered beneficial with practice, the capabilities of Vim can overwhelm new users.

Red Hat recommends to learn the following Vim keys and commands.

- The **u** key undoes the most recent edit.
- The **x** key deletes a single character.
- The **:w** command writes (saves) the file and remains in command mode for more editing.
- The **:wq** command writes (saves) the file and quits Vim.
- The **:q!** command quits Vim, and discards all file changes since the last write.

Learning these commands helps a Vim user to accomplish any editing task.

## Rearrange Existing Text

In Vim, you can *yank* and *put* (copy and paste), by using the **y** and **p** command characters. Position the cursor on the first character to select, and then enter visual mode. Use the arrow keys to expand the visual selection. When ready, press **y** to *yank* the selection into memory. Position the cursor at the new location, and then press **p** to *put* the selection at the cursor.

## Visual Mode in Vim

Visual mode is useful to highlight and manipulate text in different lines and columns. You can enter visual modes on Vim by using the following key combinations.

- Character mode : **v**
- Line mode : **Shift+v**
- Block mode : **Ctrl+v**

Character mode highlights sentences in a block of text. The word **VISUAL** appears at the bottom of the screen. Press **v** to enter visual character mode. **Shift+v** enters line mode. **VISUAL LINE** appears at the bottom of the screen.

Visual block mode is perfect for manipulating data files. Press the **Ctrl+v** keystroke to enter the visual block from the cursor. **VISUAL BLOCK** appears at the bottom of the screen. Use the arrow keys to highlight the section to change.



### Note

Become competent with the basic Vim workflow first. It takes practice to understand the many Vim capabilities. Get comfortable with the basics, then expand your Vim vocabulary by learning additional Vim keystrokes.

The exercise for this section uses the **vimtutor** command. This tutorial, from the **vim-enhanced**, is an excellent way to learn the core Vim functions.

## Vim Configuration Files

The **/etc/vimrc** and **~/.vimrc** configuration files alter the behavior of the **vim** editor for the entire system or a specific user respectively. Within these configuration files, you can specify behavior such as the default tab spacing, syntax highlighting, color schemes, and more. Modifying the behavior of the **vim** editor is particularly useful when working with languages such as YAML, which have strict syntax requirements. Consider the following **~/.vimrc** file, which sets the default tab stop (denoted by the **ts** characters) to two spaces while editing YAML files. The file also includes the **set number** parameter to display line numbers while editing all files.

```
[user@host ~]$ cat ~/.vimrc
autocmd FileType yaml setlocal ts=2
set number
```

A complete list of `vimrc` configuration options is available in the references.



## References

`vim(1)` man page

The `:help` command in `vim` (if the `vim-enhanced` package is installed).

### Vim Reference Manual: Vim Options

<https://vimhelp.org/options.txt.html#options.txt>

## ► Guided Exercise

# Edit Text Files from the Shell Prompt

In this exercise, you use the `vimtutor` command to practice basic editing techniques in the `vim` editor.

## Outcomes

- Edit files with Vim.
- Gain competency in Vim by using the `vimtutor` command.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start edit-editfile
```

## Instructions

- 1. Use the `ssh` command to log in to the `servera` machine as the `student` user.

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

- 2. Run the `vimtutor` command. Read the Welcome screen and perform *Lesson 1.1*.

In the presentation, keyboard arrow keys help to navigate the window. Initially, when the `vi` editor was developed, users could not rely on having arrow keys or working keyboard mappings for arrow keys to move the cursor. Therefore, the `vi` editor was initially designed to move the cursor by using commands with standard character keys, such as the conveniently grouped `h`, `j`, `k`, and `l`.

Here is a way to remember them:

**hang back, jump down, kick up, leap forward.**

```
[student@servera ~]$ vimtutor
```

- 3. In the `vimtutor` window, perform *Lesson 1.2*.

This lesson teaches how to quit without keeping unwanted changes. All changes are lost. Sometimes it is preferable to lose changes than to leave a critical file in an incorrect state.

- 4. In the `vimtutor` window, perform *Lesson 1.3*.

Vim has fast, efficient keystrokes to delete an exact number of words, lines, sentences, or paragraphs. Any editing is possible with the `x` key for single character deletion.

- **5.** In the `vimtutor` window, perform *Lesson 1.4*.

For most editing tasks, the first key that is pressed is the `i` key.

- **6.** In the `vimtutor` window, perform *Lesson 1.5*.

The previous lecture taught only the `i` (insert) command to enter edit mode. This lesson demonstrates other available keystrokes to change the cursor placement when entered into insert mode. In insert mode, all typed text changes the file content.

- **7.** In the `vimtutor` window, perform *Lesson 1.6*.

Type `:wq` to save the file and quit the editor.

- **8.** In the `vimtutor` window, read the *Lesson 1 Summary*.

The `vimtutor` command includes six more multistep lessons. These lessons are not assigned as part of this course, but feel free to explore them to learn more.

- **9.** Return to the workstation system as the student user.

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

## 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 edit-editfile
```

This concludes the section.

# Configure SSH Key-based Authentication

## Objectives

Configure a user account to use key-based authentication to log in to remote systems securely without a password.

## SSH Key-based Authentication

You can configure your account for passwordless access to SSH servers that have enabled key-based authentication, which is based on public key encryption (PKI).

To prepare your account, generate a cryptographically-related pair of key files. One key is private and held only by you, while the second is your related public key that is not secret. The private key acts as your authentication credential and it must be stored securely. The public key is copied to your account on servers that you will remotely access, and verifies your use of your private key.

When you log in to your account on a remote server, the SSH service uses your public key to cryptographically verify supplied with the SSH client request. If the verification succeeds, your request is trusted and you are allowed access without giving a password. Passwords can be easily learned or stolen, but securely stored private keys are harder to compromise.

## SSH Keys Generation

Use the `ssh-keygen` command to create a key pair. By default, the `ssh-keygen` command saves your private and public keys in the `~/.ssh/id_rsa` and `~/.ssh/id_rsa.pub` files, but you can specify a different name.

```
[user@host ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): Enter
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vxutUNPi03QDCyvkYm1 user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|           |
|   .     . |
| o o     o |
| . = o   o . |
| o + = S E . |
| ..o o + * + |
| .% 0 . + B . |
|=*o0 . . + * |
|++.       . +. |
+---[SHA256]-----+
```

You can choose to provide a passphrase to `ssh-keygen`, which is used to encrypt your private key. Using a passphrase is recommended, so that your private key cannot be used by someone who gains access to it. If you set a passphrase, then you must enter the passphrase each time you use the private key. The passphrase is used locally, to decrypt your private key before use, unlike your password, which must be sent in clear text across the network for use.

You can use the `ssh-agent` key manager locally, which caches your passphrase upon first use in a login session, and then provides the passphrase for all subsequent private key use in the same login session. The `ssh-agent` command is discussed later in this section.

In the following example, a passphrase-protected private key is created with the public key.

```
[user@host ~]$ ssh-keygen -f .ssh/key-with-pass
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): your_passphrase
Enter same passphrase again: your_passphrase
Your identification has been saved in .ssh/key-with-pass.
Your public key has been saved in .ssh/key-with-pass.pub.
The key fingerprint is:
SHA256:w3GGB7EyHUrY4a0cNPKmhNKS7dl1YsMVLvFZJ77VxAo user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|       . + =.0 ... |
|       = B XEo o. |
|   . o O X =.... |
| == = = B = o.    |
|= + * * S .      |
|.+= o + .        |
| + .             |
|                 |
|                 |
+---[SHA256]-----+
```

The `ssh-keygen` command `-f` option specifies the files in which to save the keys. In the preceding example, the `ssh-keygen` command saved the key pair in the `/home/user/.ssh/key-with-pass` and `/home/user/.ssh/key-with-pass.pub` files.



### Warning

During new `ssh-keygen` command use, if you specify the name of an existing pair of key files, including the default `id_rsa` pair, you will overwrite that existing key pair, which can only be restored if you have a backup for those files. Overwriting a key pair will lose the original private key which is required to access accounts that you have configured with the corresponding public on remote servers.

If you are unable to restore your local private key, you will lose access to remote servers until you distribute your new public key to replace the previous public key on each server. Always create backups of your keys in the event that they are overwritten or lost.

Generated SSH keys are stored, by default, in the `.ssh` subdirectory of your home directory. To function properly, the private must be readable only by the owner, which is a 600 permission mode. Public keys can be read by anyone, which is commonly set as a 644 permission mode.

## Share the Public Key

To configure your remote account for access, copy your public key to the remote system. The `ssh-copy-id` command copies the public key of the SSH key pair to the remote system. You can specify a specific public key with the `ssh-copy-id` command, or use the default `~/.ssh/id_rsa.pub` file.

```
[user@host ~]$ ssh-copy-id -i .ssh/key-with-pass.pub user@remotehost
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/user/.ssh/
id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
user@remotehost's password: redhat
Number of key(s) added: 1

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

Test the remote access, after placing the public key, with the corresponding private key. If the configuration is correct, you will gain access to your account on the remote system without being asked for your account password. If you do not specify a private key file, then the `ssh` command uses the default `~/.ssh/id_rsa` file if it exists.



### Important

If you configured a passphrase to protect your private key, the passphrase will be requested by SSH at first use. However, if the key authentication succeeds, you will not be asked for your *account* password.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost
Enter passphrase for key '.ssh/key-with-pass': your_passphrase
...output omitted...
[user@remotehost ~]$
```

## Non-interactive Authentication with the Key Manager

If you encrypt your private key with a passphrase, then you must enter the passphrase each time you use the private key for authentication. However, you can configure the `ssh-agent` key manager to cache passphrases. Then, each time you use SSH, the `ssh-agent` key manager provides the passphrase for you. Using a key manager convenient and can improve security by providing fewer opportunities for other people to observe your passphrase.

The `ssh-agent` key manager can be configured to start automatically when you log in. The GNOME graphical desktop environment can automatically start and configure the `ssh-agent` key manager. If you log in to a text environment, you must start the `ssh-agent` program manually for each session. Start the `ssh-agent` program with the following command:

```
[user@host ~]$ eval $(ssh-agent)
Agent pid 10155
```

When you manually start the `ssh-agent` command, it runs additional shell commands to set environment variables that are needed for use with the `ssh-add` command. You can manually load your private key passphrase to the the key manager using the `ssh-add` command.

The following example `ssh-add` commands add the private keys from the default `~/.ssh/id_rsa` file and then from a `~/.ssh/key-with-pass` file.

```
[user@host ~]$ ssh-add  
Identity added: /home/user/.ssh/id_rsa (user@host.lab.example.com)  
[user@host ~]$ ssh-add .ssh/key-with-pass  
Enter passphrase for .ssh/key-with-pass: your_passphrase  
Identity added: .ssh/key-with-pass (user@host.lab.example.com)
```

The following `ssh` command uses the default private key file to access your account on a remote SSH server.

```
[user@host ~]$ ssh user@remotehost  
Last login: Mon Mar 14 06:51:36 2022 from host.example.com  
[user@remotehost ~]$
```

The following `ssh` command uses the `~/.ssh/key-with-pass` private key access your account on the remote server. The private key in this example was previously decrypted and added to the `ssh-agent` key manager, therefore the `ssh` command does not prompt you for the passphrase to decrypt the private key.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost  
Last login: Mon Mar 14 06:58:43 2022 from host.example.com  
[user@remotehost ~]$
```

When you log out of a session that used an `ssh-agent` key manager, all cached passphrases are cleared from memory.

## Basic SSH Connection Troubleshooting

SSH can appear complex when remote access using key pair authentication is not succeeding. The `ssh` command provides three verbosity levels with the `-v`, `-vv`, and `-vvv` options, that provide increasingly greater amounts of debugging information during `ssh` command use.

The next example demonstrates the information provided when using the lowest verbosity option:

```
[user@host ~]$ ssh -v user@remotehost  
OpenSSH_8.7p1, OpenSSL 3.0.1 14 Dec 2021 ①  
debug1: Reading configuration data /etc/ssh/ssh_config ②  
debug1: Reading configuration data /etc/ssh/ssh_config.d/01-training.conf  
debug1: /etc/ssh/ssh_config.d/01-training.conf line 1: Applying options for *  
debug1: Reading configuration data /etc/ssh/ssh_config.d/50-redhat.conf  
...output omitted...  
debug1: Connecting to remotehost [192.168.1.10] port 22. ③  
debug1: Connection established.  
...output omitted...  
debug1: Authenticating to remotehost:22 as 'user' ④  
...output omitted...
```

```

debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-
mic,password ⑤
...output omitted...
debug1: Next authentication method: publickey ⑥
debug1: Offering public key: /home/user/.ssh/id_rsa RSA
  SHA256:hDVJjD7xrUjXGZVRJQixxFV6NF/ssMjS6AuQ1+VqUc4 ⑦
debug1: Server accepts key: /home/user/.ssh/id_rsa RSA
  SHA256:hDVJjD7xrUjXGZVRJQixxFV6NF/ssMjS6AuQ1+VqUc4 ⑧
Authenticated to remotehost ([192.168.1.10]:22) using "publickey".
...output omitted...
[user@remotehost ~]$
```

- ① OpenSSH and OpenSSL versions.
- ② OpenSSH configuration files.
- ③ Connection to the remote host.
- ④ Authentication methods that the remote host allows.
- ⑤ Trying to authenticate the user on the remote host.
- ⑥ Trying to authenticate the user by using the SSH key.
- ⑦ Using the /home/user/.ssh/id\_rsa key file to authenticate.
- ⑧ The remote hosts accepts the SSH key.

If an attempted authentication method fails, a remote SSH server will *fail back* to other allowed authentication methods until all the available methods are tried. The next example demonstrates a remote access with an SSH key that fails, but then the SSH server offers password authentication that succeeds.

```

[user@host ~]$ ssh -v user@remotehost
...output omitted...
debug1: Next authentication method: publickey
debug1: Offering public key: /home/user/.ssh/id_rsa RSA
  SHA256:bsB6l5R184zvxNlrcRMmYd32oBkU1LgQj09dUBZ+Z/k
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-
mic,password
...output omitted...
debug1: Next authentication method: password
user@remotehost's password: password
Authenticated to remotehost ([172.25.250.10]:22) using "password".
...output omitted...
[user@remotehost ~]$
```

## SSH Client Configuration

You can create the `~/.ssh/config` file to preconfigure SSH connections. Within the configuration file, you can specify connection parameters such as users, keys, and ports for specific hosts. This file eliminates the need to manually specify command parameters each time that you connect to a host. Consider the following `~/.ssh/config` file, which preconfigures two host connections with different users and keys:

```
[user@host ~]$ cat ~/.ssh/config
host servera
  HostName          servera.example.com
  User              usera
  IdentityFile     ~/.ssh/id_rsa_servera

host serverb
  HostName          serverb.example.com
  User              userb
  IdentityFile     ~/.ssh/id_rsa_serverb
```

The `~/.ssh/config` file is also useful for configuring SSH jump hosts. An SSH jump host is a server that acts as a proxy for SSH connections to other, usually internal, hosts. Consider a scenario where a host called `external` is accessible via the internet, but a host called `internal` is only internally accessible. Use the `ProxyHost` parameter within the `~/.ssh/config` file to connect to the `internal` host via the `external` host:

```
[user@host ~]$ cat ~/.ssh/config
host internal
  HostName          internal.example.com
  ProxyHost         external

host external
  HostName          external.example.com
```



## References

`ssh-keygen(1)`, `ssh-copy-id(1)`, `ssh-agent(1)`, and `ssh-add(1)` man pages

## ► Guided Exercise

# Configure SSH Key-based Authentication

In this exercise, you configure a user to use key-based authentication for SSH.

## Outcomes

- Generate an SSH key pair without passphrase protection.
- Generate an SSH key pair with passphrase protection.
- Authenticate with both passphrase-less and passphrase-protected SSH keys.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start ssh-configure
```

## Instructions

- 1. Log in to the serverb machine as the student user.

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

- 2. Switch to the operator1 user on the serverb machine. Use redhat as the password.

```
[student@serverb ~]$ su - operator1  
Password: redhat  
[operator1@serverb ~]$
```

- 3. Generate a set of SSH keys. Do not enter a passphrase.

```
[operator1@serverb ~]$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/operator1/.ssh/id_rsa): Enter  
Created directory '/home/operator1/.ssh'.  
Enter passphrase (empty for no passphrase): Enter  
Enter same passphrase again: Enter  
Your identification has been saved in /home/operator1/.ssh/id_rsa.  
Your public key has been saved in /home/operator1/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:JainiQdnRosC+xXh operator1@serverb.lab.example.com  
The key's randomart image is:
```

```
+-- [RSA 3072] --+
|E+*+ooo .
|= o.o o .
|o.. = . . o
|+. + * . o .
|+ = X . S +
| + @ + = .
|. + = o
|.o . . .
|o o..
+-- [SHA256] --+
```

- ▶ 4. Send the public key of the SSH key pair to the `operator1` user on the `servera` machine, with `redhat` as the password.

```
[operator1@serverb ~]$ ssh-copy-id operator1@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/operator1/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be established.
ED25519 key fingerprint is SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KWOTK80.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
operator1@servera's password: redhat

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
```

- ▶ 5. Execute the `hostname` command on the `servera` machine remotely by using the `ssh` command without accessing the remote interactive shell.

```
[operator1@serverb ~]$ ssh operator1@servera hostname
servera.lab.example.com
```

The preceding `ssh` command does not prompt you for a password because it uses the passphrase-less private key against the exported public key to authenticate as the `operator1` user on the `servera` machine. This approach is not secure because anyone who has access to the private key file can log in to the `servera` machine as the `operator1` user. The secure alternative is to protect the private key with a passphrase, which is a following step.

- ▶ 6. Generate another set of SSH keys with the default name and without a passphrase, overwriting the previously generated SSH key files. Try to connect to the `servera` machine by using the new SSH keys. The `ssh` command asks for a password, because it cannot authenticate with the SSH key. Run again the `ssh` command with the `-v` (verbose) option to verify it.

Send the new public key of the SSH key pair to the `operator1` user on the `servera` machine, to replace the previous public key. Use `redhat` as the password for the

operator1 user on the servera machine. Execute the hostname command on the servera machine remotely by using the ssh command without accessing the remote interactive shell to verify that it works again.

- 6.1. Again generate another set of SSH keys with the default name and without a passphrase, overwriting the previously generated SSH key files.

```
[operator1@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operator1/.ssh/id_rsa): Enter
/home/operator1/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/operator1/.ssh/id_rsa
Your public key has been saved in /home/operator1/.ssh/id_rsa.pub
...output omitted...
```

- 6.2. Try to connect to the servera machine by using the new SSH keys. The ssh command asks for a password, because it cannot authenticate with the SSH key. Press Ctrl+c to exit from the ssh command when it prompts for a password. Run again the ssh command with the -v (verbose) option to verify it. Press again Ctrl+c to exit from the ssh command when it prompts for a password.

```
[operator1@serverb ~]$ ssh operator1@servera hostname
operator1@servera's password: ^C
[operator1@serverb ~]$ ssh -v operator1@servera hostname
OpenSSH_8.7p1, OpenSSL 3.0.1 14 Dec 2021
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Reading configuration data /etc/ssh/ssh_config.d/01-training.conf
...output omitted...
debug1: Next authentication method: publickey
debug1: Offering public key: /home/operator1/.ssh/id_rsa RSA
SHA256:ad597Zf64xckV26xht8bjQbzqSPuOXQPXksGEWVsP80
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password
debug1: Trying private key: /home/operator1/.ssh/id_dsa
debug1: Trying private key: /home/operator1/.ssh/id_ecdsa
debug1: Trying private key: /home/operator1/.ssh/id_ecdsa_sk
debug1: Trying private key: /home/operator1/.ssh/id_ed25519
debug1: Trying private key: /home/operator1/.ssh/id_ed25519_sk
debug1: Trying private key: /home/operator1/.ssh/id_xmss
debug1: Next authentication method: password
operator1@servera's password: ^C
```

- 6.3. Send the new public key of the SSH key pair to the operator1 user on the servera machine, to replace the previous public key. Use redhat as the password for the operator1 user on the servera machine. Execute the hostname command on the servera machine remotely by using the ssh command without accessing the remote interactive shell to verify that it works again.

```
[operator1@serverb ~]$ ssh-copy-id operator1@servera
...output omitted...
operator1@servera's password: redhat
```

```
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
[operator1@serverb ~]$ ssh operator1@servera hostname
servera.lab.example.com
```

- 7. Generate another set of SSH keys with passphrase-protection. Save the key as /home/operator1/.ssh/key2. Use redhatpass as the passphrase of the private key.

```
[operator1@serverb ~]$ ssh-keygen -f .ssh/key2
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): redhatpass
Enter same passphrase again: redhatpass
Your identification has been saved in .ssh/key2.
Your public key has been saved in .ssh/key2.pub.
The key fingerprint is:
SHA256:0OctCjfPm5QrbPBgqb operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
|O=X*          |
|OB=.          |
|E*o.          |
|Booo .        |
|..= . o S    |
|+.o  o        |
|+.oo+ o       |
|+o.0.+        |
|+. . =o.      |
+---[SHA256]----+
```

- 8. Send the public key of the passphrase-protected key pair to the operator1 user on the servera machine. The command does not prompt you for a password because it uses the public key of the passphrase-less private key that you exported to the servera machine in the preceding step.

```
[operator1@serverb ~]$ ssh-copy-id -i .ssh/key2.pub operator1@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/key2.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
```

- ▶ 9. Execute the `hostname` command on the `servera` machine remotely by using the `ssh` command. Use the `/home/operator1/.ssh/key2` key as the identity file. Specify `redhatpass` as the passphrase, which you set for the private key in the preceding step.

The command prompts you for the passphrase that you used to protect the private key of the SSH key pair. If an attacker gains access to the private key, then the attacker cannot use it to access other systems because a passphrase protects the private key itself. The `ssh` command uses a different passphrase from the `operator1` user on the `servera` machine, and so users must know both.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname
Enter passphrase for key '.ssh/key2': redhatpass
servera.lab.example.com
```

Use the `ssh-agent` program, as in the following step, to avoid interactively typing the passphrase while logging in with SSH. Using the `ssh-agent` program is both more convenient and more secure when the administrators log in to remote systems regularly.

- ▶ 10. Run the `ssh-agent` program in your Bash shell and add the passphrase-protected private key (`/home/operator1/.ssh/key2`) of the SSH key pair to the shell session.

The command starts the `ssh-agent` program and configures the shell session to use it. Then, you use the `ssh-add` command to provide the unlocked private key to the `ssh-agent` program.

```
[operator1@serverb ~]$ eval $(ssh-agent)
Agent pid 1729
[operator1@serverb ~]$ ssh-add .ssh/key2
Enter passphrase for .ssh/key2: redhatpass
Identity added: .ssh/key2 (operator1@serverb.lab.example.com)
```

- ▶ 11. Execute the `hostname` command on the `servera` machine remotely without accessing a remote interactive shell. Use the `/home/operator1/.ssh/key2` key as the identity file.

The command does not prompt you to enter the passphrase interactively.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname
servera.lab.example.com
```

- ▶ 12. Open another terminal on the `workstation` machine and log in to the `serverb` machine as the `student` user.

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

- ▶ 13. On the `serverb` machine, switch to the `operator1` user and remotely log in to the `servera` machine. Use the `/home/operator1/.ssh/key2` key as the identity file to authenticate using the SSH keys.

- 13.1. Use the `su` command to switch to the `operator1` user. Use `redhat` as the password for the `operator1` user.

```
[student@serverb ~]$ su - operator1  
Password: redhat  
[operator1@serverb ~]$
```

13.2. Log in to the **servera** machine as the **operator1** user.

The command prompts you to enter the passphrase interactively because you do not invoke the SSH connection from the same shell where you started the ssh-agent program.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera  
Enter passphrase for key '.ssh/key2': redhatpass  
...output omitted...  
[operator1@servera ~]$
```

► 14. Exit and close all extra terminals and return to the **workstation** machine.

14.1. Exit and close extra terminal windows. The **exit** commands leave the **operator1** user's shell; terminate the shell session where **ssh-agent** is active; and return to the **student** user's shell on the **serverb** machine.

```
[operator1@servera ~]$ exit  
logout  
Connection to servera closed.  
[operator1@serverb ~]$
```

14.2. Return to the **workstation** system as the **student** user.

```
[operator1@serverb ~]$ exit  
logout  
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

## 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 ssh-configure
```

This concludes the section.

# Get Help From Red Hat Customer Portal

---

## Objectives

Describe and use Red Hat Customer Portal key resources to find information from Red Hat documentation and the Knowledgebase.

## Resources on the Red Hat Customer Portal

The Red Hat Customer Portal at <https://access.redhat.com> gives customers access to documentation, downloads, tools, and technical expertise. The Knowledgebase allows customers to search for solutions, FAQs, and articles. The following list shows some functions of the Red Hat Customer Portal:

- Access official product documentation, solutions, and FAQs.
- Submit and manage support cases.
- Manage software subscriptions and entitlements.
- Obtain software downloads, updates, and evaluations.
- Access a catalog of security advisories for Red Hat products.
- Access an integrated search engine for Red Hat resources.
- Access white papers, information sheets, and multimedia presentations.
- Participate in community discussions.

Parts of the site are public-accessible to everyone, and other areas require an active subscription. Visit <https://access.redhat.com/help/> for help with accessing the Red Hat Customer Portal.

## Tour of the Red Hat Customer Portal

Access the Red Hat Customer Portal by visiting <https://access.redhat.com/>. This section introduces the Red Hat Customer Portal tour at <https://access.redhat.com/start>.

With the tour, you can discover portal features and maximize the benefits of your Red Hat subscription. After you log in to the Red Hat Customer Portal, click the **Tour the Customer Portal** button.

The **WELCOME TO THE RED HAT CUSTOMER PORTAL** window appears. Click the **Let's go** button to start the tour.

## The Top Navigation Bar

The first menus on the tour, on the top navigation bar, are Subscriptions, Downloads, Containers, and Support Cases.

The **Subscriptions** menu opens a new page to manage your registered systems, subscriptions, and entitlements. This page lists applicable errata information. You can create activation keys for registering systems and ensuring correct entitlements. The Organization Administrator for your account might restrict your access to this page.

The **Downloads** menu opens a new page to access your product downloads and request evaluation for the products with no entitlements.

## Chapter 1 | Access Systems and Obtaining Support

The **Support Cases** menu opens a new page to create, track, and manage your support cases through the Case Management system, if authorized by your organization.

With the **User Menu** menu, manage your account, any accounts for which you are an Organization Administrator, your profile, and email notification options.

The globe icon opens the **Language** menu to specify your language preferences for the Red Hat Customer Portal.

## Navigate the Red Hat Customer Portal Menus

Underneath the top navigation bar on the main page are menus to navigate to major categories of resources on the site.

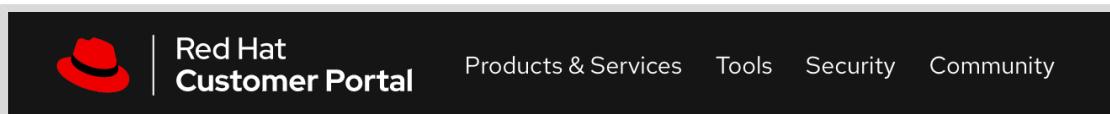


Figure 1.2: Red Hat Customer Portal Menus

The **Products & Services** menu gives access to the Product Hubs, for access to product-specific evaluations, getting started guides, and other product support information. You can also access documentation for Red Hat products, a knowledge base of support articles, and support policies, and can contact Red Hat Support. You can access services that Red Hat provides, such as Consulting, Technical Account Management, and Training and Certifications.

The **Tools** menu provides tools to help you to succeed with Red Hat products. The tools help to troubleshoot a product issue, and provide packages and errata information. The **Customer Portal Labs** section has a collection of web-based applications and tools to help you to improve performance, diagnose issues, identify security problems, and optimize your configurations. The **Red Hat Insights** section helps to analyze platforms and applications to predict risk, take recommended actions, and track costs to manage hybrid cloud environments. Insights alert administrators before an outage, or about a security event or overspending.

The **Security** menu provides access to the Red Hat Product Security Center for security updates and prevents environments from exposure to security vulnerabilities. This section provides information about high-profile security issues, with access to the security advisories, the Red Hat *Common Vulnerabilities and Exposures (CVE)* database, the security labs, the Red Hat security blog, security measurement, severity ratings, backporting policies, and product signing *GNU Privacy Guard (GPG)* keys.

The **Community** menu gives access to the **Customer Portal Community** section for discussions and private groups. This section is a place where Red Hat experts, customers, and partners communicate and collaborate. This section contains discussion forums, blogs, and information about upcoming events.



### Note

Red Hat recommends viewing the complete tour at Getting Started with Red Hat [<https://access.redhat.com/start>], including the sections on the **How to Personalize Your Customer Portal experience** menu, the **Explore the Benefits of Your Red Hat subscription** menu, and the **How to Engage Red Hat Support** menu. An active subscription is required to access these subscription resources.

## Contact Red Hat Customer Support

The Red Hat Customer Portal provides access to technical support for customers with an active subscription. You can contact support by opening a support case or a chat session, or by phone. For detailed information, visit the [https://access.redhat.com/support/policy/support\\_process](https://access.redhat.com/support/policy/support_process) address.

### Prepare a Support Case

Before contacting Red Hat support, it is important to gather relevant information for the report.

*Define the problem.* State the problem and its symptoms specifically. Provide detailed steps to reproduce the problem.

*Gather background information.* Which product and version are affected? Be ready to provide relevant diagnostic information. This information might include the output of the `sos report` command. For kernel problems, the information might consist of the system's kdump crash dump or a digital photo of the kernel backtrace that is displayed on the monitor of a crashed system.

*Determine the severity level.* Red Hat uses four severity levels to classify issues. *Urgent* and *High* severity problem reports must be followed by a phone call to the relevant local support center (see <https://access.redhat.com/support/contact/technicalSupport>).

Severity	Description
Urgent (Severity 1)	A problem that severely impacts your use of the software in a production environment. This severity includes loss of production data or malfunctioning production systems. The situation halts your business operations, and no procedural workaround exists.
High (Severity 2)	A problem where the software is functioning but its use in a production environment is severely reduced. The situation is causing a high impact on your business operations, and no procedural workaround exists.
Medium (Severity 3)	A problem that involves partial, non-critical loss of use of the software in a production environment or development environment. The problem involves a medium to low impact on your business for production environments. The business continues to function by using a procedural workaround. For development environments, the situation is causing problems with migrating your project into production.
Low (Severity 4)	A general usage question, reporting of a documentation error, or recommendation for a future product enhancement or modification. The problem involves low to no impact on your business or the performance or functioning of your system. The problem involves medium to low impact on your business for development environments, but your business continues to function by using a procedural workaround.

### The `sos Report` Utility

The `sos report` is generally the starting point for Red Hat technical support to investigate the reported problem. This utility provides a standardized way to collect diagnostic information that Red Hat technical support needs for investigating the reported issues. The `sos report` command collects various debugging information from one or more systems and provides an option to remove sensitive data. This report is attached to the Red Hat support case. The `sos`

`collect` command runs and collects individual `sos` reports from a specified set of nodes. The `sos clean` command obfuscates potentially sensitive information such as usernames, hostnames, IP or MAC addresses, or other user-specified data.

The following list contains information that can be collected in a report:

- The running kernel version
- Loaded kernel modules
- System and service configuration files
- Diagnostic command output
- A list of installed packages

## Generate the `sos` Report

Red Hat Enterprise Linux installs the `sos` report utility with the `sos` package:

```
[root@host ~]# dnf install sos
...output omitted...
Complete!
```

Generating the `sos` report requires root privileges. Run the `sos report` command to generate the report.

```
[root@host ~]# sos report
...output omitted...
Press ENTER to continue, or CTRL-C to quit.

Optionally, please enter the case id that you are generating this report for []:
...output omitted...
Your sosreport has been generated and saved in:
/var/tmp/sosreport-host-2022-03-29-wixbhpz.tar.xz
..output omitted...
Please send this file to your support representative.
```

When you provide any support case ID in the previous command, the report attaches directly to the previously created support case. You can also use the `sos report` command `--utility` option to send the report to technical support.

Verify that the `sos report` command created the archive file at the previous location.

```
[root@host ~]# ls -l /var/tmp/
total 9388
-rw----- 1 root root 9605952 Mar 29 02:09 sosreport-host-2022-03-29-
wixbhpz.tar.xz
-rw-r--r-- 1 root root      65 Mar 29 02:09 sosreport-host-2022-03-29-
wixbhpz.tar.xz.sha256
...output omitted...
```

The `sos clean` command obfuscates personal information from the report.

```
[root@host ~]# sos clean /var/tmp/sosreport-host-2022-03-29-wixbhpz.tar.xz*
...output omitted...
Press ENTER to continue, or CTRL-C to quit.
...output omitted...
The obfuscated archive is available at
/var/tmp/sosreport-host0-2022-03-29-wixbhpz-obfuscated.tar.xz
...output omitted...
Please send the obfuscated archive to your support representative and keep the
mapping file private
```

## Send the sos Report to Red Hat Technical Support

Select one of these methods to send an sos report to Red Hat Technical Support.

- Send the sos report by using the `sos report` command `--upload` option.
- Send the sos report to the Red Hat Customer Portal by attaching it with the support case.

## Join the Red Hat Developer Program

The Red Hat Developer Program at <https://developers.redhat.com> provides subscription entitlements to Red Hat software for development purposes, documentation, and premium books from experts on microservices, serverless computing, Kubernetes, and Linux. Blog links to information about upcoming events and training and other helpful resources are also available.

Visit <https://developers.redhat.com/> for more information.



### References

`sosreport(1)` man page

#### Contacting Red Hat Technical Support

[https://access.redhat.com/support/policy/support\\_process/](https://access.redhat.com/support/policy/support_process/)

#### Help - Red Hat Customer Portal

<https://access.redhat.com/help/>

For further information, refer to *Generating an SOS Report for Technical Support* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/getting\\_the\\_most\\_from\\_your\\_support\\_experience/generating-an-sos-report-for-technical-support\\_getting-the-most-from-your-support-experience](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/getting_the_most_from_your_support_experience/generating-an-sos-report-for-technical-support_getting-the-most-from-your-support-experience)

## ► Guided Exercise

# Get Help From Red Hat Customer Portal

In this exercise, you generate a diagnostics report by using the web console.

### Outcomes

- Generate a diagnostics report by using the web console and submit it to the Red Hat Customer Portal as part of a support case.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start support-portal
```

### Instructions

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

```
[student@workstation ~]$ ssh student@servera
Warning: Permanently added 'servera' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket
...output omitted...
[student@servera ~]$
```

- 2. Start the `cockpit` service.

```
[student@servera ~]$ systemctl start cockpit.socket
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'cockpit.socket'.
Authenticating as: Student User (student)
Password: student
==== AUTHENTICATION COMPLETE ====
[student@servera ~]$
```

- 3. Verify the status of the `cockpit` service.

```
[student@servera ~]$ systemctl status cockpit.socket
● cockpit.socket - Cockpit Web Service Socket
   Loaded: loaded (/usr/lib/systemd/system/cockpit.socket; disabled; vendor
   preset: disabled)
     Active: active (listening) since Mon 2022-03-28 01:41:13 EDT; 1min 27s ago
       Until: Mon 2022-03-28 01:41:13 EDT; 1min 27s ago
     Triggers: • cockpit.service
```

```
Docs: man:cockpit-ws(8)
Listen: [::]:9090 (Stream)
...output omitted...
Mar 28 01:41:13 servera.lab.example.com systemd[1]: Starting Cockpit Web Service
Socket...
Mar 28 01:41:13 servera.lab.example.com systemd[1]: Listening on Cockpit Web
Service Socket.
```

- 4. Return to the **workstation** machine as the **student** user.

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

- 5. On the **workstation** machine, open the Firefox web browser and log in to the web console interface that is running at the `servera.lab.example.com` address. Log in as the **root** user with **redhat** as the password.
- 5.1. Open the Firefox web browser and go to the `https://servera.lab.example.com:9090` address.
  - 5.2. When prompted, accept the self-signed certificate by adding it as an exception.
  - 5.3. Log in as the **root** user with **redhat** as the password. You are now logged in as a privileged user, which is necessary to create a diagnostic report.
  - 5.4. Click the **Diagnostic Reports** menu in the left navigation pane. Click the **Create Report** button. The report takes a few minutes to create.
- 6. When the report is ready, click the **Download report** button to save the file.
- 6.1. Click the **Download report** button, followed by the **Save File** button.
  - 6.2. Log out from the web console session and close the Firefox web browser.

## 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 support-portal
```

This concludes the section.

# Detect and Resolve Issues with Red Hat Insights

## Objectives

Use Red Hat Insights to analyze servers for issues, remediate or resolve them, and confirm that the solution worked.

## Introduction to Red Hat Insights

Red Hat Insights is a predictive analytics tool to help you identify and remediate threats to security, performance, availability, and stability on systems in your infrastructure that run Red Hat products. Red Hat delivers Red Hat Insights as a Software-as-a-Service (SaaS) product, so that you can deploy and scale it quickly with no additional infrastructure requirements. In addition, you can take advantage of the latest recommendations and updates from Red Hat that apply to your deployed systems.

Red Hat regularly updates the knowledge base, based on common support risks, security vulnerabilities, known-bad configurations, and other issues that Red Hat identifies. Red Hat validates and verifies the actions to mitigate or remediate these issues. With this support, you can proactively identify, prioritize, and resolve issues before they become a larger problem.

For each detected issue, Red Hat Insights provides risk estimates and recommendations on how to mitigate or remediate the problem. These recommendations might suggest materials such as Ansible Playbooks or provide step-by-step instructions to help you to resolve the issue.

Red Hat Insights tailors recommendations to each system that you register to the service. To start using Red Hat Insights, install the agent in each client system to collect metadata about the runtime configuration of the system. This data is a subset of what you might provide to Red Hat Support by using the `sosreport` command to resolve a support ticket.

You can limit or obfuscate the data that your client systems send. By limiting the data, you might block some of the analytic rules from operating, depending on what you limit.

After you register a server and it completes the initial system metadata synchronization, you should be able to see your server and any recommendations for it in the Insights console in the Red Hat Cloud Portal.

Red Hat Insights currently provides predictive analytics and recommendations for these Red Hat products:

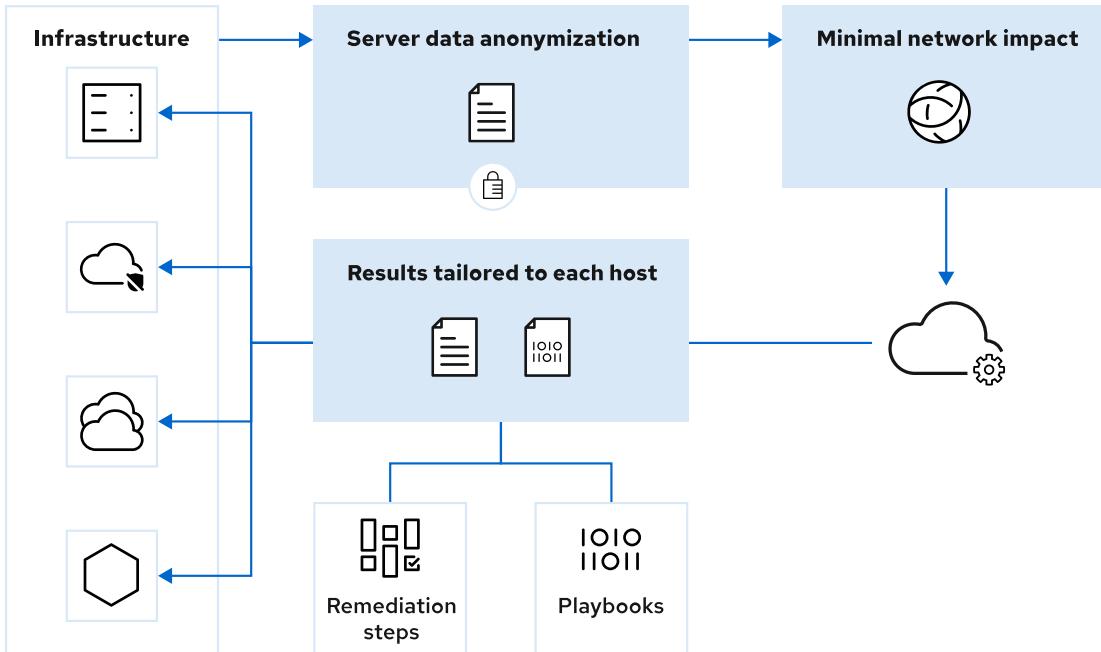
- Red Hat Enterprise Linux 6.4 and later
- Red Hat Virtualization
- Red Hat Satellite 6 and later
- Red Hat OpenShift Container Platform
- Red Hat OpenStack Platform 7 and later
- Red Hat Ansible Automation Platform

## Red Hat Insights Architecture Description

When you register a system with Red Hat Insights, it immediately sends metadata about its current configuration to the Red Hat Insights platform. After registration, the system periodically updates

the metadata that it provides to Red Hat Insights. The system sends the metadata with TLS encryption to protect it in transit.

When the Red Hat Insights platform receives the data, it analyzes it and displays the result on the web console at the <https://cloud.redhat.com/insights> site.



**Figure 1.3: Insights high-level architecture**

## Install Red Hat Insights Clients

Insights is included with Red Hat Enterprise Linux 9 as part of the subscription. Earlier versions of Red Hat Enterprise Linux servers require installing the `insights-client` package on the system. The `insights-client` package replaced the `redhat-access-insights` package starting with Red Hat Enterprise Linux 7.5.

If you register your system for software entitlements through the Customer Portal Subscription Management service, then you can activate Insights with one command. Use the `insights-client --register` command to register the system.

```
[root@host ~]# insights-client --register
```

The Insights client periodically updates the metadata that is provided to Insights. Use the `insights-client` command to refresh the client's metadata.

```
[root@host ~]# insights-client
Starting to collect Insights data for host.example.com
Uploading Insights data.
Successfully uploaded report for host.example.com.
View details about this system on cloud.redhat.com:
https://cloud.redhat.com/insights/inventory/dc480efd-4782-417e-a496-cb33e23642f0
```

## Register a RHEL System with Red Hat Insights

Registering a RHEL server to Red Hat Insights is a quick task.

Interactively register the system with the Red Hat Subscription Management service.

```
[root@host ~]# subscription-manager register --auto-attach
```

Ensure that the `insights-client` package is installed on your system. The package is installed by default on RHEL 8 and later systems.

```
[root@host ~]# dnf install insights-client
```

Use the `insights-client --register` command to register the system with the Insights service and upload initial system metadata.

```
[root@host ~]# insights-client --register
```

Confirm that the system is visible under the **Inventory** section in the Insights web console at the <https://cloud.redhat.com/insights> site.

The screenshot shows the Red Hat Hybrid Cloud Console interface. On the left, there's a sidebar with various navigation options like Dashboard, Advisor, Drift, and Inventory. The Inventory option is currently selected. The main area is titled "Inventory" and displays a list of registered systems. The table has columns for Name, Tags, OS, and Last seen. Three entries are listed: "serverb.lab.example.com" (Tags: None, OS: RHEL 9.1, Last seen: Just now), "servera.lab.example.com" (Tags: None, OS: RHEL 9.1, Last seen: 3 minutes ago), and "workstation.lab.example.com" (Tags: None, OS: RHEL 9.1, Last seen: 10 minutes ago). There are also buttons for "Feedback" and "Help".

**Figure 1.4: Insights inventory on the Cloud Portal**

## Red Hat Insights Console Navigation

Insights provides a family of services that you access through the web console at the <https://cloud.redhat.com/insights> site.

## Detect Configuration Issues with the Advisor Service

The Advisor service reports configuration issues that impact your systems. You can access the service from the **Advisor > Recommendations** menu.

Name	Added	Category	Total risk	Risk of change	Systems	Ansible
SMBLoris' Samba denial of service with externally listening process	5 years ago	Security	Important	Very Low	1	Yes
Decreased security: Adobe Flash Player installed	6 months ago	Security	Moderate	Moderate	1	Yes
Decreased security: Yum GPG verification disabled (third-party repos)	4 years ago	Security	Important	Very Low	1	No
Traffic occurs or services are allowed unexpectedly when firewall zone drifting is enabled	2 years ago	Availability	Moderate	Moderate	1	No
Performance degradation of I/O when commands timeout due to faulty storage hardware	5 years ago	Stability	Important	Very Low	0	No
D-Bus timeout occurs when there are a large number of existing abrt crash directories	2 years ago	Availability	Moderate	Moderate	0	Yes
Asynchronous I/O related operations fail when kernel parameter fs.aio-max-nr limit is reached	3 years ago	Performance	Moderate	Low	0	No
Low density nodes detected	3 years ago	Performance	Moderate	Low	0	No
Running workload on nodes of an OpenShift cluster with an over-provisioned instance type size can result in cost increase	3 years ago	Performance	Moderate	Low	0	No

**Figure 1.5: Recommendations from the Advisor Service**

For each issue, Red Hat Insights provides information to help you to understand the problem, prioritize work to address it, determine what mitigation or remediation is available, and automate resolution with an Ansible Playbook. Red Hat Insights also provides links to Knowledgebase articles on the Customer Portal.

A denial of service flaw exists in Samba that allows a remote attacker to supply crafted NetBIOS Session Service headers and cause an out-of-memory state. A remote attacker in a position to supply the crafted data can render system unusable.

[Knowledgebase article](#)

[View the affected system](#)

**Total risk**  
The total risk of this remediation is **important**, based on the combination of likelihood and impact to remediate.

**Critical likelihood**  
**High impact**

**Risk of change**  
**Very Low**  
The risk of change is **very low**, because the change takes very little time to implement and there is minimal impact to system operations.

System reboot is **not required**.

**Figure 1.6: Details of an issue**

The Advisor service evaluates in two categories the risk that an issue presents to your system:

### Total risk

Indicates the impact of the issue on your system.

### Risk of change

Indicates the impact of the remediation action to your system. For example, you might need to restart the system.

## Assess Security with the Vulnerability Service

The Vulnerability service reports common vulnerabilities and exposures (CVEs) that impact your systems. You access the service from the Vulnerability > CVEs menu.

Figure 1.7: Report from the Vulnerability service

For each CVE, Insights provides additional information and lists the exposed systems. You can click the **Remediate** button to create an Ansible Playbook for remediation.

Figure 1.8: Details of a CVE

## Analyze Compliance by Using the Compliance Service

The Compliance service analyzes your systems and reports their compliance level to an OpenSCAP policy. The OpenSCAP project implements tools to check the compliance of a system against a set of rules. Red Hat Insights provides the rules to evaluate your systems against different policies, such as the Payment Card Industry Data Security Standard (PCI DSS).

## Update Packages with the Patch Service

The Patch service lists the Red Hat Product Advisories that apply to your systems. It can also generate an Ansible Playbook, which you can run to update the relevant RPM packages for the applicable advisories. To access the list of advisories for a specific system, use the **Patch > Systems** menu. Click the **Apply all applicable advisories** button to generate the Ansible Playbook for a system.

**Figure 1.9: Patching a system**

## Compare Systems with the Drift Service

With the Drift service, you can compare systems, or obtain a system history. You can use this service for troubleshooting, by comparing a system to a similar system, or to a previous system state. You can access the service from the Drift > Comparison menu.

The following figure shows that you can use Red Hat Insights to compare the same system at two different times:

**Figure 1.10: Comparing system history**

## Trigger Alerts with the Policies Service

By using the Policies service, you can create rules to monitor your systems and send alerts when a system does not comply with your rules. Red Hat Insights evaluates the rules every time that a system synchronizes its metadata. You can access the Policies service from the Policies menu.

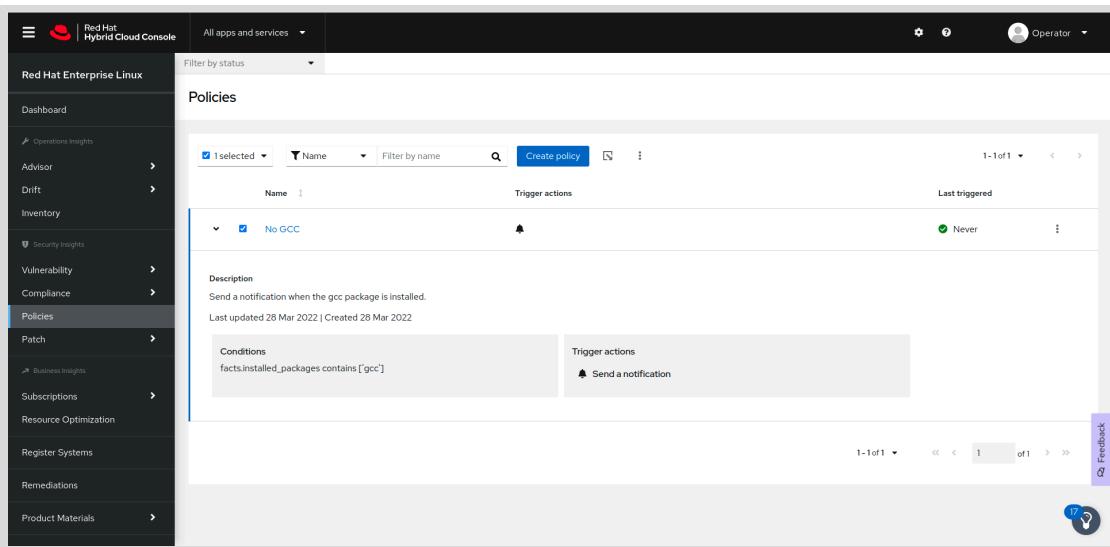


Figure 1.11: Details of a custom rule

## Inventory, Remediation Playbooks, and Subscriptions Monitoring

The **Inventory** page provides a list of the systems that you registered with Red Hat Insights. The **Last seen** column displays the time of the most recent metadata update for each system. By clicking a system name, you can review its details and directly access the Advisor, Vulnerability, Compliance, and Patch services for that system.

The **Remediations** page lists all the Ansible Playbooks that you created for remediation. You can download the playbooks from that page.

By using the **Subscription** page, you can monitor your Red Hat subscription usage.

□

### References

[insights-client\(8\)](#) and [insights-client.conf\(5\)](#) man pages

For more information about Red Hat Insights, refer to the *Product Documentation for Red Hat Insights* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_insights](https://access.redhat.com/documentation/en-us/red_hat_insights)

For more information about excluding data collected by Insights, refer to the *Red Hat Insights Client Data Obfuscation* and *Red Hat Insights Client Data Redaction* chapters in the *Client Configuration Guide for Red Hat Insights* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_insights/2021/html-single/client\\_configuration\\_guide\\_for\\_red\\_hat\\_insights/assembly-main-client-cg](https://access.redhat.com/documentation/en-us/red_hat_insights/2021/html-single/client_configuration_guide_for_red_hat_insights/assembly-main-client-cg)

Information about the data collected by Red Hat Insights is available at  
**System Information Collected by Red Hat Insights**  
<https://access.redhat.com/articles/1598863>

## ► Quiz

# Detect and Resolve Issues with Red Hat Insights

Choose the correct answers to the following questions:

► 1. In which order do the following events occur when managing a Red Hat Enterprise Linux system with Red Hat Insights?

- 1) Red Hat Insights analyzes system metadata to determine which issues and recommendations apply.
  - 2) The Insights client uploads system metadata to the Red Hat Insights service.
  - 3) The administrator views the recommended actions in the Red Hat Insights customer portal.
  - 4) The Insights client collects system metadata on the Red Hat Enterprise Linux system.
- a. 1, 2, 3, 4  
b. 4, 2, 1, 3  
c. 4, 2, 3, 1  
d. 4, 1, 2, 3

► 2. Which command do you use to register a client to Red Hat Insights?

- a. insights-client --register
- b. insights-client --no-upload
- c. subscription-manager register
- d. insights-client --unregister

► 3. From which page in the Red Hat Insights console can you generate an Ansible Playbook to update the RPM packages on a system?

- a. Advisor > Recommendations
- b. Vulnerability > Systems
- c. Patch > Systems
- d. Remediations

## ► Solution

# Detect and Resolve Issues with Red Hat Insights

Choose the correct answers to the following questions:

► 1. In which order do the following events occur when managing a Red Hat Enterprise Linux system with Red Hat Insights?

- 1) Red Hat Insights analyzes system metadata to determine which issues and recommendations apply.
  - 2) The Insights client uploads system metadata to the Red Hat Insights service.
  - 3) The administrator views the recommended actions in the Red Hat Insights customer portal.
  - 4) The Insights client collects system metadata on the Red Hat Enterprise Linux system.
- a. 1, 2, 3, 4  
b. 4, 2, 1, 3  
c. 4, 2, 3, 1  
d. 4, 1, 2, 3

► 2. Which command do you use to register a client to Red Hat Insights?

- a. insights-client --register
- b. insights-client --no-upload
- c. subscription-manager register
- d. insights-client --unregister

► 3. From which page in the Red Hat Insights console can you generate an Ansible Playbook to update the RPM packages on a system?

- a. Advisor > Recommendations
- b. Vulnerability > Systems
- c. Patch > Systems
- d. Remediations

# Summary

---

- You can use the `vim` editor to create and modify files from the command line.
- The `ssh-keygen` command generates an SSH key pair for authentication. The `ssh-copy-id` command exports the public key to remote systems.
- You can preconfigure SSH connections in the `~/.ssh/config` configuration file.
- The Red Hat Customer Portal provides access to documentation, downloads, optimization tools, support case management, and subscription and entitlement management for your Red Hat products.
- Red Hat Insights is a SaaS-based predictive analytics tool to help you to identify and remediate threats to your systems' security, performance, availability, and stability.

## Chapter 2

# Manage Files from the Command Line

### Goal

Copy, move, create, delete, and organize files from the Bash shell.

### Objectives

- Describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.
- Make multiple file names reference the same file with hard links and symbolic (or "soft") links.
- Efficiently run commands that affect many files by using pattern matching features of the Bash shell.

### Sections

- Describe Linux File System Hierarchy Concepts (and Quiz)
- Make Links Between Files (and Guided Exercise)
- Match File Names with Shell Expansions (and Quiz)

### Lab

- Manage Files from the Command Line

# Describe Linux File System Hierarchy Concepts

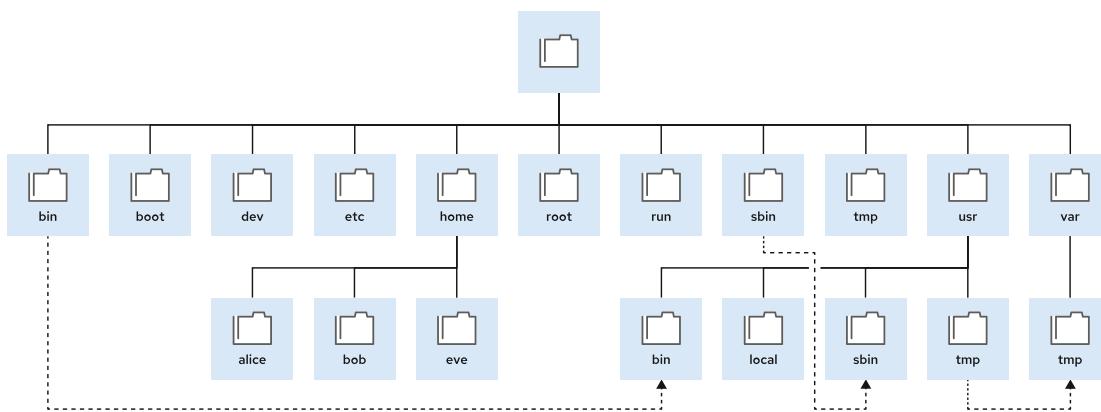
---

## Objectives

Describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.

## The File-system Hierarchy

The Linux system stores all files on file systems, which are organized into a single inverted tree known as a file-system hierarchy. This hierarchy is an inverted tree because the tree root is at the top, and the branches of directories and subdirectories stretch below the root.



**Figure 2.1: Significant file-system directories in Red Hat Enterprise Linux 9**

The `/` directory is the root directory at the top of the file-system hierarchy. The `/` character is also used as a directory separator in file names. For example, if `etc` is a subdirectory of the `/` directory, then refer to that directory as `/etc`. Likewise, if the `/etc` directory contains a file named `issue`, then refer to that file as `/etc/issue`.

Subdirectories of `/` are used for standardized purposes to organize files by type and purpose to make it easier to find files. For example, in the root directory, the subdirectory `/boot` is used for storing files to boot the system.



### Note

The following terms help to describe file-system directory contents:

- *Static* content remains unchanged until explicitly edited or reconfigured.
- *Dynamic* or *variable* content might be modified or appended by active processes.
- *Persistent* content remains after a reboot, such as configuration settings.
- *Runtime* content from a process or from the system is deleted on reboot.

The following table lists some of the significant directories on the system by name and purpose.

## Significant Red Hat Enterprise Linux Directories

Location	Purpose
/boot	Files to start the boot process.
/dev	Special device files that the system uses to access hardware.
/etc	System-specific configuration files.
/home	Home directory, where regular users store their data and configuration files.
/root	Home directory for the administrative superuser, root.
/run	Runtime data for processes that started since the last boot. This data includes process ID files and lock files. The contents of this directory are re-created on reboot. This directory consolidates the /var/run and /var/lock directories from earlier versions of Red Hat Enterprise Linux.
/tmp	A world-writable space for temporary files. Files that are not accessed, changed, or modified for 10 days are deleted from this directory automatically. The /var/tmp directory is also a temporary directory, in which files that are not accessed, changed, or modified in more than 30 days are deleted automatically.
/usr	Installed software, shared libraries, including files, and read-only program data. Significant subdirectories include: <ul style="list-style-type: none"> <li>• /usr/bin: User commands</li> <li>• /usr/sbin: System administration commands</li> <li>• /usr/local: Locally customized software</li> </ul>
/var	System-specific variable data should persist between boots. Files that dynamically change, such as databases, cache directories, log files, printer-spooled documents, and website content, might be found under /var.



### Important

In Red Hat Enterprise Linux 7 and later, four older directories in / have identical contents to their counterparts in /usr:

- /bin and /usr/bin
- /sbin and /usr/sbin
- /lib and /usr/lib
- /lib64 and /usr/lib64

Earlier versions of Red Hat Enterprise Linux had distinct directories with different sets of files. In Red Hat Enterprise Linux 7 and later, the directories in / are symbolic links to the matching directories in /usr.



### References

hier(7) man page

## ► Quiz

# Describe Linux File System Hierarchy Concepts

Choose the correct answers to the following questions:

- ▶ 1. Which directory contains persistent, system-specific configuration data?
  - a. /etc
  - b. /root
  - c. /run
  - d. /usr
  
- ▶ 2. Which directory is the top of the system's file-system hierarchy?
  - a. /etc
  - b. /
  - c. /home/root
  - d. /root
  
- ▶ 3. Which directory contains user home directories?
  - a. /
  - b. /home
  - c. /root
  - d. /user
  
- ▶ 4. Which directory contains files to boot the system?
  - a. /boot
  - b. /home/root
  - c. /bootable
  - d. /etc
  
- ▶ 5. Which directory contains system files to access hardware?
  - a. /etc
  - b. /run
  - c. /dev
  - d. /usr
  
- ▶ 6. Which directory is the administrative superuser's home directory?
  - a. /etc
  - b. /
  - c. /home/root
  - d. /root

► 7. Which directory contains regular commands and utilities?

- a. /commands
- b. /run
- c. /usr/bin
- d. /usr/sbin

► 8. Which directory contains non-persistent process runtime data?

- a. /tmp
- b. /etc
- c. /run
- d. /var

► 9. Which directory contains installed software programs and libraries?

- a. /etc
- b. /lib
- c. /usr
- d. /var

## ► Solution

# Describe Linux File System Hierarchy Concepts

Choose the correct answers to the following questions:

- ▶ 1. Which directory contains persistent, system-specific configuration data?
  - a. /etc
  - b. /root
  - c. /run
  - d. /usr
  
- ▶ 2. Which directory is the top of the system's file-system hierarchy?
  - a. /etc
  - b. /
  - c. /home/root
  - d. /root
  
- ▶ 3. Which directory contains user home directories?
  - a. /
  - b. /home
  - c. /root
  - d. /user
  
- ▶ 4. Which directory contains files to boot the system?
  - a. /boot
  - b. /home/root
  - c. /bootable
  - d. /etc
  
- ▶ 5. Which directory contains system files to access hardware?
  - a. /etc
  - b. /run
  - c. /dev
  - d. /usr
  
- ▶ 6. Which directory is the administrative superuser's home directory?
  - a. /etc
  - b. /
  - c. /home/root
  - d. /root

► **7. Which directory contains regular commands and utilities?**

- a. /commands
- b. /run
- c. /usr/bin
- d. /usr/sbin

► **8. Which directory contains non-persistent process runtime data?**

- a. /tmp
- b. /etc
- c. /run
- d. /var

► **9. Which directory contains installed software programs and libraries?**

- a. /etc
- b. /lib
- c. /usr
- d. /var

# Make Links Between Files

## Objectives

Make multiple file names reference the same file with hard links and symbolic (or "soft") links.

## Manage Links Between Files

You can create multiple file names that point to the same file. These file names are called *links*.

You can create two types of links: a *hard link*, or a *symbolic link* (sometimes called a *soft link*). Each way has its advantages and disadvantages.

### Create Hard Links

Every file starts with a single hard link, from its initial name to the data on the file system. When you create a hard link to a file, you create another name that points to that same data. The new hard link acts exactly like the original file name. After the link is created, you cannot tell the difference between the new hard link and the original name of the file.

You can determine whether a file has multiple hard links by using the `ls -l` command. One item it reports is each file's *link count*, the number of hard links that the file has. In the next example, the link count of the `newfile.txt` file is 1. It has exactly one absolute path, which is the `/home/user/newfile.txt` location.

```
[user@host ~]$ pwd  
/home/user  
[user@host ~]$ ls -l newfile.txt  
-rw-r--r--. 1 user user 0 Mar 11 19:19 newfile.txt
```

You can use the `ln` command to create a hard link (another file name) that points to an existing file. The command needs at least two arguments: a path to the existing file, and the path to the hard link that you want to create.

The following example creates a hard link called `newfile-hlink2.txt` for the existing `newfile.txt` file in the `/tmp` directory.

```
[user@host ~]$ ln newfile.txt /tmp/newfile-hlink2.txt  
[user@host ~]$ ls -l newfile.txt /tmp/newfile-hlink2.txt  
-rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt  
-rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```

To determine whether two files are hard linked, use the `ls` command `-i` option to list each file's *inode number*. If the files are on the same file system and their inode numbers are the same, then the files are hard links that point to the same data file content.

```
[user@host ~]$ ls -il newfile.txt /tmp/newfile-hlink2.txt  
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt  
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```



### Important

Hard links that reference the same file share the same inode struct with the link count, access permissions, user and group ownership, time stamps, and file content. When that information is changed for one hard link, then the other hard links for the same file show also the new information. This is because each hard link points to the same data on the storage device.

Even if the original file is deleted, you can still access the contents of the file provided that at least one other hard link exists. Data is deleted from storage only when the last hard link is deleted, making the file contents unreferenced by any hard link.

```
[user@host ~]$ rm -f newfile.txt
[user@host ~]$ ls -l /tmp/newfile-hlink2.txt
-rw-rw-r-- 1 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
[user@host ~]$ cat /tmp/newfile-hlink2.txt
Hello World
```

## Limitations of Hard Links

Hard links have some limitations. First, you can use hard links only with regular files. You cannot use the `ln` command to create a hard link to a directory or special file.

Second, you can use hard links only if both files are on the same *file system*. The file-system hierarchy can be composed of multiple storage devices. Depending on the configuration of your system, when you change into a new directory, that directory and its contents might be stored on a different file system.

You can use the `df` command to list the directories that are on different file systems. For example, you might see the following output:

```
[user@host ~]$ df
Filesystem      1K-blocks   Used Available Use% Mounted on
devtmpfs          886788     0   886788  0% /dev
tmpfs            902108     0   902108  0% /dev/shm
tmpfs            902108   8696   893412  1% /run
tmpfs            902108     0   902108  0% /sys/fs/cgroup
/dev/mapper/rhel_rhel9--root 10258432 1630460   8627972 16% /
/dev/sda1        1038336  167128   871208 17% /boot
tmpfs           180420     0   180420  0% /run/user/1000
```

Files in two different "Mounted on" directories and their subdirectories are on different file systems. So, in the system in this example, you can create a hard link between the `/var/tmp/link1` and `/home/user/file` files, because they are both subdirectories of the `/` directory but not of any other directory on the list. However, you cannot create a hard link between the `/boot/test/badlink` and `/home/user/file` files, because the first file is in a subdirectory of the `/boot` directory (on the "Mounted on" list) and it is in the `/dev/sda1` file system, while the second file is in the `/dev/mapper/rhel_rhel9--root` file system.

## Create Symbolic Links

The `ln` command `-s` option creates a symbolic link, which is also called a "soft link". A symbolic link is not a regular file, but a special type of file that points to an existing file or directory.

Symbolic links have some advantages over hard links:

- Symbolic links can link two files on different file systems.
- Symbolic links can point to a directory or special file, not just to a regular file.

In the following example, the `ln -s` command creates a symbolic link for the `/home/user/newfile-link2.txt` file. The name for the symbolic link is `/tmp/newfile-symlink.txt`.

```
[user@host ~]$ ln -s /home/user/newfile-link2.txt /tmp/newfile-symlink.txt
[user@host ~]$ ls -l newfile-link2.txt /tmp/newfile-symlink.txt
-rw-rw-r--. 1 user user 12 Mar 11 19:19 newfile-link2.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
Symbolic Hello World
```

In the preceding example, the first character of the long listing for the `/tmp/newfile-symlink.txt` file is `l` (letter l) instead of `-`. This character indicates that the file is a symbolic link and not a regular file.

When the original regular file is deleted, the symbolic link still points to the file but the target is gone. A symbolic link that points to a missing file is called a "dangling symbolic link".

```
[user@host ~]$ rm -f newfile-link2.txt
[user@host ~]$ ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```



### Important

One side-effect of the dangling symbolic link in the preceding example is that if you create a file with the same name as the deleted file (`/home/user/newfile-link2.txt`), then the symbolic link is no longer "dangling" and points to the new file. Hard links do not work in this way. If you delete a hard link and then use normal tools (rather than `ln`) to create a file with the same name, then the new file is not linked to the old file. Consider the following way to compare hard links and symbolic links, to understand how they work:

- A hard link points a name to data on a storage device.
- A symbolic link points a name to another name, which points to data on a storage device.

A symbolic link can point to a directory. The symbolic link then acts like the directory. If you use `cd` to change to the symbolic link, then the current working directory becomes the linked directory. Some tools might track that you followed a symbolic link to get there. For example, by default, `cd` updates your current working directory by using the name of the symbolic link rather than the name of the actual directory. If you want to update the current working directory by using the name of the actual directory, then you can use the `-P` option.

The following example creates a symbolic link called `/home/user/configfiles` that points to the `/etc` directory.

```
[user@host ~]$ ln -s /etc /home/user/configfiles
[user@host ~]$ cd /home/user/configfiles
[user@host configfiles]$ pwd
/home/user/configfiles
[user@host configfiles]$ cd -P /home/user/configfiles
[user@host etc]$ pwd
/etc
```



## References

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

[info ln \(Make links between files\)](#)

## ► Guided Exercise

# Make Links Between Files

In this exercise, you create hard links and symbolic links and compare the results.

### Outcomes

- Create hard links and symbolic links between files.

### Before You Begin

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start files-make
```

### Instructions

- 1. Use the ssh command to log in to the servera machine as the student user. The system's configuration supports the use SSH keys for authentication; therefore, you do not require a password.

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

- 2. Create a hard link called /home/student/links/file.hardlink for the /home/student/files/target.file file. Verify the link count for the original file and the new linked file.

- 2.1. View the link count for the /home/student/files/target.file file.

```
[student@servera ~]$ ls -l files/target.file  
total 4  
-rw-r--r--. 1 student student 11 Mar 3 06:51 files/target.file
```

- 2.2. Create a hard link called /home/student/links/file.hardlink. Link it to the /home/student/files/target.file file.

```
[student@servera ~]$ ln /home/student/files/target.file \  
/home/student/links/file.hardlink
```

- 2.3. Verify the link count for the original /home/student/files/target.file file and the new linked file, /home/student/files/file.hardlink. The link count should be 2 for both files.

```
[student@servera ~]$ ls -l files/target.file links/file.hardlink
-rw-r--r-- 2 student student 11 Mar 3 06:51 files/target.file
-rw-r--r-- 2 student student 11 Mar 3 06:51 links/file.hardlink
```

- 3. Create a symbolic link called /home/student/tempdir that points to the /tmp directory on the servera machine. Verify the newly created symbolic link.

- 3.1. Create a symbolic link called /home/student/tempdir and link it to the /tmp directory.

```
[student@servera ~]$ ln -s /tmp /home/student/tempdir
```

- 3.2. Use the ls -l command to verify the newly created symbolic link.

```
[student@servera ~]$ ls -l /home/student/tempdir
lrwxrwxrwx. 1 student student 4 Mar 3 06:55 /home/student/tempdir -> /tmp
```

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

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

## 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 files-make
```

This concludes the section.

# Match File Names with Shell Expansions

---

## Objectives

Efficiently run commands that affect many files by using pattern matching features of the Bash shell.

## Command-line Expansions

The Bash shell has multiple ways of expanding a command line, including *pattern matching*, home directory expansion, string expansion, and variable substitution. Perhaps the most powerful of these ways is the path name-matching capability, historically called *globbing*. With the Bash globbing feature, sometimes called *wildcards*, it is easier to manage many files. By using *metacharacters* that "expand" to match file and path names that are sought, commands act on a focused set of files at once.

## Pattern Matching

Globbing is a shell command-parsing operation that expands a wildcard pattern into a list of matching path names. Before command execution, the shell replaces the command-line metacharacters by the match list. Patterns that do not return matches display the original pattern request as literal text. The following table lists common metacharacters and pattern classes.

**Table of Metacharacters and Matches**

Pattern	Matches
*	Any string of zero or more characters.
?	Any single character.
[abc...]	Any one character in the enclosed class (between the square brackets).
[!abc...]	Any one character <i>not</i> in the enclosed class.
[^abc...]	Any one character <i>not</i> in the enclosed class.
[[:alpha:]]	Any alphabetic character.
[[:lower:]]	Any lowercase character.
[[:upper:]]	Any uppercase character.
[[:alnum:]]	Any alphabetic character or digit.
[[:punct:]]	Any printable character that is not a space or alphanumeric.
[[:digit:]]	Any single digit from 0 to 9.
[[:space:]]	Any single white space character, which might include tabs, newlines, carriage returns, form feeds, or spaces.

For the next example, imagine that you ran the following commands to create some sample files:

```
[user@host ~]$ mkdir glob; cd glob
[user@host glob]$ touch alpha bravo charlie delta echo able baker cast dog easy
[user@host glob]$ ls
able alpha baker bravo cast charlie delta dog easy echo
[user@host glob]$
```

In the next example, the first two commands use simple pattern matches with the asterisk (\*) to match all the filenames that start with "a" and all the filenames that contain an "a", respectively. The third command uses the asterisk and square brackets to match all the filenames that start with "a" or "c".

```
[user@host glob]$ ls a*
able alpha
[user@host glob]$ ls *a*
able alpha baker bravo cast charlie delta easy
[user@host glob]$ ls [ac]*
able alpha cast charlie
```

The next example uses also question mark (?) characters to match some of those file names. The two commands match only filenames with four and five characters in length, respectively.

```
[user@host glob]$ ls ****
able alpha cast easy echo
[user@host glob]$ ls ??????
baker bravo delta
```

## Tilde Expansion

The tilde character (~), matches the current user's home directory. If it starts with a string of characters other than a slash (/), then the shell interprets the string up to that slash as a user name, if one matches, and replaces the string with the absolute path to that user's home directory. If no user name matches, then the shell uses an actual tilde followed by the string of characters.

In the following example, the echo command is used to display the value of the tilde character. You can also use the echo command to display the values of brace and variable expansion characters, and others.

```
[user@host glob]$ echo ~root
/root
[user@host glob]$ echo ~user
/home/user
[user@host glob]$ echo ~/glob
/home/user/glob
[user@host glob]$ echo ~nonexistinguser
~nonexistinguser
```

## Brace Expansion

Brace expansion is used to generate discretionary strings of characters. Braces contain a comma-separated list of strings, or a sequence expression. The result includes the text that precedes or

## Chapter 2 | Manage Files from the Command Line

follows the brace definition. Brace expansions might be nested, one inside another. You can also use double-dot syntax (..), which expands to a sequence. For example, the {m .. p} double-dot syntax inside braces expands to m n o p.

```
[user@host glob]$ echo {Sunday,Monday,Tuesday,Wednesday}.log
Sunday.log Monday.log Tuesday.log Wednesday.log
[user@host glob]$ echo file{1..3}.txt
file1.txt file2.txt file3.txt
[user@host glob]$ echo file{a..c}.txt
filea.txt fileb.txt filec.txt
[user@host glob]$ echo file{a,b}{1,2}.txt
filea1.txt filea2.txt fileb1.txt fileb2.txt
[user@host glob]$ echo file{a{1,2},b,c}.txt
filea1.txt filea2.txt fileb.txt filec.txt
```

A practical use of brace expansion is to quickly create multiple files or directories.

```
[user@host glob]$ mkdir ../RHEL{7,8,9}
[user@host glob]$ ls ../RHEL*
RHEL7 RHEL8 RHEL9
```

## Variable Expansion

A variable acts like a named container that stores a value in memory. Variables simplify accessing and modifying the stored data either from the command line or within a shell script.

You can assign data as a value to a variable with the following syntax:

```
[user@host ~]$ VARIABLENAME=value
```

You can use variable expansion to convert the variable name to its value on the command line. If a string starts with a dollar sign (\$), then the shell tries to use the rest of that string as a variable name and replace it with the variable value.

```
[user@host ~]$ USERNAME=operator
[user@host ~]$ echo $USERNAME
operator
```

To prevent mistakes due to other shell expansions, you can put the name of the variable in curly braces, for example \${VARIABLENAME}.

```
[user@host ~]$ USERNAME=operator
[user@host ~]$ echo ${USERNAME}
operator
```

Variable names can contain only letters (uppercase and lowercase), numbers, and underscores. Variable names are case-sensitive and cannot start with a number.

## Command Substitution

Command substitution allows the output of a command to replace the command itself on the command line. Command substitution occurs when you enclose a command in parentheses and

precede it by a dollar sign (\$). The `$(command)` form can nest multiple command expansions inside each other.

```
[user@glob]$ echo Today is $(date +%A).
Today is Wednesday.
[user@glob]$ echo The time is $(date +%M) minutes past $(date +%l%p).
The time is 26 minutes past 11AM.
```



### Note

An older form of command substitution uses backticks: `command`. Although the Bash shell still accepts this format, try to avoid it because it is easy to visually confuse backticks with single quotation marks, and backticks cannot be nested.

## Protecting Arguments from Expansion

Many characters have a special meaning in the Bash shell. To prevent shell expansions on parts of your command line, you can quote and escape characters and strings.

The backslash (\) is an escape character in the Bash shell. It protects the following character from expansion.

```
[user@glob]$ echo The value of $HOME is your home directory.
The value of /home/user is your home directory.
[user@glob]$ echo The value of \$HOME is your home directory.
The value of $HOME is your home directory.
```

In the preceding example, with the dollar sign protected from expansion, Bash treats it as a regular character, without variable expansion on \$HOME.

To protect longer character strings, you can use single quotation marks (' ) or double quotation marks (" ) to enclose strings. They have slightly different effects. Single quotation marks stop all shell expansion. Double quotation marks stop most shell expansion.

Double quotation marks suppress globbing and shell expansion, but still allow command and variable substitution.

```
[user@glob]$ myhost=$(hostname -s); echo $myhost
host
[user@glob]$ echo "***** hostname is ${myhost} ****"
***** hostname is host *****
```

Use single quotation marks to interpret *all* text literally.

```
[user@glob]$ echo "Will variable $myhost evaluate to $(hostname -s)?"
Will variable host evaluate to host?
[user@glob]$ echo 'Will variable $myhost evaluate to $(hostname -s)?'
Will variable $myhost evaluate to $(hostname -s)?
```



### Important

It is easy to confuse the single quotation mark ( ' ) and the command substitution backtick ( ` ), on both the screen and the keyboard. Use of one when you mean to use the other leads to unexpected shell behavior.



### References

`bash(1)`, `cd(1)`, `glob(7)`, `isalpha(3)`, `ls(1)`, `path_resolution(7)`, and `pwd(1)`  
man pages

## ► Quiz

# Match File Names with Shell Expansions

Choose the correct answers to the following questions:

► 1. Which pattern matches only file names that end with "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [ !b ]\*

► 2. Which pattern matches only file names that begin with "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [ !b ]\*

► 3. Which pattern matches only file names where the first character is not "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [ !b ]\*

► 4. Which pattern matches all file names that contain a "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [ !b ]\*

► 5. Which pattern matches only file names that contain a number?

- a. \*#\*
- b. \*[[ :digit: ]]\*
- c. \*[digit]\*
- d. [0-9]

► 6. Which pattern matches only file names that begin with an uppercase letter?

- a. ^?\*
- b. ^\*
- c. [upper]\*
- d. [[ :upper: ]]\*
- e. [[CAP]]\*

► 7. Which pattern matches only file names with at least three characters?

- a. ???\*
- b. ???
- c. \3\*
- d. +++\*
- e. ....\*

## ► Solution

# Match File Names with Shell Expansions

Choose the correct answers to the following questions:

► 1. Which pattern matches only file names that end with "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [!b]\*

► 2. Which pattern matches only file names that begin with "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [!b]\*

► 3. Which pattern matches only file names where the first character is not "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [!b]\*

► 4. Which pattern matches all file names that contain a "b"?

- a. b\*
- b. \*b
- c. \*b\*
- d. [!b]\*

► 5. Which pattern matches only file names that contain a number?

- a. \*#\*
- b. \*[[digit:]]\*
- c. \*[digit]\*
- d. [0-9]

► 6. Which pattern matches only file names that begin with an uppercase letter?

- a. ^?\*
- b. ^\*
- c. [upper]\*
- d. [[upper:]]\*
- e. [[CAP]]\*

► 7. Which pattern matches only file names with at least three characters?

- a. ???\*
- b. ???
- c. \3\*
- d. +++\*
- e. ...\*

## ▶ Lab

# Manage Files from the Command Line

In this lab, you efficiently create, move, and remove files and directories by using the shell and various file name matching techniques.

## Outcomes

- Use wildcards to locate and manipulate files.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. Use the `ssh` command to log in to the `serverb` machine as the `student` user. The system's configuration supports the use of SSH keys for authentication.
2. Create a directory called `project_plans` in the `Documents` directory. The `Documents` directory should be placed in the `student` user's home directory. Create two empty files in the `project_plans` directory called `season1_project_plan.odf` and `season2_project_plan.odf`. Hint: If the `~/Documents` directory does not exist, then use the `mkdir` command `-p` option to create it.
3. Create sets of empty practice files to use in this lab. If you do not immediately recognize the intended shell expansion shortcut, then use the solution to learn and practice. Use shell tab completion to locate file path names easily. Create 12 files with names `tv_seasonX_episodeY.ogg` in the `/home/student` directory. Replace `X` with the season number and `Y` with that season's episode, for two seasons of six episodes each.
4. As the author of a successful series of mystery novels, you are editing your next bestseller's chapters for publishing. Create eight files with names `mystery_chapterX.odf`. Replace `X` with the numbers 1 through 8.
5. Use a single command to create two subdirectories called `season1` and `season2` under the `Videos` directory to organize the TV episodes. Move the appropriate TV episodes into the season subdirectories. Use only two commands, and specify destinations with relative syntax.
6. Create a two-level directory hierarchy with a single command to organize the mystery book chapters. Create the `my_bestseller` subdirectory under the `Documents` directory, and the `chapters` subdirectory under the new `my_bestseller` directory. Create three more subdirectories directly under the `my_bestseller` directory with a single command. Name these subdirectories `editor`, `changes`, and `vacation`. You do not need to use the `mkdir -p` command to create parents because the `my_bestseller` parent directory exists.

7. Change to the `chapters` directory. Use the tilde (~) home directory shortcut to move all book chapters to the `chapters` directory, which is now your current directory. Use the simplest syntax to specify the destination directory.
- You want to send the first two chapters to the editor for review. Move only those two chapters to the `editor` directory to avoid modifying them during the review. Starting from the `chapters` subdirectory, use brace expansion with a range to specify the chapter file names to move and a relative path for the destination directory.
- While on vacation, you intend to write chapters 7 and 8. Use a single command to move the files from the `chapters` directory to the `vacation` directory. Specify the chapter file names by using brace expansion with a list of strings and without using wildcard characters.
8. Change your working directory to `~/Videos/season2`, and then copy the first episode of the season to the `vacation` directory. Use a single `cd` command to change from your working directory to the `~/Documents/my_bestseller/vacation` directory. List its files. Use the *previous working directory* argument to return to the `season2` directory. (This argument succeeds if the last directory change with the `cd` command used one command rather than several `cd` commands.) From the `season2` directory, copy the episode 2 file into the `vacation` directory. Use the shortcut again to return to the `vacation` directory.
9. The authors of chapters 5 and 6 want to experiment with possible changes. Copy both files from the `~/Documents/my_bestseller/chapters` directory to the `~/Documents/my_bestseller/changes` directory to prevent these changes from modifying original files. Navigate to the `~/Documents/my_bestseller` directory. Use square-bracket pattern matching to specify which chapter numbers to match in the filename argument of the `cp` command.
10. Change your current directory to the `changes` directory and use the `date +%F` command with command substitution to copy `mystery_chapter5.odf` to a new file that includes the full date. Use the `mystery_chapter5_YYYY-MM-DD.odf` name format.
- By using command substitution with the `date +%s` command, make another copy of `mystery_chapter5.odf`, and append the current time stamp (as the number of seconds since the epoch, 1970-01-01 00:00 UTC) to ensure a unique file name.
11. After further review, you decide that you do not need the plot changes. Delete the `changes` directory.
- If it is necessary, then navigate to the `changes` directory and delete all the files within the directory. You cannot delete a directory while it is the current working directory.
- Change to the parent directory of the `changes` directory. Try to delete the empty directory by using the `rm` command without the `-r` recursive option. This attempt should fail. Finally, use the `rmdir` command to delete the empty directory, which succeeds.
- When the vacation is over, you no longer need the `vacation` directory. Delete it by using the `rm` command with the `recursive` option.
- When finished, return to the `student` user's home directory.
12. Create a hard link to the `~/Documents/project_plans/season2_project_plan.odf` file called `~/Documents/backups/season2_project_plan.odf.back`. A hard link protects against accidental deletion of the original file and keeps the backup file updated as you change the original file. Hint: If the `~/Documents/backups` directory does not exist, then use the `mkdir` command to create it.

## 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 files-review
```

## 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 files-review
```

This concludes the section.

## ► Solution

# Manage Files from the Command Line

In this lab, you efficiently create, move, and remove files and directories by using the shell and various file name matching techniques.

## Outcomes

- Use wildcards to locate and manipulate files.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. Use the `ssh` command to log in to the `serverb` machine as the `student` user. The system's configuration supports the use of SSH keys for authentication.

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

2. Create a directory called `project_plans` in the `Documents` directory. The `Documents` directory should be placed in the `student` user's home directory. Create two empty files in the `project_plans` directory called `season1_project_plan.odf` and `season2_project_plan.odf`. Hint: If the `~/Documents` directory does not exist, then use the `mkdir` command `-p` option to create it.

```
[student@serverb ~]$ mkdir -p ~/Documents/project_plans  
[student@serverb ~]$ touch \  
~/Documents/project_plans/{season1,season2}_project_plan.odf  
[student@serverb ~]$ ls -lR Documents/  
Documents/:  
total 0  
drwxr-xr-x. 2 student student 70 Mar 7 03:50 project_plans  
  
Documents/project_plans:  
total 0  
-rw-r--r--. 1 student student 0 Mar 7 03:50 season1_project_plan.odf  
-rw-r--r--. 1 student student 0 Mar 7 03:50 season2_project_plan.odf
```

3. Create sets of empty practice files to use in this lab. If you do not immediately recognize the intended shell expansion shortcut, then use the solution to learn and practice.

Use shell tab completion to locate file path names easily. Create 12 files with names `tv_seasonX_episodeY.ogg` in the `/home/student` directory. Replace `X` with the season number and `Y` with that season's episode, for two seasons of six episodes each.

```
[student@serverb ~]$ touch tv_season{1..2}_episode{1..6}.ogg
[student@serverb ~]$ ls tv*
tv_season1_episode1.ogg  tv_season1_episode5.ogg  tv_season2_episode3.ogg
tv_season1_episode2.ogg  tv_season1_episode6.ogg  tv_season2_episode4.ogg
tv_season1_episode3.ogg  tv_season2_episode1.ogg  tv_season2_episode5.ogg
tv_season1_episode4.ogg  tv_season2_episode2.ogg  tv_season2_episode6.ogg
```

4. As the author of a successful series of mystery novels, you are editing your next bestseller's chapters for publishing. Create eight files with names `mystery_chapterX.odf`. Replace `X` with the numbers 1 through 8.

```
[student@serverb ~]$ touch mystery_chapter{1..8}.odf
[student@serverb ~]$ ls mys*
mystery_chapter1.odf  mystery_chapter4.odf  mystery_chapter7.odf
mystery_chapter2.odf  mystery_chapter5.odf  mystery_chapter8.odf
mystery_chapter3.odf  mystery_chapter6.odf
```

5. Use a single command to create two subdirectories called `season1` and `season2` under the `Videos` directory to organize the TV episodes. Move the appropriate TV episodes into the `season` subdirectories. Use only two commands, and specify destinations with relative syntax.

- 5.1. Create two subdirectories called `season1` and `season2` under the `Videos` directory by using a single command.

```
[student@serverb ~]$ mkdir -p Videos/season{1..2}
[student@serverb ~]$ ls Videos
season1  season2
```

- 5.2. Move the appropriate TV episodes into the `season` subdirectories by using only two commands.

```
[student@serverb ~]$ mv tv_season1* Videos/season1
[student@serverb ~]$ mv tv_season2* Videos/season2
[student@serverb ~]$ ls -R Videos
Videos:
season1  season2

Videos/season1:
tv_season1_episode1.ogg  tv_season1_episode3.ogg  tv_season1_episode5.ogg
tv_season1_episode2.ogg  tv_season1_episode4.ogg  tv_season1_episode6.ogg

Videos/season2:
tv_season2_episode1.ogg  tv_season2_episode3.ogg  tv_season2_episode5.ogg
tv_season2_episode2.ogg  tv_season2_episode4.ogg  tv_season2_episode6.ogg
```

6. Create a two-level directory hierarchy with a single command to organize the mystery book chapters. Create the `my_bestseller` subdirectory under the `Documents` directory, and the `chapters` subdirectory under the new `my_bestseller` directory. Create three more subdirectories directly under the `my_bestseller` directory with a single command. Name

these subdirectories `editor`, `changes`, and `vacation`. You do not need to use the `mkdir -p` command to create parents because the `my_bestseller` parent directory exists.

- 6.1. Create the `my_bestseller` directory under the `Documents` directory. Create the `chapters` directory under the `my_bestseller` directory.

```
[student@serverb ~]$ mkdir -p Documents/my_bestseller/chapters
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller  project_plans

Documents/my_bestseller:
chapters

Documents/my_bestseller/chapters:
season1_project_plan.odf  season2_project_plan.odf
```

- 6.2. Create three directories called `editor`, `changes`, and `vacation`, under the `my_bestseller` directory by using a single command.

```
[student@serverb ~]$ mkdir Documents/my_bestseller/{editor,changes,vacation}
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller  project_plans

Documents/my_bestseller:
changes  chapters  editor  vacation

Documents/my_bestseller/changes:

Documents/my_bestseller/chapters:

Documents/my_bestseller/editor:

Documents/my_bestseller/vacation:

Documents/project_plans:
season1_project_plan.odf  season2_project_plan.odf
```

7. Change to the `chapters` directory. Use the tilde (~) home directory shortcut to move all book chapters to the `chapters` directory, which is now your current directory. Use the simplest syntax to specify the destination directory.

You want to send the first two chapters to the `editor` for review. Move only those two chapters to the `editor` directory to avoid modifying them during the review. Starting from the `chapters` subdirectory, use brace expansion with a range to specify the chapter file names to move and a relative path for the destination directory.

While on vacation, you intend to write chapters 7 and 8. Use a single command to move the files from the `chapters` directory to the `vacation` directory. Specify the chapter file names by using brace expansion with a list of strings and without using wildcard characters.

- 7.1. Change to the `chapters` directory and use the tilde (~) home directory shortcut to move all book chapters to the `chapters` directory.

```
[student@serverb ~]$ cd Documents/my_bestseller/chapters
[student@serverb chapters]$ mv ~/mystery_chapter* .
[student@serverb chapters]$ ls
mystery_chapter1.odf  mystery_chapter4.odf  mystery_chapter7.odf
mystery_chapter2.odf  mystery_chapter5.odf  mystery_chapter8.odf
mystery_chapter3.odf  mystery_chapter6.odf
```

- 7.2. Move the first two chapters to the `editor` directory. Use brace expansion with a range to specify the chapter file names to move and a relative path for the destination directory.

```
[student@serverb chapters]$ mv mystery_chapter{1..2}.odf ../editor
[student@serverb chapters]$ ls
mystery_chapter3.odf  mystery_chapter5.odf  mystery_chapter7.odf
mystery_chapter4.odf  mystery_chapter6.odf  mystery_chapter8.odf
[student@serverb chapters]$ ls ../editor
mystery_chapter1.odf  mystery_chapter2.odf
```

- 7.3. Use a single command to move the chapters 7 and 8 from the `chapters` directory to the `vacation` directory. Specify the chapter file names by using brace expansion with a list of strings and without using wildcard characters.

```
[student@serverb chapters]$ mv mystery_chapter{7,8}.odf ../vacation
[student@serverb chapters]$ ls
mystery_chapter3.odf  mystery_chapter5.odf
mystery_chapter4.odf  mystery_chapter6.odf
[student@serverb chapters]$ ls ../vacation
mystery_chapter7.odf  mystery_chapter8.odf
```

8. Change your working directory to `~/Videos/season2`, and then copy the first episode of the season to the `vacation` directory. Use a single `cd` command to change from your working directory to the `~/Documents/my_bestseller/vacation` directory. List its files. Use the *previous working directory* argument to return to the `season2` directory. (This argument succeeds if the last directory change with the `cd` command used one command rather than several `cd` commands.) From the `season2` directory, copy the episode 2 file into the `vacation` directory. Use the shortcut again to return to the `vacation` directory.

- 8.1. Change your working directory to `~/Videos/season2`, and then copy the first episode of the season to the `vacation` directory.

```
[student@serverb chapters]$ cd ~/Videos/season2
[student@serverb season2]$ cp *episode1.ogg ~/Documents/my_bestseller/vacation
```

- 8.2. Use a single `cd` command to change from your working directory to the `~/Documents/my_bestseller/vacation` directory, list its files, and use the `-` argument to return to the previous directory. Copy the episode 2 file into the `vacation` directory. Use the `cd` command with the `-` argument to return to the `vacation` directory.

```
[student@serverb season2]$ cd ~/Documents/my_bestseller/vacation
[student@serverb vacation]$ ls
mystery_chapter7.odf  mystery_chapter8.odf  tv_season2_episode1.ogg
```

```
[student@serverb vacation]$ cd -
/home/student/Videos/season2
[student@serverb season2]$ cp *episode2.ogg ~/Documents/my_bestseller/vacation
[student@serverb season2]$ cd -
/home/student/Documents/my_bestseller/vacation
[student@serverb vacation]$ ls
mystery_chapter7.odf  tv_season2_episode1.ogg
mystery_chapter8.odf  tv_season2_episode2.ogg
```

9. The authors of chapters 5 and 6 want to experiment with possible changes. Copy both files from the ~/Documents/my\_bestseller/chapters directory to the ~/Documents/my\_bestseller/changes directory to prevent these changes from modifying original files. Navigate to the ~/Documents/my\_bestseller directory. Use square-bracket pattern matching to specify which chapter numbers to match in the filename argument of the cp command.

```
[student@serverb vacation]$ cd ~/Documents/my_bestseller
[student@serverb my_bestseller]$ cp chapters/mystery_chapter[56].odf changes
[student@serverb my_bestseller]$ ls chapters
mystery_chapter3.odf  mystery_chapter5.odf
mystery_chapter4.odf  mystery_chapter6.odf
[student@serverb my_bestseller]$ ls changes
mystery_chapter5.odf  mystery_chapter6.odf
```

10. Change your current directory to the changes directory and use the date +%F command with command substitution to copy mystery\_chapter5.odf to a new file that includes the full date. Use the mystery\_chapter5\_YYYY-MM-DD.odf name format.

By using command substitution with the date +%s command, make another copy of mystery\_chapter5.odf, and append the current time stamp (as the number of seconds since the epoch, 1970-01-01 00:00 UTC) to ensure a unique file name.

```
[student@serverb my_bestseller]$ cd changes
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%F).odf
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%s).odf
[student@serverb changes]$ ls
mystery_chapter5_1646644424.odf  mystery_chapter5.odf
mystery_chapter5_2022-03-07.odf  mystery_chapter6.odf
```

11. After further review, you decide that you do not need the plot changes. Delete the changes directory.

If it is necessary, then navigate to the changes directory and delete all the files within the directory. You cannot delete a directory while it is the current working directory.

Change to the parent directory of the changes directory. Try to delete the empty directory by using the rm command without the -r recursive option. This attempt should fail. Finally, use the rmdir command to delete the empty directory, which succeeds.

When the vacation is over, you no longer need the vacation directory. Delete it by using the rm command with the recursive option.

When finished, return to the student user's home directory.

- 11.1. Delete the changes directory. Change to the parent directory of the changes directory and try to delete the empty directory by using the rm command without the

-r recursive option, which should fail. Use the rmdir command to delete the empty directory.

```
[student@serverb changes]$ rm mystery*
[student@serverb changes]$ cd ..
[student@serverb my_bestseller]$ rm changes
rm: cannot remove 'changes': Is a directory
[student@serverb my_bestseller]$ rmdir changes
[student@serverb my_bestseller]$ ls
chapters editor vacation
```

- 11.2. Delete the vacation directory by using the rm command with the -r option. Return to the student user's home directory.

```
[student@serverb my_bestseller]$ rm -r vacation
[student@serverb my_bestseller]$ ls
chapters editor
[student@serverb my_bestseller]$ cd
[student@serverb ~]$
```

12. Create a hard link to the ~/Documents/project\_plans/season2\_project\_plan.odf file called ~/Documents/backups/season2\_project\_plan.odf.back. A hard link protects against accidental deletion of the original file and keeps the backup file updated as you change the original file. Hint: If the ~/Documents/backups directory does not exist, then use the mkdir command to create it.

  - 12.1. Create a hard link to the ~/Documents/project\_plans/season2\_project\_plan.odf file called ~/Documents/backups/season2\_project\_plan.odf.back.

```
[student@serverb ~]$ mkdir ~/Documents/backups
[student@serverb ~]$ ln ~/Documents/project_plans/season2_project_plan.odf \
~/Documents/backups/season2_project_plan.odf.back
[student@serverb ~]$ ls -lR ~/Documents/
/home/student/Documents:
total 0
drwxr-xr-x. 2 student student 43 Mar  7 04:18 backups
drwxr-xr-x. 4 student student 36 Mar  7 04:16 my_bestseller
drwxr-xr-x. 2 student student 70 Mar  7 03:50 project_plans

/home/student/Documents/backups:
total 0
-rw-r--r--. 2 student student 0 Mar  7 03:50 season2_project_plan.odf.back

/home/student/Documents/my_bestseller:
total 0
drwxr-xr-x. 2 student student 118 Mar  7 04:07 chapters
drwxr-xr-x. 2 student student  62 Mar  7 04:06 editor

/home/student/Documents/my_bestseller/chapters:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter3.odf
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter4.odf
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter5.odf
```

```
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter6.odf

/home/student/Documents/my_bestseller/editor:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter1.odf
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter2.odf

/home/student/Documents/project_plans:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:50 season1_project_plan.odf
-rw-r--r--. 2 student student 0 Mar  7 03:50 season2_project_plan.odf
```

Notice that the link count is 2 for both `season2_project_plan.odf.back` and `season2_project_plan.odf` files.

12.2. Return to the `workstation` system 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 files-review
```

## 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 files-review
```

This concludes the section.

# Summary

---

- Files on a Linux system are organized into a single inverted tree of directories, a file-system hierarchy.
- Hard links and soft links are different ways for multiple file names to point to the same data.
- The Bash shell provides pattern matching, expansion, and substitution features to help you to run commands efficiently.



## Chapter 3

# Manage Local Users and Groups

### Goal

Create, manage, and delete local users and groups, and administer local password policies.

### Objectives

- Describe the purpose of users and groups on a Linux system.
- Switch to the superuser account to manage a Linux system, and grant other users superuser access through the sudo command.
- Create, modify, and delete local user accounts.
- Create, modify, and delete local group accounts.
- Set a password management policy for users, and manually lock and unlock user accounts.

### Sections

- Describe User and Group Concepts (and Quiz)
- Gain Superuser Access (and Guided Exercise)
- Manage Local User Accounts (and Guided Exercise)
- Manage Local Group Accounts (and Guided Exercise)
- Manage User Passwords (and Guided Exercise)

### Lab

Manage Local Users and Groups

# Describe User and Group Concepts

## Objectives

Describe the purpose of users and groups on a Linux system.

### What Is a User?

A *user* account provides security boundaries between different people and programs that can run commands.

Users have *user names* to identify them to human users and for ease of working. Internally, the system distinguishes user accounts by the unique identification number, the user ID or *UID*, which is assigned to them. In most scenarios, if a human uses a user account, then the system assigns a *secret password* for the user to prove that they are the authorized user to log in.

User accounts are fundamental to system security. Every process (running program) on the system runs as a particular user. Every file has a particular user as its owner. With file ownership, the system enforces access control for users of the files. The user that is associated with a running process determines the files and directories that are accessible to that process.

User accounts are of the following main types: the *superuser*, *system users*, and *regular users*.

- The *superuser* account administers the system. The superuser name is `root` and the account has a UID of 0. The superuser has full system access.
- The *system user* accounts are used by processes that provide supporting services. These processes, or *daemons*, usually do not need to run as the superuser. They are assigned non-privileged accounts to secure their files and other resources from each other and from regular users on the system. Users do not interactively log in with a system user account.
- Most users have *regular user* accounts for their day-to-day work. Like system users, regular users have limited access to the system.

Use the `id` command to show information about the currently logged-in user:

```
[user01@host ~]$ id  
uid=1000(user01) gid=1000(user01) groups=1000(user01)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

To view basic information about another user, pass the username to the `id` command as an argument:

```
[user01@host ~]$ id user02  
uid=1002(user02) gid=1001(user02) groups=1001(user02)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Use the `ls -l` command to view the owner of a file. Use the `ls -ld` command to view the owner of a directory, rather than the contents of that directory. In the following output, the third column shows the username.

```
[user01@host ~]$ ls -l mytextfile.txt
-rw-rw-r-- 1 user01 user01 0 Feb 5 11:10 mytextfile.txt
[user01@host]$ ls -ld Documents
drwxrwxr-x. 2 user01 user01 6 Feb 5 11:10 Documents
```

Use the `ps` command to view process information. The default is to show only processes in the current shell. Use the `ps` command `-a` option to view all processes with a terminal. Use the `ps` command `-u` option to view the user that is associated with a process. In the following output, the first column shows the username.

```
[user01@host ~]$ ps -au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root     1690  0.0  0.0 220984  1052  ttyp0  Ss+  22:43  0:00 /sbin/agetty -o -p --
          \u --keep-baud 1
user01   1769  0.0  0.1 377700  6844  ttyp2  Ssl+ 22:45  0:00 /usr/libexec/gdm-x-
          session --register-
user01   1773  1.3  1.3 528948  78356  ttyp2  S1+  22:45  0:03 /usr/libexec/Xorg vt2
          -displayfd 3 -au
user01   1800  0.0  0.3 521412  19824  ttyp2  S1+  22:45  0:00 /usr/libexec/gnome-
          session-binary
user01   3072  0.0  0.0 224152  5756  pts/1   Ss    22:48  0:00 -bash
user01   3122  0.0  0.0 225556  3652  pts/1   R+    22:49  0:00 ps -au
```

The output of the preceding command displays users by name, but internally the operating system uses UIDs to track users. The mapping of usernames to UIDs is defined in databases of account information. By default, systems use the `/etc/passwd` file to store information about local users.

Each line in the `/etc/passwd` file contains information about one user. The file is divided into seven colon-separated fields. An example of a line from `/etc/passwd` follows:

```
[user01@host ~]$ cat /etc/passwd
...output omitted...
user01:x:1000:1000:User One:/home/user01:/bin/bash
```

Consider each part of the code block, separated by a colon:

- **user01**: The username for this user.
- **x**: The user's encrypted password was historically stored here; this is now a placeholder.
- **1000**: The UID number for this user account.
- **1000**: The GID number for this user account's primary group. Groups are discussed later in this section.
- **User One**: A brief comment, description, or the real name for this user.
- **/home/user01**: The user's home directory, and the initial working directory when the login shell starts.
- **/bin/bash**: The default shell program for this user that runs at login. Some accounts use the `/sbin/nologin` shell to disallow interactive logins with that account.

## What Is a Group?

A group is a collection of users that need to share access to files and other system resources. Groups can be used to grant access to files to a set of users instead of to a single user.

Like users, groups have *group names* for easier recognition. Internally, the system distinguishes groups by the unique identification number, the *group ID* or *GID*, which is assigned to them. The mapping of group names to GIDs is defined in identity management databases of group account information. By default, systems use the `/etc/group` file to store information about local groups.

Each line in the `/etc/group` file contains information about one group. Each group entry is divided into four colon-separated fields. An example of a line from `/etc/group` follows:

```
[user01@host ~]$ cat /etc/group
...output omitted...
group01:x:10000:user01,user02,user03
```

Consider each part of the code block, separated by a colon:

- **group01**: Name for this group.
- **x**: Obsolete group password field; this is now a placeholder.
- **10000**: The GID number for this group (10000).
- **user01, user02, user03**: A list of users that are members of this group as a secondary group.

## Primary Groups and Secondary Groups

Every user has exactly one primary group. For local users, this group is listed by GID in the `/etc/passwd` file. The primary group owns files that the user creates.

When creating a regular user, a group is created with the same name as the user, to be the primary group for the user. The user is the only member of this *User Private Group*. This group membership design simplifies the management of file permissions, to have user groups segregated by default.

Users might also have *secondary groups*. Membership in secondary groups is stored in the `/etc/group` file. Users are granted access to files based on whether any of their groups have access, regardless of whether the groups are primary or secondary. For example, if the `user01` user has a `user01` primary group and `wheel` and `webadmin` secondary groups, then that user can read files that any of those three groups can read.

The `id` command can show group membership for a user. In the following example, the `user01` user has the `user01` group as their primary group (`gid`). The `groups` item lists all group memberships for this user, and the user also has the `wheel` and `group01` groups as secondary groups.

```
[user01@host ~]$ id
uid=1001(user01) gid=1003(user01) groups=1003(user01),10(wheel),10000(webadmin)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```



### References

`id(1)`, `passwd(5)`, and `group(5)` man pages

`info libc` (*GNU C Library Reference Manual*)

- Section 30: Users and Groups

(The `glibc-devel` package must be installed for this info node to be available.)

## ► Quiz

# Describe User and Group Concepts

Choose the correct answer to the following questions:

- ▶ 1. **Which item represents a number that identifies the user at the most fundamental level?**
  - a. Primary user
  - b. UID
  - c. GID
  - d. Username
  
- ▶ 2. **Which item represents the program that provides the user's command-line prompt?**
  - a. Primary shell
  - b. Home directory
  - c. Login shell
  - d. Command name
  
- ▶ 3. **Which item or file represents the location of the local group information?**
  - a. Home directory
  - b. /etc/passwd
  - c. /etc/GID
  - d. /etc/group
  
- ▶ 4. **Which item or file represents the location of the user's personal files?**
  - a. Home directory
  - b. Login shell
  - c. /etc/passwd
  - d. /etc/group
  
- ▶ 5. **Which item represents a number that identifies the group at the most fundamental level?**
  - a. Primary group
  - b. UID
  - c. GID
  - d. Groupid

► 6. Which item or file represents the location of the local user account information?

- a. Home directory
- b. /etc/passwd
- c. /etc/UID
- d. /etc/group

► 7. What is the fourth field of the /etc/passwd file?

- a. Home directory
- b. UID
- c. Login shell
- d. Primary group

## ► Solution

# Describe User and Group Concepts

Choose the correct answer to the following questions:

- ▶ 1. **Which item represents a number that identifies the user at the most fundamental level?**
  - a. Primary user
  - b. UID
  - c. GID
  - d. Username
  
- ▶ 2. **Which item represents the program that provides the user's command-line prompt?**
  - a. Primary shell
  - b. Home directory
  - c. Login shell
  - d. Command name
  
- ▶ 3. **Which item or file represents the location of the local group information?**
  - a. Home directory
  - b. /etc/passwd
  - c. /etc/GID
  - d. /etc/group
  
- ▶ 4. **Which item or file represents the location of the user's personal files?**
  - a. Home directory
  - b. Login shell
  - c. /etc/passwd
  - d. /etc/group
  
- ▶ 5. **Which item represents a number that identifies the group at the most fundamental level?**
  - a. Primary group
  - b. UID
  - c. GID
  - d. Groupid

► 6. Which item or file represents the location of the local user account information?

- a. Home directory
- b. /etc/passwd
- c. /etc/UID
- d. /etc/group

► 7. What is the fourth field of the /etc/passwd file?

- a. Home directory
- b. UID
- c. Login shell
- d. Primary group

# Gain Superuser Access

## Objectives

Switch to the superuser account to manage a Linux system, and grant other users superuser access through the sudo command.

## The Superuser

Most operating systems have a *superuser* that has all power over the system. In Red Hat Enterprise Linux, it is the `root` user. This user has the power to override normal privileges on the file system, and you can use it to manage and administer the system. For tasks such as installing or removing software, and to manage system files and directories, users must escalate their privileges to the `root` user.

Usually, only the `root` user among normal users can control most devices, but some exceptions apply. For example, normal users can control removable devices, such as USB devices. Thus, normal users can add and remove files and otherwise manage a removable device, but only `root` can manage hard drives by default.

This unlimited privilege, however, comes with responsibility. The `root` user has unlimited power to damage the system: remove files and directories, remove user accounts, add back doors, and so on. If the `root` user account is compromised, then the system is in danger and you might lose administrative control. Red Hat encourages system administrators to log in always as a normal user and escalate privileges to `root` only when needed.

The `root` account on Linux is roughly equivalent to the local `Administrator` account on Microsoft Windows. In Linux, most system administrators log in to the system as an unprivileged user and use various tools to temporarily gain `root` privileges.



### Warning

Microsoft Windows users might be familiar with the practice of logging in as the local `Administrator` user to perform system administrator duties. Today, this practice is not recommended; users obtain privileges to perform administration by memberships in the `Administrators` group. Similarly in RHEL, Red Hat recommends that system administrators never log in directly as `root`. Instead, system administrators log in as a normal user and use mechanisms (`su`, `sudo`, or `PolicyKit`, for example) to temporarily gain superuser privileges.

When logged in as `root`, the entire desktop environment unnecessarily runs with administrative privileges. A security vulnerability that might normally compromise only a normal user account can potentially compromise the entire system.

## Switch User Accounts

With the `su` command, users can switch to a different user account. If you run the `su` command from a regular user account with another user account as a parameter, then you must provide the password of the account to switch to. When the `root` user runs the `su` command, you do not need to enter the user's password.

This example uses the `su` command from the `user01` account to switch to the `user02` account:

```
[user01@host ~]$ su - user02
Password: user02_password
[user02@host ~]$
```

If you omit the user name, then the `su` or `su -` command attempts to switch to `root` by default:

```
[user01@host ~]$ su -
Password: root_password
[root@host ~]#
```

The command `su` starts a *non-login shell*, while the command `su -` (with the dash option) starts a *login shell*. The main distinction between the two commands is that `su -` sets up the shell environment as if it is a new login as that user, while `su` starts a shell as that user, but uses the original user's environment settings.

In most cases, administrators should run `su -` to get a shell with the target user's normal environment settings. For more information, see the `bash(1)` man page.



### Note

The most frequent use for the `su` command is to get a command-line interface (shell prompt) that runs as another user, typically `root`. However, you can use it with the `su` command `-c` option to run an arbitrary program as another user. This behavior is similar to the Windows `runas` utility. Run `info su` to view more details.

## Run Commands with Sudo

For security reasons, in some cases system administrators configure the `root` user not to have a valid password. Thus, users cannot log in to the system as `root` directly with a password. Moreover, you cannot use `su` to get an interactive shell. In this case, you can use the `sudo` command to get `root` access.

Unlike the `su` command, `sudo` normally requires users to enter their own password for authentication, not the password of the user account they are trying to access. That is, users who use the `sudo` command to run commands as `root` do not need to know the `root` password. Instead, they use their own passwords to authenticate access.

The next table summarizes the differences between the `su`, `su -`, and `sudo` commands:

	<b>su</b>	<b>su -</b>	<b>sudo</b>
Become new user	Yes	Yes	Per escalated command
Environment	Current user's	New user's	Current user's
Password required	New user's	New user's	Current user's
Privileges	Same as new user	Same as new user	Defined by configuration
Activity logged	su command only	su command only	Per escalated command

Additionally, you can configure the `sudo` command to allow specific users to run any command as some other user, or only some commands as that user. For example, if you configure the `sudo` command to allow the `user01` user to run the `usermod` command as `root`, then you can run the following command to lock or unlock a user account:

```
[user01@host ~]$ sudo usermod -L user02
[sudo] password for user01: user01_password
[user01@host ~]$ su - user02
Password: user02_password
su: Authentication failure
[user01@host ~]$
```

If a user tries to run a command as another user, and the `sudo` configuration does not permit it, then bash blocks the command, logs the attempt, and sends by default an email to the `root` user.

```
[user02@host ~]$ sudo tail /var/log/secure
[sudo] password for user02: user02_password
user02 is not in the sudoers file. This incident will be reported.
[user02@host ~]$
```

Another benefit of `sudo` is to log by default all the executed commands to `/var/log/secure`.

```
[user01@host ~]$ sudo tail /var/log/secure
...output omitted...
Mar  9 20:45:46 host sudo[2577]: user01 : TTY=pts/0 ; PWD=/home/user01 ;
USER=root ; COMMAND=/sbin/usermod -L user02
...output omitted...
```

In Red Hat Enterprise Linux 7 and later versions, all members of the `wheel` group can use `sudo` to run commands as any user, including `root`, by using their own password.



### Warning

Historically, UNIX systems use membership in the `wheel` group to grant or control superuser access. RHEL 6 and earlier versions do not grant the `wheel` group any special privileges by default. System administrators who have previously used this group for a non-standard purpose should update a previous configuration, to avoid unexpected and unauthorized users obtaining administrative access on RHEL 7 and later systems.

## Get an Interactive Root Shell with Sudo

To access the `root` account with `sudo`, use the `sudo -i` command. This command switches to the `root` account and runs that user's default shell (usually `bash`) and associated interactive login scripts. To run the shell without the interactive scripts, use the `sudo -s` command.

For example, an administrator can get an interactive shell as `root` on an AWS Elastic Cloud Computing (EC2) instance by using SSH public-key authentication to log in as the `ec2-user` normal user, and then run the `sudo -i` command to access the `root` user's shell.

```
[ec2-user@host ~]$ sudo -i
[sudo] password for ec2-user: ec2-user_password
[root@host ~]#
```

## Configure sudo

The `/etc/sudoers` file is the main configuration file for the `sudo` command. To avoid problems if multiple administrators try to edit the file at the same time, you can edit it only with the special `visudo` command. The `visudo` editor also validates the file, to ensure no syntax errors.

For example, the following line from the `/etc/sudoers` file enables `sudo` access for `wheel` group members:

```
%wheel      ALL=(ALL:ALL)      ALL
```

- The `%wheel` string is the user or group that the rule applies to. The `%` symbol before the word `wheel` specifies a group.
- The `ALL=(ALL:ALL)` command specifies that on any host with this file (the first `ALL`), users in the `wheel` group can run commands as any other user (the second `ALL`) and any other group (the third `ALL`) on the system.
- The final `ALL` command specifies that users in the `wheel` group can run any command.

By default, the `/etc/sudoers` file also includes the contents of any files in the `/etc/sudoers.d` directory as part of the configuration file. With this hierarchy, you can add `sudo` access for a user by putting an appropriate file in that directory.



### Note

Placing configuration files under the `/etc/sudoers.d` directory is convenient. You can enable or disable `sudo` access by copying a file into the directory or removing it from the directory.

In this course, you create and remove files in the `/etc/sudoers.d` directory to configure `sudo` access for users and groups.

To enable full `sudo` access for the `user01` user, you can create the `/etc/sudoers.d/user01` file with the following content:

```
user01      ALL=(ALL)      ALL
```

To enable full `sudo` access for the `group01` group, you can create the `/etc/sudoers.d/group01` file with the following content:

```
%group01      ALL=(ALL)      ALL
```

To enable users in the `games` group to run the `id` command as the `operator` user, you can create the `/etc/sudoers.d/games` file with the following content:

```
%games  ALL=(operator) /bin/id
```

You can also set up sudo to allow a user to run commands as another user without entering their password, by using the NOPASSWD: ALL command:

```
ansible          ALL=(ALL)      NOPASSWD: ALL
```

While obvious security risks apply to granting this level of access to a user or group, system administrators frequently use this approach with cloud instances, virtual machines, and provisioning systems for configuring servers. You must carefully protect the account with this access and require SSH public-key authentication for a user on a remote system to access it at all.

For example, the official Amazon Machine Image (AMI) for Red Hat Enterprise Linux in the Amazon Web Services Marketplace ships with the `root` and the `ec2-user` passwords locked. The `ec2-user` account is set up to allow remote interactive access through SSH public-key authentication. The user `ec2-user` can also run any command as `root` without a password because the last line of the AMI's `/etc/sudoers` file is set up as follows:

```
ec2-user          ALL=(ALL)      NOPASSWD: ALL
```

You can re-enable the requirement to enter a password for sudo or introduce other changes to tighten security as part of the system configuration.



## References

`su(1)`, `sudo(8)`, `visudo(8)`, and `sudoers(5)` man pages

`info libc persona` (*GNU C Library Reference Manual*)

- Section 30.2: The Persona of a Process

(The `glibc-doc` package must be installed for this info node to be available.)

## ► Guided Exercise

# Gain Superuser Access

In this exercise, you practice switching to the `root` account and running commands as `root`.

### Outcomes

- Use the `sudo` command to switch to the `root` user and access the interactive shell as `root` without knowing the password of the superuser.
- Explain how the `su` and `sudo` commands affect the shell environment through running or not running the login scripts.
- Use the `sudo` command to run other commands as the `root` user.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start users-superuser
```

### Instructions

- 1. From `workstation`, open an SSH session to `servera` as the `student` user.

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

- 2. Explore the shell environment of the `student` user. View the current user and group information and display the current working directory. Also view the environment variables that specify the user's home directory and the locations of the user's executable files.

2.1. Run `id` to view the current user and group information.

```
[student@servera ~]$ id  
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel)  
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

2.2. Run `pwd` to display the current working directory.

```
[student@servera ~]$ pwd  
/home/student
```

2.3. Print the values of the `HOME` and `PATH` variables to determine the home directory and user executables' path, respectively.

```
[student@servera ~]$ echo $HOME
/home/student
[student@servera ~]$ echo $PATH
/home/student/.local/bin:/home/student/bin:/usr/local/bin:/usr/bin:/usr/local/
sbin:/usr/sbin
```

- 3. Switch to the `root` user in a non-login shell and explore the new shell environment.

- 3.1. Run the `sudo su` command at the shell prompt to become the `root` user.

```
[student@servera ~]$ sudo su
[sudo] password for student: student
[root@servera student]#
```

- 3.2. Run `id` to view the current user and group information.

```
[root@servera student]# id
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 3.3. Run `pwd` to display the current working directory.

```
[root@servera student]# pwd
/home/student
```

- 3.4. Print the values of the `HOME` and `PATH` variables to determine the home directory and user executables' path, respectively.

```
[root@servera student]# echo $HOME
/root
[root@servera student]# echo $PATH
/root/.local/bin:/root/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/
local/bin
```

When you use the `su` command to become the `root` user, you do not keep the current path of the `student` user. As you can see in the next step, the path is not the `root` user path either.

What happened? The difference is that you do not run `su` directly. Instead, you run the `su` command as the `root` user by using `sudo` because you do not have the password of the superuser. The `sudo` command overrides the `PATH` variable from the environment for security reasons. Any command that runs after the initial override can still update the `PATH` variable, as you can see in the following steps.

- 3.5. Exit the `root` user's shell to return to the `student` user's shell.

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

- 4. Switch to the `root` user in a login shell and explore the new shell environment.

- 4.1. Run the `sudo su -` command at the shell prompt to become the `root` user.

The `sudo` command might or might not prompt you for the `student` password, depending on the time-out period of `sudo`. The default time-out period is five minutes. If you authenticated to `sudo` within the last five minutes, then the `sudo` command does not prompt you for the password. If more than five minutes elapsed since you authenticated to `sudo`, then you must enter `student` as the password for authentication to `sudo`.

```
[student@servera ~]$ sudo su -
[root@servera ~]#
```

Notice the difference in the shell prompt compared to that of `sudo su` in the preceding step.

- 4.2. Run `id` to view the current user and group information.

```
[root@servera ~]# id
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 4.3. Run `pwd` to display the current working directory.

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

- 4.4. Print the values of the `HOME` and `PATH` variables to determine the home directory and the user executables' path, respectively.

```
[root@servera ~]# echo $HOME
/root
[root@servera ~]# echo $PATH
/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

As in the preceding step, after the `sudo` command resets the `PATH` variable from the settings in the `student` user's shell environment, the `su -` command runs the shell login scripts for `root` and sets the `PATH` variable to yet another value. The `su` command without the dash (-) option does not have the same behavior.

- 4.5. Exit the `root` user's shell to return to the `student` user's shell.

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

- 5. Verify that the `operator1` user can run any command as any user by using the `sudo` command.

```
[student@servera ~]$ sudo cat /etc/sudoers.d/operator1
operator1 ALL=(ALL) ALL
```

- 6. Become the `operator1` user and view the contents of the `/var/log/messages` file. Copy the `/etc/motd` file to `/etc/motdOLD`. Remove the `/etc/motdOLD` file. As these

operations require administrative rights, use the `sudo` command to run those commands as the superuser. Do not switch to root by using `sudo su` or `sudo su -`. Use `redhat` as the password of `operator1` user.

- 6.1. Switch to the `operator1` user.

```
[student@servera ~]$ su - operator1
Password: redhat
[operator1@servera ~]$
```

- 6.2. Attempt to view the last five lines of `/var/log/messages` without using `sudo`. It should fail.

```
[operator1@servera ~]$ tail -5 /var/log/messages
tail: cannot open '/var/log/messages' for reading: Permission denied
```

- 6.3. Attempt to view the last five lines of `/var/log/messages` by using `sudo`. It should succeed. Example result:Return to the workstation system as the `student` user.

```
[operator1@servera ~]$ sudo tail -5 /var/log/messages
[sudo] password for operator1: redhat
Mar 9 15:53:36 servera su[2304]: FAILED SU (to operator1) student on pts/1
Mar 9 15:53:51 servera su[2307]: FAILED SU (to operator1) student on pts/1
Mar 9 15:53:58 servera su[2310]: FAILED SU (to operator1) student on pts/1
Mar 9 15:54:12 servera su[2322]: (to operator1) student on pts/1
Mar 9 15:54:25 servera su[2353]: (to operator1) student on pts/1
```



### Note

The preceding output might differ on your system.

- 6.4. Attempt to copy `/etc/motd` as `/etc/motdOLD` without using `sudo`. It should fail.

```
[operator1@servera ~]$ cp /etc/motd /etc/motdOLD
cp: cannot create regular file '/etc/motdOLD': Permission denied
```

- 6.5. Attempt to copy `/etc/motd` as `/etc/motdOLD` by using `sudo`. It should succeed.

```
[operator1@servera ~]$ sudo cp /etc/motd /etc/motdOLD
[operator1@servera ~]$
```

- 6.6. Attempt to delete `/etc/motdOLD` without using `sudo`. It should fail.

```
[operator1@servera ~]$ rm /etc/motdOLD
rm: remove write-protected regular empty file '/etc/motdOLD'? y
rm: cannot remove '/etc/motdOLD': Permission denied
[operator1@servera ~]$
```

- 6.7. Attempt to delete `/etc/motdOLD` by using `sudo`. It should succeed.

```
[operator1@servera ~]$ sudo rm /etc/motdOLD  
[operator1@servera ~]$
```

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

```
[operator1@servera ~]$ exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

## 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 users-superuser
```

This concludes the section.

# Manage Local User Accounts

---

## Objectives

Create, modify, and delete local user accounts.

## Manage Local Users

You can use different command-line tools to manage local user accounts. This section reviews some of the most important ones.

### Create Users from the Command Line

The `useradd` *username* command creates a user called *username*. It sets up the user's home directory and account information, and creates a private group for the user called *username*. At this point, a valid password is not set for the account, and the user cannot log in until a password is set.

The `useradd --help` command displays the basic options to override the defaults. In most cases, you can use the same options with the `usermod` command to modify an existing user.

The `/etc/login.defs` file sets some of the default options for user accounts, such as the range of valid UID numbers and default password aging rules. The values in this file affect only newly created user accounts. A change to this file does not affect existing users.

In Red Hat Enterprise Linux 9, the `useradd` command assigns new users the first free UID that is greater than or equal to 1000, unless you explicitly specify one by using the `-u` option.

### Modify Existing Users from the Command Line

The `usermod --help` command displays the basic options to modify an account. Some common options are as follows:

<b>usermod options:</b>	<b>Usage</b>
<code>-a, --append</code>	Use it with the <code>-G</code> option to add the secondary groups to the user's current set of group memberships instead of replacing the set of secondary groups with a new set.
<code>-c, --comment COMMENT</code>	Add the <code>COMMENT</code> text to the comment field.
<code>-d, --home HOME_DIR</code>	Specify a home directory for the user account.
<code>-g, --gid GROUP</code>	Specify the primary group for the user account.
<code>-G, --groups GROUPS</code>	Specify a comma-separated list of secondary groups for the user account.

<b>usermod options:</b>	<b>Usage</b>
<b>-L, --lock</b>	Lock the user account.
<b>-m, --move-home</b>	Move the user's home directory to a new location. You must use it with the <b>-d</b> option.
<b>-s, --shell SHELL</b>	Specify a particular login shell for the user account.
<b>-U, --unlock</b>	Unlock the user account.

## Delete Users from the Command Line

The `userdel username` command removes the `username` user from `/etc/passwd`, but leaves the user's home directory intact. The `userdel -r username` command removes the user from `/etc/passwd` and deletes the user's home directory.



### Warning

When you remove a user without specifying the `userdel -r` option, the user's files are now owned by an unassigned UID. If you create a user and that user is assigned the deleted user's UID, then the new account will own those files, which is a security risk. Typically, organization security policies disallow deleting user accounts, and instead lock them from being used, to avoid this scenario.

The following example demonstrates how this can lead to information leakage:

```
[root@host ~]# useradd user01
[root@host ~]# ls -l /home
drwx----- 3 user01 user01    74 Mar  4 15:22 user01
[root@host ~]# userdel user01
[root@host ~]# ls -l /home
drwx----- 3      1000      1000    74 Mar  4 15:22 user01
[root@host ~]# useradd -u 1000 user02
[root@host ~]# ls -l /home
drwx----- 3 user02      user02    74 Mar  4 15:23 user02
drwx----- 3 user02      user02    74 Mar  4 15:22 user01
```

Notice that `user02` now owns all files that `user01` previously owned. The root user can use the `find / -nouser -o -nogroup` command to find all unowned files and directories.

## Set Passwords from the Command Line

The `passwd username` command sets the initial password or changes the existing password for `username` user. The root user can set a password to any value. The terminal displays a message if the password does not meet the minimum recommended criteria, but then you can retype the new password and the `passwd` command updates it successfully.

```
[root@host ~]# passwd user01
Changing password for user user01.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
[root@host ~]#
```

A regular user must choose a password at least eight characters long. Do not use a dictionary word, the username, or the previous password.

## UID Ranges

Red Hat Enterprise Linux uses specific UID numbers and ranges of numbers for specific purposes.

- **UID 0** : The superuser (`root`) account UID.
- **UID 1-200** : System account UIDs statically assigned to system processes.
- **UID 201-999** : UIDs assigned to system processes that do not own files on this system. Software that requires an unprivileged UID is dynamically assigned UID from this available pool.
- **UID 1000+** : The UID range to assign to regular, unprivileged users.



### Note

RHEL 6 and earlier versions use UIDs in the range 1-499 for system users and UIDs higher than 500 for regular users. You can change the `useradd` and `groupadd` default ranges in the `/etc/login.defs` file.



### References

`useradd(8)`, `usermod(8)`, and `userdel(8)` man pages

## ► Guided Exercise

# Manage Local User Accounts

In this exercise, you create several users on your system and set passwords for those users.

### Outcomes

- Configure a Linux system with additional user accounts.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start users-user
```

### Instructions

- 1. From workstation, open an SSH session to servera as student user and switch to root user.

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

- 2. Create the operator1 user and confirm that it exists in the system.

```
[root@servera ~]# useradd operator1
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002::/home/operator1:/bin/bash
```

- 3. Set the password for operator1 to redhat.

```
[root@servera ~]# passwd operator1
Changing password for user operator1.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 4. Create the additional operator2 and operator3 users. Set their passwords to redhat.

- 4.1. Add the operator2 user. Set the password for operator2 to redhat.

```
[root@servera ~]# useradd operator2
[root@servera ~]# passwd operator2
Changing password for user operator2.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 4.2. Add the operator3 user. Set the password for operator3 to redhat.

```
[root@servera ~]# useradd operator3
[root@servera ~]# passwd operator3
Changing password for user operator3.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 5. Update the operator1 and operator2 user accounts to include the Operator One and Operator Two comments, respectively. Verify that the comments exist for the user accounts.

- 5.1. Run the usermod -c command to update the comments of the operator1 user account.

```
[root@servera ~]# usermod -c "Operator One" operator1
```

- 5.2. Run the usermod -c command to update the comments of the operator2 user account.

```
[root@servera ~]# usermod -c "Operator Two" operator2
```

- 5.3. View the /etc/passwd file to confirm that the comments for each of the operator1 and operator2 users exist.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002:Operator One:/home/operator1:/bin/bash
operator2:x:1003:1003:Operator Two:/home/operator2:/bin/bash
operator3:x:1004:1004::/home/operator3:/bin/bash
```

- 6. Delete the operator3 user along with any personal data of the user. Confirm that the operator3 does not exist.

- 6.1. Remove the operator3 user from the system.

```
[root@servera ~]# userdel -r operator3
```

- 6.2. Confirm that the operator3 user does not exist.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002:Operator One:/home/operator1:/bin/bash
operator2:x:1003:1003:Operator Two:/home/operator2:/bin/bash
```

Notice that the preceding output does not display the user account information of operator3.

- 6.3. Confirm that the operator3 user home folder does not exist.

```
[root@servera ~]# ls -l /home
total 0
drwx----- 4 devops    devops    90 Mar  3 09:59 devops
drwx----- 2 operator1 operator1 62 Mar  9 10:19 operator1
drwx----- 2 operator2 operator2 62 Mar  9 10:19 operator2
drwx----- 3 student   student   95 Mar  3 09:49 student
```

- 6.4. Exit the root user's shell to return to the student user's shell.

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

- 6.5. Log off from servera.

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

## 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 users-user
```

This concludes the section.

# Manage Local Group Accounts

## Objectives

Create, modify, and delete local group accounts.

## Manage Local Groups

Several command-line tools facilitate group management. While you can use the **Users** GUI utility to manage groups, Red Hat recommends to use command-line tools.

### Create Groups from the Command Line

The **groupadd** command creates groups. Without options, the **groupadd** command uses the next available GID from the range specified by the **GID\_MIN** and **GID\_MAX** variables in the **/etc/login.defs** file. By default, the command assigns a GID value greater than any other existing GIDs, even if a lower value becomes available.

The **groupadd** command **-g** option specifies a particular GID for the group to use.

```
[root@host ~]# groupadd -g 10000 group01
[root@host ~]# tail /etc/group
...output omitted...
group01:x:10000:
```



#### Note

Because of the automatic creation of user private groups (GID 1000+), some administrators set aside a separate range of GIDs for creating secondary groups for other purposes. However, this extra management is unnecessary, because a user's UID and primary GID do not need to be the same number.

The **groupadd** command **-r** option creates system groups. As with normal groups, system groups use a GID from the range of listed valid system GIDs in the **/etc/login.defs** file. The **SYS\_GID\_MIN** and **SYS\_GID\_MAX** configuration items in the **/etc/login.defs** file define the range of system GIDs.

```
[root@host ~]# groupadd -r group02
[root@host ~]# tail /etc/group
...output omitted...
group01:x:10000:
group02:x:988:
```

### Modify Existing Groups from the Command Line

The **groupmod** command changes the properties of an existing group. The **groupmod** command **-n** option specifies a new name for the group.

```
[root@host ~]# groupmod -n group0022 group02
[root@host ~]# tail /etc/group
...output omitted...
group0022:x:988:
```

Notice that the group name updates to group0022 from group02. The groupmod command -g option specifies a new GID.

```
[root@host ~]# groupmod -g 20000 group0022
[root@host ~]# tail /etc/group
...output omitted...
group0022:x:20000:
```

Notice that the GID changes to 20000 from 988.

## Delete Groups from the Command Line

The groupdel command removes groups.

```
[root@host ~]# groupdel group0022
```



### Note

You cannot remove a group if it is the primary group of an existing user. Similar to using the userdel command, check first to ensure to locate files that the group owns.

## Change Group Membership from the Command Line

The membership of a group is controlled with user management. Use the usermod -g command to change a user's primary group.

```
[root@host ~]# id user02
uid=1006(user02) gid=1008(user02) groups=1008(user02)
[root@host ~]# usermod -g group01 user02
[root@host ~]# id user02
uid=1006(user02) gid=10000(group01) groups=10000(group01)
```

Use the usermod -aG command to add a user to a secondary group.

```
[root@host ~]# id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03)
[root@host ~]# usermod -aG group01 user03
[root@host ~]# id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
```

**Important**

The usermod command -a option enables the *append* mode. Without the -a option, the command removes the user from any of their current secondary groups that are not included in the -G option's list.

## Compare Primary and Secondary Group Membership

A user's primary group is the group that is viewed on the user's account in the /etc/passwd file. A user can only belong to one primary group at a time.

A user's secondary groups are the additional groups configured for the user and viewed on the user's entry in the /etc/group file. A user can belong to as many secondary groups as is necessary to implement file access and permission effectively.

For the purpose of configuring group-based file permissions, there is no difference between a user's primary and secondary groups. If the user belongs to any group that has been assigned access to specific files, then that user has access to those files.

The only distinction between a user's primary and secondary memberships is when a user creates a file. New files must have a user owner and a group owner, which is assigned as the file is created. The user's primary group is used for the new file's group ownership, unless overridden by command options.

## Temporarily Change Your Primary Group

Only a user's primary group is used for new file creation attributes. However, you can temporarily switch your primary group to another group, but you can only choose from secondary groups to which you already belong. You might switch if you are about to create a number of new files, manually or scripted, and want them to have a different group assigned as owner as they are being created.

Use the newgrp command to switch your primary group, in this shell session. You can switch between any primary or secondary group to which you belong, but only one at a time can be primary. Your primary group will return to the default if you log out and log in again. In this example, the group called group01 temporarily becomes this user's primary group.

```
[user03@host ~]# id
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
[user03@host ~]$ newgrp group01
[user03@host ~]# id
uid=1007(user03) gid=10000(group01) groups=1009(user03),10000(group01)
```

**References**

group(5), groupadd(8), groupdel(8), and usermod(8) man pages

## ► Guided Exercise

# Manage Local Group Accounts

In this exercise, you create groups, use them as secondary groups for some users without changing those users' primary groups, and configure one of the groups with sudo access to run commands as root.

## Outcomes

- Create groups and use them as secondary groups.
- Configure sudo access for a group.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command creates the necessary user accounts to set up the environment correctly.

```
[student@workstation ~]$ lab start users-group
```

## Instructions

- 1. From workstation, open an SSH session to servera as the student user and switch to root user.

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

- 2. Create the operators secondary group with a GID of 30000.

```
[root@servera ~]# groupadd -g 30000 operators
```

- 3. Create the admin secondary group without specifying a GID.

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

- 4. Verify that both the operators and admin secondary groups exist.

```
[root@servera ~]# tail /etc/group  
...output omitted...  
operators:x:30000:  
admin:x:30001:
```

- 5. Ensure that the operator1, operator2, and operator3 users belong to the operators group.

5.1. Add the operator1, operator2, and operator3 users to the operators group.

```
[root@servera ~]# usermod -aG operators operator1
[root@servera ~]# usermod -aG operators operator2
[root@servera ~]# usermod -aG operators operator3
```

5.2. Confirm that the users are in the group.

```
[root@servera ~]# id operator1
uid=1002(operator1) gid=1002(operator1) groups=1002(operator1),30000(operators)
[root@servera ~]# id operator2
uid=1003(operator2) gid=1003(operator2) groups=1003(operator2),30000(operators)
[root@servera ~]# id operator3
uid=1004(operator3) gid=1004(operator3) groups=1004(operator3),30000(operators)
```

- 6. Ensure that the sysadmin1, sysadmin2 and sysadmin3 users belong to the admin group. Enable administrative rights for all the admin group members. Verify that any member of the admin group can run administrative commands.

6.1. Add the sysadmin1, sysadmin2, and sysadmin3 users to the admin group.

```
[root@servera ~]# usermod -aG admin sysadmin1
[root@servera ~]# usermod -aG admin sysadmin2
[root@servera ~]# usermod -aG admin sysadmin3
```

6.2. Confirm that the users are in the group.

```
[root@servera ~]# id sysadmin1
uid=1005(sysadmin1) gid=1005(sysadmin1) groups=1005(sysadmin1),30001(admin)
[root@servera ~]# id sysadmin2
uid=1006(sysadmin2) gid=1006(sysadmin2) groups=1006(sysadmin2),30001(admin)
[root@servera ~]# id sysadmin3
uid=1007(sysadmin3) gid=1007(sysadmin3) groups=1007(sysadmin3),30001(admin)
```

6.3. Examine the /etc/group file to verify the secondary group memberships.

```
[root@servera ~]# tail /etc/group
...output omitted...
operators:x:30000:operator1,operator2,operator3
admin:x:30001:sysadmin1,sysadmin2,sysadmin3
```

6.4. Create the /etc/sudoers.d/admin file so that the members of the admin group have full administrative privileges.

```
[root@servera ~]# echo "%admin ALL=(ALL) ALL" >> /etc/sudoers.d/admin
```

6.5. Switch to the sysadmin1 user (a member of the admin group) and verify that you can run a sudo command.

```
[root@servera ~]# su - sysadmin1
[sysadmin1@servera ~]$ sudo cat /etc/sudoers.d/admin
[sudo] password for sysadmin1: redhat
%admin  ALL=(ALL)  ALL
```

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

```
[sysadmin1@servera ~]$ exit
logout
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

## 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 users-group
```

This concludes the section.

# Manage User Passwords

---

## Objectives

Set a password management policy for users, and manually lock and unlock user accounts.

## Shadow Passwords and Password Policy

Originally, encrypted passwords were stored in the world-readable `/etc/passwd` file. This was considered adequate until dictionary attacks on encrypted passwords became common. The encrypted passwords were moved to the `/etc/shadow` file, which only the `root` user can read.

Like the `/etc/passwd` file, each user has an entry with in the `/etc/shadow` file. An example entry from the `/etc/shadow` file has nine colon-separated fields:

```
[root@host ~]# cat /etc/shadow
...output omitted...
user03:$6$CSsXsd3rwghsdfarf:17933:0:99999:7:2:18113:
```

Each field of this code block is separated by a colon:

- **user03** : Name of the user account.
- **\$6\$CSsXsd3rwghsdfarf** : The encrypted password of the user.
- **17933** : The days from the epoch when the password was last changed, where the epoch is `1970-01-01` in the UTC time zone.
- **0** : The minimum days since the last password change before the user can change it again.
- **99999** : The maximum days without a password change before the password expires. An empty field means that the password never expires.
- **7** : The number of days ahead to warn the user that their password will expire.
- **2** : The number of days without activity, starting with the day the password expired, before the account is automatically locked.
- **18113** : The day when the account expires in days since the epoch. An empty field means that the account never expires.
- The last field is typically empty and reserved for future use.

## Format of an Encrypted Password

The encrypted password field stores three pieces of information: the hashing algorithm in use, the *salt*, and the encrypted hash. Each piece of information is delimited by the dollar (\$) character.

```
$6$CSsXcYG1L/4ZfHr/$2W6evvJahUfzfHpc9X.45Jc6H30E
```

- **6** : The hashing algorithm in use for this password. A 6 indicates a SHA-512 hash, the RHEL 9 default, a 1 indicates MD5, and a 5 indicates SHA-256.
- **CSsXcYG1L/4ZfHr/** : The salt in use to encrypt the password; originally chosen at random.
- **2W6evvJahUfzfHpc9X.45Jc6H30E** : The encrypted hash of the user's password; combining the salt and the unencrypted password and then encrypting to generate the password hash.

The primary reason to combine a salt with the password is to defend against attacks that use pre-computed lists of password hashes. Adding salts changes the resulting hashes, which makes the

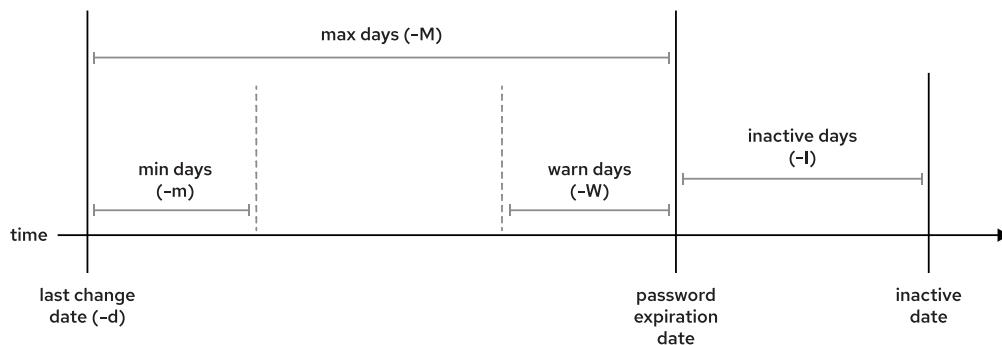
pre-computed list useless. If an attacker obtains a copy of an /etc/shadow file that uses salts, then they need to guess passwords with brute force, which requires more time and effort.

## Password Verification

When a user tries to log in, the system looks up the entry for the user in the /etc/shadow file, combines the salt for the user with the unencrypted typed password, and encrypts the combination of the salt and unencrypted password with the specified hashing algorithm. If the result matches the encrypted hash, then the user typed the right password. If the result does not match the encrypted hash, then the user typed the wrong password and the login attempt fails. This method allows the system to determine whether the user typed the correct password without storing that password in a usable form for logging in.

## Configure Password Aging

The following diagram shows the relevant password aging parameters, which can be adjusted by using the chage command to implement a password aging policy. Notice that the command name is chage which stands for "change age", and it should not be confused with the word "change".



**Figure 3.1: Password aging parameters**

The following example demonstrates the chage command to change the password policy of the sysadmin05 user. The command defines a minimum age (-m) of zero days, a maximum age (-M) of 90 days, a warning period (-W) of 7 days, and an inactivity period (-I) of 14 days.

```
[root@host ~]# chage -m 0 -M 90 -W 7 -I 14 sysadmin05
```

Assume that you manage the user password policies on a Red Hat server. The clouddadmin10 user is new in the system and you want to set a custom password aging policy. You want to set the account expiration 30 days from today, so you use the following commands:

```
[root@host ~]# date +%F ①
2022-03-10
[root@host ~]# date -d "+30 days" +%F ②
2022-04-09
[root@host ~]# chage -E $(date -d "+30 days" +%F) clouddadmin10 ③
[root@host ~]# chage -l clouddadmin10 | grep "Account expires" ④
Account expires      : Apr 09, 2022
```

- ① Use the date command to get the current date.

- ② Use the `date` command to get the date 30 days from now.
- ③ Use the `chage` command -E option to change the expiration date for the `cloudadmin10` user.
- ④ Use the `chage` command -l option to display the password aging policy for the `cloudadmin10` user.

After a few days, you notice in the `/var/log/secure` log file that the `cloudadmin10` user has a strange behavior. The user tried to use `sudo` to interact with files that belong to other users. You suspect that the user might have left an `ssh` session open while working in another machine. You want the `cloudadmin10` user to change the password on the next login, so you use the following command.

```
[root@host ~]# chage -d 0 cloudadmin10
```

The next time the `cloudadmin10` user logs in, the user is prompted to change the password.



### Note

The `date` command can calculate a future date. The `-u` option reports the time in UTC.

```
[user01@host ~]$ date -d "+45 days" -u
Thu May 23 17:01:20 UTC 2019
```

You can change the default password aging configuration in the `/etc/login.defs` file. The `PASS_MAX_DAYS` and `PASS_MIN_DAYS` options set the default maximum and minimum age of the password respectively. The `PASS_WARN_AGE` sets the default warning period of the password. Any change in the default password aging policies affects users that are created after the change. The existing users continue to use the old password aging settings rather than the new ones. For more information about the `/etc/login.defs` file, refer to the *Red Hat Security: Linux in Physical, Virtual, and Cloud* (RH415) course and the `login.defs(5)` man page.

## Restrict Access

You can use the `usermod` command to modify account expiration for a user. For example, the `usermod` command -L option locks a user account and the user cannot log in to the system.

```
[root@host ~]# usermod -L sysadmin03
[user01@host ~]$ su - sysadmin03
Password: redhat
su: Authentication failure
```

If a user leaves the company on a certain date, then you can lock and expire the account with a single `usermod` command. The date must be the number of days since `1970-01-01`, or use the `YYYY-MM-DD` format. In the following example, the `usermod` command locks and expires the `cloudadmin10` user at `2022-08-14`.

```
[root@host ~]# usermod -L -e 2022-08-14 cloudadmin10
```

When you lock an account, you prevent the user from authenticating with a password to the system. This method is recommended to prevent access to an account by a former employee of the company. Use the `usermod` command `-U` option to enable the access to the account again.

## The nologin Shell

The `nologin` shell acts as a replacement shell for the user accounts that are not intended to interactively log in to the system. It is good security practice to disable an account from logging in to the system when the account does not require it. For example, a mail server might require an account to store mail and a password for the user to authenticate with a mail client to retrieve mail. That user does not need to log directly in to the system.

A common solution to this situation is to set the user's login shell to `/sbin/nologin`. If the user attempts to log in to the system directly, then the `nologin` shell closes the connection.

```
[root@host ~]# usermod -s /sbin/nologin newapp
[root@host ~]# su - newapp
Last login: Wed Feb  6 17:03:06 IST 2019 on pts/0
This account is currently not available.
```



### Important

The `nologin` shell prevents interactive use of the system, but does not prevent all access. Users might be able to authenticate and upload or retrieve files through applications such as web applications, file transfer programs, or mail readers if they use the user's password for authentication.



### References

`chage(1)`, `usermod(8)`, `shadow(5)`, `crypt(3)`, and `login.defs(5)` man pages.

## ► Guided Exercise

# Manage User Passwords

In this exercise, you set password policies for several users.

## Outcomes

- Force a password change when the user logs in to the system for the first time.
- Force a password change every 90 days.
- Set the account to expire 180 days from the current day.

## Before You Begin

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start users-password
```

## Instructions

- 1. From workstation, open an SSH session as student to servera.

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

- 2. On servera, use the usermod command to lock and unlock the operator1 user.

- 2.1. As the student user, use administrative rights to lock the operator1 account.

```
[student@servera ~]$ sudo usermod -L operator1  
[sudo] password for student:
```

- 2.2. Attempt to log in as operator1. This command should fail.

```
[student@servera ~]$ su - operator1  
Password: redhat  
su: Authentication failure
```

- 2.3. Unlock the operator1 account.

```
[student@servera ~]$ sudo usermod -U operator1
```

- 2.4. Attempt to log in as operator1 again. This time, the command should succeed.

```
[student@servera ~]$ su - operator1
Password: redhat
...output omitted...
[operator1@servera ~]$
```

2.5. Log out of the operator1 user shell to return to the student user shell.

```
[operator1@servera ~]$ exit
logout
```

- 3. Change the password policy for the operator1 user to require a new password every 90 days. Confirm that the password age is successfully set.

3.1. Switch to the root user.

```
[student@servera ~]$ sudo -i
[sudo] password for student:
[root@servera ~]#
```

3.2. Set the maximum age of the operator1 user's password to 90 days.

```
[root@servera ~]# chage -M 90 operator1
```

3.3. Verify that the operator1 user's password expires 90 days after it is changed.

```
[root@servera ~]# chage -l operator1
Last password change      : Mar 10, 2022
Password expires          : Jun 10, 2022
Password inactive         : never
Account expires           : never
Minimum number of days between password change   : 0
Maximum number of days between password change   : 90
Number of days of warning before password expires : 7
```

- 4. Force a password change on the first login for the operator1 account.

```
[root@servera ~]# chage -d 0 operator1
```

- 5. Exit as the root user from the servera machine.

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

- 6. Log in as operator1 and change the password to forsooth123. After setting the password, return to the student user's shell.

6.1. Log in as operator1 and change the password to forsooth123 when prompted.

```
[student@servera ~]$ su - operator1
Password: redhat
You are required to change your password immediately (administrator enforced)
Current password: redhat
New password: forsooth123
Retype new password: forsooth123
...output omitted...
[operator1@servera ~]$
```

- 6.2. Exit the **operator1** user's shell to return to the **student** user and then switch to the **root** user.

```
[operator1@servera ~]$ exit
logout
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 7. Set the **operator1** account to expire 180 days from the current day.
- 7.1. Determine a date 180 days in the future. Use the format %F with the **date** command to get the exact value. This returned date is an example; use the value on your system for the steps after this one.

```
[root@servera ~]# date -d "+180 days" +%F
2022-09-06
```

- 7.2. Set the account to expire on the date that is displayed in the preceding step. For example:

```
[root@servera ~]# chage -E 2022-09-06 operator1
```

- 7.3. Verify that the account expiry date is successfully set.

```
[root@servera ~]# chage -l operator1
Last password change      : Mar 10, 2022
Password expires          : Jun 10, 2022
Password inactive         : never
Account expires          : Sep 06, 2022
Minimum number of days between password change   : 0
Maximum number of days between password change   : 90
Number of days of warning before password expires : 7
```

- ▶ 8. Set the passwords to expire 180 days from the current date for all users. Use administrative rights to edit the configuration file.
- 8.1. Set **PASS\_MAX\_DAYS** to 180 in **/etc/login.defs**. Use administrative rights when you open the file with the text editor. You can use the **vim /etc/login.defs** command to perform this step.

```
...output omitted...
# Password aging controls:
#
#      PASS_MAX_DAYS   Maximum number of days a password may be
#      used.
#      PASS_MIN_DAYS   Minimum number of days allowed between
#      password changes.
#      PASS_MIN_LEN     Minimum acceptable password length.
#      PASS_WARN_AGE    Number of days warning given before a
#      password expires.
#
PASS_MAX_DAYS 180
PASS_MIN_DAYS  0
PASS_WARN_AGE  7
...output omitted...
```



### Important

The default password and account expiry settings apply to new users but not to existing users.

8.2. 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, 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 users-password
```

This concludes the section.

## ► Lab

# Manage Local Users and Groups

In this lab, you set a default local password policy, create a secondary group for three users, allow that group to use `sudo` to run commands as `root`, and modify the password policy for one user.

## Outcomes

- Set a default password aging policy of the local user's password.
- Create and use a secondary group for new users.
- Create three new users with the new secondary group.
- Set an initial password for the created users.
- Configure the secondary group members to use the `sudo` command to run any command as any user.
- Set a user-specific password aging policy.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. From the `workstation` machine, open an SSH session to the `serverb` machine as the `student` user and switch to the `root` user.
2. On the `serverb` machine, ensure that newly created users must change their passwords every 30 days.
3. Create the `consultants` group with a GID of 35000.
4. Configure administrative rights to enable all `consultants` group members to execute any command as any user.
5. Create the `consultant1`, `consultant2`, and `consultant3` users with the `consultants` group as their secondary group.
6. Set the `consultant1`, `consultant2`, and `consultant3` passwords to `redhat`.
7. Set the `consultant1`, `consultant2`, and `consultant3` accounts to expire in 90 days from the current day.
8. Change the password policy for the `consultant2` account to require a new password every 15 days.
9. Additionally, force the `consultant1`, `consultant2`, and `consultant3` users to change their passwords on the first login.

## 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 users-review
```

## 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 users-review
```

This concludes the section.

## ► Solution

# Manage Local Users and Groups

In this lab, you set a default local password policy, create a secondary group for three users, allow that group to use `sudo` to run commands as `root`, and modify the password policy for one user.

## Outcomes

- Set a default password aging policy of the local user's password.
- Create and use a secondary group for new users.
- Create three new users with the new secondary group.
- Set an initial password for the created users.
- Configure the secondary group members to use the `sudo` command to run any command as any user.
- Set a user-specific password aging policy.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. From the `workstation` machine, open an SSH session to the `serverb` machine as the `student` user and switch to the `root` user.

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

2. On the `serverb` machine, ensure that newly created users must change their passwords every 30 days.
  - 2.1. Set `PASS_MAX_DAYS` to 30 in the `/etc/login.defs` file. Use administrative rights while opening the file with the text editor. You can use the `vim /etc/login.defs` command to perform this step.

```
...output omitted...
# Password aging controls:
#
#       PASS_MAX_DAYS    Maximum number of days a password may be
```

```
#      used.
#      PASS_MIN_DAYS   Minimum number of days allowed between
#      password changes.
#      PASS_MIN_LEN    Minimum acceptable password length.
#      PASS_WARN_AGE   Number of days warning given before a
#      password expires.
#
PASS_MAX_DAYS 30
PASS_MIN_DAYS  0
PASS_WARN_AGE  7
...output omitted...
```

3. Create the consultants group with a GID of 35000.

```
[root@serverb ~]# groupadd -g 35000 consultants
```

4. Configure administrative rights to enable all consultants group members to execute any command as any user.
  - 4.1. Create the /etc/sudoers.d/consultants file and add the following content to it. You can use the `vim /etc/sudoers.d/consultants` command to perform this step.

```
%consultants  ALL=(ALL) ALL
```

5. Create the consultant1, consultant2, and consultant3 users with the consultants group as their secondary group.

```
[root@serverb ~]# useradd -G consultants consultant1
[root@serverb ~]# useradd -G consultants consultant2
[root@serverb ~]# useradd -G consultants consultant3
```

6. Set the consultant1, consultant2, and consultant3 passwords to redhat.

```
[root@serverb ~]# passwd consultant1
Changing password for user consultant1.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
[root@serverb ~]# passwd consultant2
Changing password for user consultant2.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully
[root@serverb ~]# passwd consultant3
Changing password for user consultant3.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully
```

7. Set the `consultant1`, `consultant2`, and `consultant3` accounts to expire in 90 days from the current day.

- 7.1. Determine the date 90 days in the future. This returned date is an example; the value that you see, to use in the following step, is based on the current date and time in your system.

```
[root@serverb ~]# date -d "+90 days" +%F  
2022-06-08
```

- 7.2. Set the account expiry date of the `consultant1`, `consultant2`, and `consultant3` accounts to the same value as determined in the preceding step. For example:

```
[root@serverb ~]# chage -E 2022-06-08 consultant1  
[root@serverb ~]# chage -E 2022-06-08 consultant2  
[root@serverb ~]# chage -E 2022-06-08 consultant3
```

8. Change the password policy for the `consultant2` account to require a new password every 15 days.

```
[root@serverb ~]# chage -M 15 consultant2
```

9. Additionally, force the `consultant1`, `consultant2`, and `consultant3` users to change their passwords on the first login.

- 9.1. Set the last day of the password change to 0 so that users must change the password when they first log in to the system.

```
[root@serverb ~]# chage -d 0 consultant1  
[root@serverb ~]# chage -d 0 consultant2  
[root@serverb ~]# chage -d 0 consultant3
```

- 9.2. 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 users-review
```

## 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 users-review
```

This concludes the section.

# Summary

---

- The user account types in Linux are the superuser, system users, and regular users.
- A user has a primary group and might be a member of secondary groups.
- The `/etc/passwd`, `/etc/group`, and `/etc/shadow` critical files contain user and group information.
- You can run commands as the superuser with the `su` and `sudo` commands.
- The `useradd`, `usermod`, and `userdel` commands manage users.
- The `groupadd`, `groupmod`, and `groupdel` commands manage groups.
- The `passwd` command manages passwords for users.
- The `chage` command displays and configures password expiration settings for users.



## Chapter 4

# Control Access to Files

### Goal

Set Linux file-system permissions on files and to interpret the security effects of different permission settings.

### Objectives

- Change the permissions and ownership of files with command-line tools.
- Control the default permissions of user-created files, explain the effect of special permissions, and use special and default permissions to set the group owner of files that are created in a directory.

### Sections

- Manage File System Permissions from the Command Line (and Guided Exercise)
- Manage Default Permissions and File Access (and Guided Exercise)

### Lab

- Control Access to Files

# Manage File System Permissions from the Command Line

---

## Objectives

Change the permissions and ownership of files with command-line tools.

## Change File and Directory Permissions

The `chmod` command changes file and directory permissions from the command line. The `chmod` command can be interpreted as "change mode", because the *mode* of a file is another name for file permissions. The `chmod` command takes a permission instruction followed by a list of files or directories to change. You can set the permission instruction either symbolically or in octal (numeric) notation.

### Change Permissions with the Symbolic Method

Use the `chmod` command to modify file and directory permissions. The following example can help you to understand the `chmod` command usage:

```
chmod Who/What/Which file|directory
```

*Who* is the class of user, as in the following table. If you do not provide a class of user, then the `chmod` command uses the `all` group as default.

Who	Set	Description
u	user	The file owner.
g	group	Member of the file's group.
o	other	Users who are not the file owner nor members of the file's group.
a	all	All the three previous groups.

*What* is the operator that modifies the *Which*, as in the table below.

What	Operation	Description
+	add	Adds the permissions to the file.
-	remove	Removes the permissions to the file.
=	set exactly	Set exactly the provided permissions to the file.

*Which* is the mode, and specifies the permissions to the files or directories, as in the table below.

Which	Mode	Description
r	read	Read access to the file. Listing access to the directory.

Which	Mode	Description
w	<i>write</i>	Write permissions to the file or directory.
x	<i>execute</i>	Execute permissions to the file. Allows to enter the directory, and access files and subdirectories inside the directory.
X	<i>special execute</i>	Execute permissions for a directory, or execute permissions to a file if it has at least one of the execute bits set.

The *symbolic* method of changing file permissions uses letters to represent the different groups of permissions: **u** for user, **g** for group, **o** for other, and **a** for all.

With the symbolic method, you do not need to set a complete new group of permissions. Instead, you can change one or more of the existing permissions. Use the plus (+) or the minus (-) characters to add or remove permissions, respectively, or use the equal (=) character to replace the entire set for a group of permissions.

A single letter represents the permissions themselves: r for read, w for write, and x for execute. You can use an uppercase X as the permission flag to add execute permissions only if the file is a directory or if execute is already set for user, group, or other.

The following list shows some examples for changing permissions with the symbolic method:

Remove read and write permission for group and other on the `document.pdf` file:

```
[user@host ~]$ chmod go-rw document.pdf
```

Add execute permission for everyone on the `myscript.sh` file:

```
[user@host ~]$ chmod a+x myscript.sh
```

You can use the `chmod` command **-R** option to **recursively set permissions** on the files in an entire directory tree. For example, the next command recursively adds read, write, and execute permissions for the members of the `myfolder` directory and the files and directories inside it.

```
[user@host ~]$ chmod -R g+rwx /home/user/myfolder
```

You can also use the `chmod` command **-R** option with the **-X** option to set permissions symbolically. The `chmod` command X option allows you to set the execute (search) permission on directories so that their contents can be accessed, without changing permissions on most files. However, be cautious with the X option because if a file has any execute permission set, then the X option sets the specified execute permission on that file as well.

For example, the following command recursively sets read and write access on the `demodir` directory and all its children for their group owner, but only applies group execute permissions to directories and files that already have execute set for user, group, or other.

```
[root@host opt]# chmod -R g+rwx demodir
```

## Change Permissions with the Octal Method

You can use the `chmod` command to change file permissions with the octal method instead of the symbolic method. In the following example, the # character represents a digit.

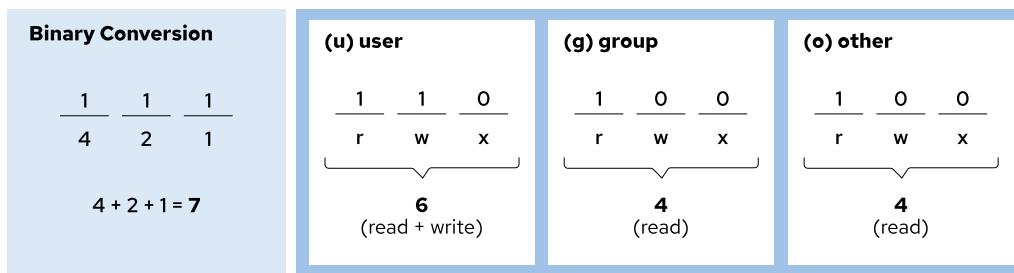
```
chmod ### file|directory
```

With the octal method, you can represent permissions as a 3-digit (or 4-digit, when setting advanced permissions) *octal* number. A single octal digit can represent any single value from 0-7.

In the 3-digit octal representation of permissions, each digit stands for one access level, from left to right: user, group, and other. To determine each digit:

- Start with 0.
- If you want to add read permissions for this access level, then add 4.
- If you want to add write permissions, then add 2.
- If you want to add execute permissions, then add 1.

The following diagram illustrates how systems interpret the 644 octal permission value.



**Figure 4.1: Visual representation of the octal method**

Experienced administrators often use octal permissions because they are easier to implement on single or matching files, and still provides full permission control.

The following list shows some examples for changing permissions with the octal method:

Set read and write permissions for user, and read permission for group and other, on the `sample.txt` file:

```
[user@host ~]$ chmod 644 sample.txt
```

Set read, write, and execute permissions for user, read and execute permissions for group, and no permission for other on the `sampledir` directory:

```
[user@host ~]$ chmod 750 sampledir
```

## Change File and Directory User or Group Ownership

The user owns a file that it creates. By default, new files have a group ownership that is the primary group of the user that creates the file. In Red Hat Enterprise Linux, a user's primary group is usually a private group with only that user as a member. To grant access to a file based on group membership, you might need to change the group that owns the file.

## Chapter 4 | Control Access to Files

Only the `root` user can change the user that owns a file. However, the file's owner and the `root` user can set group ownership. The `root` user can grant file ownership to any group, but only regular users can change the file's group ownership if they are a member of the destination group.

You can change file ownership by using the `chown` (change owner) command. For example, to grant ownership of the `app.conf` file to the `student` user, use the following command:

```
[root@host ~]# chown student app.conf
```

The `chown` command `-R` option recursively changes the ownership of an entire directory tree. The following command grants ownership of the `Pictures` directory and all files and subdirectories within it to the `student` user:

```
[root@host ~]# chown -R student Pictures
```

You can also use the `chown` command to change group ownership of a file by preceding the group name with a colon (`:`). For example, the following command changes the group ownership of the `Pictures` directory to `admins`:

```
[root@host ~]# chown :admins Pictures
```

You can use the `chown` command to change both owner and group at the same time by using the `owner:group` syntax. For example, to change the ownership of the `Pictures` directory to the `visitor` user and the group to `guests`, use the following command:

```
[root@host ~]# chown visitor:guests Pictures
```

Instead of using the `chown` command, some users change the group ownership by using the `chgrp` command. This command works similarly to `chown`, except that you can use it only to change group ownership and the colon (`:`) before the group name is not required.



### Important

You might encounter alternative `chown` syntax that separates owner and group with a period character instead of a colon:

```
[root@host ~]# chown owner.group filename
```

Red Hat recommends to not use this syntax, and to always use a colon. Because a period is a valid character in a user name, a `chown` command might misinterpret your intent. The command may interpret the user and group as a file name. Instead, only use a colon character when setting the user and group at the same time.



### References

`ls(1)`, `chmod(1)`, `chown(1)`, and `chgrp(1)` man pages

## ► Guided Exercise

# Manage File System Permissions from the Command Line

In this exercise, you use file system permissions to create a directory in which all members of a particular group can add and delete files.

### Outcomes

- Create a collaborative directory that all members of a particular group can access.

### Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start perms-cli
```

### Instructions

- 1. From **workstation**, log in to **servera** as the **student** user and switch to the **root** user.

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

- 2. Create the **/home/consultants** directory.

```
[root@servera ~]# mkdir /home/consultants
```

- 3. Change the group ownership of the **consultants** directory to **consultants**.

```
[root@servera ~]# chown :consultants /home/consultants
```

- 4. Verify that the permissions of the **consultants** group allow its group members to create files in, and delete files from the **/home/consultants** directory. Use the symbolic method for setting the appropriate permissions.

The permissions should forbid others from accessing the files. Use the octal method for setting the appropriate permissions.

- 4.1. Verify that the permissions of the **consultants** group allow its group members to create files in, and delete files from the **/home/consultants** directory.

**Chapter 4 |** Control Access to Files

Note that the **consultants** group currently does not have write permission.

```
[root@servera ~]# ls -ld /home/consultants  
drwxr-xr-x. 2 root     consultants   6 Mar  1 12:08 /home/consultants
```

4.2. Add write permission to the **consultants** group.

```
[root@servera ~]# chmod g+w /home/consultants  
[root@servera ~]# ls -ld /home/consultants  
drwxrwxr-x. 2 root consultants 6 Mar  1 13:21 /home/consultants
```

4.3. Forbid others from accessing files in the **/home/consultants** directory.

```
[root@servera ~]# chmod 770 /home/consultants  
[root@servera ~]# ls -ld /home/consultants  
drwxrwx---. 2 root consultants 6 Mar  1 12:08 /home/consultants/
```

► 5. Exit the **root** shell and switch to the **consultant1** user. The password is **redhat**.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ su - consultant1  
Password: redhat  
[consultant1@servera ~]$
```

► 6. Navigate to the **/home/consultants** directory and create a file called **consultant1.txt**.

6.1. Change to the **/home/consultants** directory.

```
[consultant1@servera ~]$ cd /home/consultants
```

6.2. Create an empty file called **consultant1.txt**.

```
[consultant1@servera consultants]$ touch consultant1.txt
```

► 7. List the default user and group ownership of the new file and its permissions.

```
[consultant1@servera consultants]$ ls -l consultant1.txt  
-rw-rw-r--. 1 consultant1 consultant1 0 Mar  1 12:53 consultant1.txt
```

► 8. Ensure that all members of the **consultants** group can edit the **consultant1.txt** file. Change the group ownership of the **consultant1.txt** file to **consultants**.

8.1. Use the **chown** command to change the group ownership of the **consultant1.txt** file to **consultants**.

```
[consultant1@servera consultants]$ chown :consultants consultant1.txt
```

- 8.2. List the new ownership of the `consultant1.txt` file.

```
[consultant1@servera consultants]$ ls -l consultant1.txt  
-rw-rw-r--. 1 consultant1 consultants 0 Mar 1 12:53 consultant1.txt
```

- 9. Exit the shell and switch to the `consultant2` user. The password is `redhat`.

```
[consultant1@servera consultants]$ exit  
logout  
[student@servera ~]$ su - consultant2  
Password: redhat  
[consultant2@servera ~]$
```

- 10. Navigate to the `/home/consultants` directory. Ensure that the `consultant2` user can add content to the `consultant1.txt` file.

- 10.1. Change to the `/home/consultants` directory. Add text to the `consultant1.txt` file.

```
[consultant2@servera ~]$ cd /home/consultants/  
[consultant2@servera consultants]$ echo "text" >> consultant1.txt
```

- 10.2. Verify that the text is present in the `consultant1.txt` file.

```
[consultant2@servera consultants]$ cat consultant1.txt  
text
```

- 10.3. Return to the `workstation` system as the `student` user.

```
[consultant2@servera consultants]$ exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

## 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 perms-cli
```

This concludes the section.

# Manage Default Permissions and File Access

---

## Objectives

Control the default permissions of user-created files, explain the effect of special permissions, and use special and default permissions to set the group owner of files that are created in a directory.

## Special Permissions

*Special permissions* are a fourth permission type in addition to the basic user, group, and other types. As the name implies, special permissions provide additional access-related features beyond what the basic permission types allow. This section describes the impact of special permissions, which are summarized in the table below.

### Effects of Special Permissions on Files and Directories

Permission	Effect on files	Effect on directories
u+s (suid)	File executes as the user that owns the file, not as the user that ran the file.	No effect.
g+s (sgid)	File executes as the group that owns the file.	Files that are created in the directory have a group owner to match the group owner of the directory.
o+t (sticky)	No effect.	Users with write access to the directory can remove only files that they own; they cannot remove or force saves to files that other users own.

The *setuid* permission on an executable file means that commands run as the user that owns that file, rather than as the user that ran the command. One example is the `passwd` command:

```
[user@host ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 35504 Jul 16 2010 /usr/bin/passwd
```

In a long listing, you can identify the *setuid* permissions by a lowercase s character in the place where you would normally expect the x character (owner execute permissions). If the owner does not have execute permissions, then this character is replaced by an uppercase S character.

The *setgid* special permission on a directory means that created files in the directory inherit their group ownership from the directory, rather than inheriting group ownership from the creating user. This feature is commonly used on group collaborative directories to automatically change a file from the default private group to the shared group, or if a specific group should always own files in a directory. An example of this behavior is the `/run/log/journal` directory:

```
[user@host ~]$ ls -ld /run/log/journal
drwxr-sr-x. 3 root systemd-journal 60 May 18 09:15 /run/log/journal
```

If `setgid` is set on an executable file, then commands run as the group that owns that file, rather than as the user that ran the command. This condition is similar to the way that `setuid` works. One example is the `locate` command:

```
[user@host ~]$ ls -ld /usr/bin/locate
-rwx--s--x. 1 root slocate 47128 Aug 12 17:17 /usr/bin/locate
```

In a long listing, you can identify the `setgid` permissions by a lowercase `s` character in the place where you would normally expect the `x` character (group execute permissions). If the group does not have execute permissions, then this character is replaced by an uppercase `S` character.

Finally, the *sticky bit* for a directory sets a special restriction on deletion of files. Only the owner of the file (and the `root` user) can delete files within the directory. An example is the `/tmp` directory:

```
[user@host ~]$ ls -ld /tmp
drwxrwxrwt. 39 root root 4096 Feb 8 20:52 /tmp
```

In a long listing, you can identify the sticky permissions by a lowercase `t` character in the place where you would normally expect the `x` character (other execute permissions). If other does not have execute permissions, then this character is replaced by an uppercase `T` character.

#### Setting Special Permissions

- **Symbolic** : `setuid = u+s; setgid = g+s; sticky = o+t`
- **Octal** : In the added fourth preceding digit; `setuid = 4; setgid = 2; sticky = 1`

Examples of Special permissions Add the `setgid` bit on the `example` directory by using the symbolic method:

```
[user@host ~]# chmod g+s example
```

Remove the `setuid` bit on the `example` directory by using the symbolic method:

```
[user@host ~]# chmod u-s example
```

Set the `setgid` bit and add read, write, and execute permissions for user and group, with no access for others, on the `example` directory by using the octal method:

```
[user@host ~]# chmod 2770 example
```

Remove the `setgid` bit and add read, write, and execute permissions for user and group, with no access for others, on the `example` directory by using the octal method. Note that you need to add an extra `0` at the beginning of the permissions value when removing special permissions by using the octal method:

```
[user@host ~]# chmod 00770 example
```

## Default File Permissions

On creation, a file is assigned initial permissions. Two things affect these initial permissions. The first is whether you are creating a regular file or a directory. The second is the current *umask*, which stands for user file-creation mask.

## Chapter 4 | Control Access to Files

If you create a directory, then its initial octal permissions are 0777 (drwxrwxrwx). If you create a regular file, then its initial octal permissions are 0666 (-rw-rw-rw-). You must always explicitly add execute permission to a regular file. This step makes it harder for an attacker to compromise a system, create a malicious file, and run it.

Additionally, the shell session sets a umask to further restrict the initial permissions of a file. The umask is an octal bitmask that clears the permissions of new files and directories that a process creates. If a bit is set in the umask, then the corresponding permission is cleared on new files. For example, the umask 0002 clears the write bit for other users. The leading zeros indicate that the special, user, and group permissions are not cleared. A umask of 0077 clears all the group and other permissions of newly created files.

The `umask` command without arguments displays the current value of the shell's umask:

```
[user@host ~]$ umask
0002
```

Use the `umask` command with a single octal argument to change the umask of the current shell. The argument should be an octal value that corresponds to the new umask value. You can omit any leading zeros in the umask.

The system's default umask values for Bash shell users are defined in the `/etc/profile` and `/etc/bashrc` files. Users can override the system defaults in the `.bash_profile` or `.bashrc` files in their home directories.

## Effect of umask Utility on Permissions

The following example explains how the umask affects the permissions of files and directories. Look at the default umask permissions for both files and directories in the current shell.

If you create a regular file, then its initial octal permissions are 0666 (000 110 110 110, in binary representation). Then, the 0002 umask (000 000 000 010) disables the write permission bit for others. Thus, the owner and group both have read and write permission on files, and other is set to read (000 110 110 100).

	Symbolic	Numeric octal	Numeric binary
<b>Initial file permissions</b>	<code>rw-rw-rw-</code>	0666	000 110 110 110
<b>umask</b>	<code>-----w-</code>	0002	000 000 000 010
<b>Resulting file permissions</b>	<code>rw-rw-r--</code>	0664	000 110 110 100

Figure 4.2: Example of umask calculation on a file

```
[user@host ~]$ umask
0002
[user@host ~]$ touch default.txt
[user@host ~]$ ls -l default.txt
-rw-rw-r-- 1 user user 0 May  9 01:54 default.txt
```

If you create a directory, then its initial octal permissions are 0777 (000 111 111 111). Then, the 0002 umask (000 000 000 010) disables the write permission bit for other. Thus, the owner and group both have read, write, and execute permissions on directories, and other is set for read and execution (000 111 111 101).

	Symbolic	Numeric octal	Numeric binary
<b>Initial directory permissions</b>	rwxrwxrwx	0777	000 111 111 111
<b>umask</b>	-----w-	0002	000 000 000 010
<b>Resulting directory permissions</b>	rwxrwxr-x	0775	000 111 111 101

Figure 4.3: Example of umask calculation on a directory

```
[user@host ~]$ umask
0002
[user@host ~]$ mkdir default
[user@host ~]$ ls -ld default
drwxrwxr-x. 2 user user 0 May  9 01:54 default
```

By setting the umask value to 0, the file permissions for other change from read to read and write. The directory permissions for other change from read and execute to read, write, and execute.

```
[user@host ~]$ umask 0
[user@host ~]$ touch zero.txt
[user@host ~]$ ls -l zero.txt
-rw-rw-rw-. 1 user user 0 May  9 01:54 zero.txt
[user@host ~]$ mkdir zero
[user@host ~]$ ls -ld zero
drwxrwxrwx. 2 user user 0 May  9 01:54 zero
```

To mask all file and directory permissions for other, set the umask value to 007.

```
[user@host ~]$ umask 007
[user@host ~]$ touch seven.txt
[user@host ~]$ ls -l seven.txt
-rw-rw----. 1 user user 0 May  9 01:55 seven.txt
[user@host ~]$ mkdir seven
[user@host ~]$ ls -ld seven
drwxrwx---. 2 user user 0 May  9 01:54 seven
```

A umask of 027 ensures that new files have read and write permissions for user and read permission for group. New directories have read and write access for group and no permissions for other.

```
[user@host ~]$ umask 027
[user@host ~]$ touch two-seven.txt
[user@host ~]$ ls -l two-seven.txt
-rw-r-----. 1 user user 0 May  9 01:55 two-seven.txt
[user@host ~]$ mkdir two-seven
[user@host ~]$ ls -ld two-seven
drwxr-x---. 2 user user 0 May  9 01:54 two-seven
```

The shell startup scripts set the default umask for users. By default, if the account's UID is 200 or greater and the username and primary group name are the same, then the account is assigned a umask of 002. Otherwise, the umask is 022.

The `root` user can change the default umask by adding a `local-umask.sh` shell startup script in the `/etc/profile.d/` directory. The following example shows a `local-umask.sh` file:

```
[root@host ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration asda sda
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

The preceding example sets the umask to 007 for users with a UID greater than 199 and with a username and primary group name that match, and to 022 for everyone else. If you want to set the umask to 022 for everyone, then create that file with the following content:

```
# Overrides default umask configuration
umask 022
```

The current umask of a shell applies until you log out of the shell and log back in.



## References

`bash(1)`, `ls(1)`, `chmod(1)`, and `umask(1)` man pages

## ► Guided Exercise

# Manage Default Permissions and File Access

In this exercise, you control the permissions on files that are created in a directory by using umask settings and the setgid permission.

### Outcomes

- Create a shared directory where the operators group automatically owns new files.
- Experiment with various umask settings.
- Adjust default permissions for specific users.
- Verify your adjustment.

### Before You Begin

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start perms-default
```

### Instructions

- 1. Log in to the servera system as the student user.

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

- 2. Switch to the operator1 user with redhat as the password.

```
[student@servera ~]$ su - operator1  
Password: redhat  
[operator1@servera ~]$
```

- 3. List the operator1 user's default umask value.

```
[operator1@servera ~]$ umask  
0002
```

- 4. Create a /tmp/shared directory. In the /tmp/shared directory, create a defaults file. Look at the default permissions.

- 4.1. Create the /tmp/shared directory. List the permissions of the new directory.

```
[operator1@servera ~]$ mkdir /tmp/shared  
[operator1@servera ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 operator1 operator1 6 Feb 4 14:06 /tmp/shared
```

- 4.2. Create a `defaults` file in the `/tmp/shared` directory.

```
[operator1@servera ~]$ touch /tmp/shared/defaults
```

- 4.3. List the permissions of the new file.

```
[operator1@servera ~]$ ls -l /tmp/shared/defaults  
-rw-rw-r--. 1 operator1 operator1 0 Feb 4 14:09 /tmp/shared/defaults
```

- 5. Change the group ownership of the `/tmp/shared` directory to the `operators` group. Confirm the new ownership and permissions.

- 5.1. Change the group ownership of the `/tmp/shared` directory to the `operators` group.

```
[operator1@servera ~]$ chown :operators /tmp/shared
```

- 5.2. List the permissions of the `/tmp/shared` directory.

```
[operator1@servera ~]$ ls -ld /tmp/shared  
drwxrwxr-x. 2 operator1 operators 22 Feb 4 14:09 /tmp/shared
```

- 5.3. Create a group file in the `/tmp/shared` directory. List the file permissions.

```
[operator1@servera ~]$ touch /tmp/shared/group  
[operator1@servera ~]$ ls -l /tmp/shared/group  
-rw-rw-r--. 1 operator1 operator1 0 Feb 4 17:00 /tmp/shared/group
```



### Note

The group owner of the `/tmp/shared/group` file is not `operators` but `operator1`.

- 6. Ensure that the `operators` group owns files that are created in the `/tmp/shared` directory.

- 6.1. Set the group ID to the `operators` group for the `/tmp/shared` directory.

```
[operator1@servera ~]$ chmod g+s /tmp/shared
```

- 6.2. Create a `ops_db.txt` file in the `/tmp/shared` directory.

```
[operator1@servera ~]$ touch /tmp/shared/ops_db.txt
```

- 6.3. Verify that the `operators` group is the group owner for the new file.

```
[operator1@servera ~]$ ls -l /tmp/shared/ops_db.txt  
-rw-rw-r--. 1 operator1 operators 0 Feb 4 16:11 /tmp/shared/ops_db.txt
```

- 7. Create an ops\_net.txt file in the /tmp/shared directory. Record the ownership and permissions. Change the umask for the operator1 user. Create an ops\_prod.txt file. Record the ownership and permissions of the ops\_prod.txt file.

7.1. Create an ops\_net.txt file in the /tmp/shared directory.

```
[operator1@servera ~]$ touch /tmp/shared/ops_net.txt
```

7.2. List the permissions of the ops\_net.txt file.

```
[operator1@servera ~]$ ls -l /tmp/shared/ops_net.txt  
-rw-rw-r--. 1 operator1 operators 5 Feb 0 15:43 /tmp/shared/ops_net.txt
```

7.3. Change the umask for the operator1 user to 027. Confirm the change.

```
[operator1@servera ~]$ umask 027  
[operator1@servera ~]$ umask  
0027
```

7.4. Create an ops\_prod.txt file in the /tmp/shared/ directory. Verify that newly created files have read-only access for the operators group and no access for other users.

```
[operator1@servera ~]$ touch /tmp/shared/ops_prod.txt  
[operator1@servera ~]$ ls -l /tmp/shared/ops_prod.txt  
-rwxr-----. 1 operator1 operators 0 Feb 0 15:56 /tmp/shared/ops_prod.txt
```

- 8. Open a new terminal window and log in to servera as operator1.

```
[student@workstation ~]$ ssh operator1@servera  
...output omitted...  
[operator1@servera ~]$
```

- 9. List the umask value for operator1.

```
[operator1@servera ~]$ umask  
0002
```

- 10. Change the default umask for the operator1 user. The new umask prohibits all access for users that are not in their group. Confirm that the umask is changed.

10.1. Change the default umask for the operator1 user to 007.

```
[operator1@servera ~]$ echo "umask 007" >> ~/.bashrc  
[operator1@servera ~]$ cat ~/.bashrc  
# .bashrc
```

```
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
...output omitted...
umask 007
```

- 10.2. Log out and log in again as the `operator1` user. Confirm that the change is permanent.

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh operator1@servera
...output omitted...
[operator1@servera ~]$ umask
0007
```

- 11. On `servera`, close all `operator1` and `student` user shells. Return to the `workstation` system as the `student` user.



### Warning

Failure to exit from all `operator1` shells causes the finish script to fail.

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

## 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 perms-default
```

This concludes the section.

## ► Lab

# Control Access to Files

In this lab, you configure permissions on files and set up a directory that users in a particular group can use to conveniently share files on the local file system.

## Outcomes

- Create a directory where users can work collaboratively on files.
- Create files that are automatically assigned group ownership.
- Create files that are not accessible outside the group.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. Log in to `serverb` as the student user. Switch to the root user, and use `redhat` as the password.
2. Create a `/home/techdocs` directory.
3. Change the group ownership of the `/home/techdocs` directory to the `techdocs` group.
4. Verify that users in the `techdocs` group cannot create files in the `/home/techdocs` directory.
5. Set permissions on the `/home/techdocs` directory. On the `/home/techdocs` directory, configure `setgid` (2); read, write, and execute permissions (7) for the owner/user and group; and no permissions (0) for other users.
6. Verify that the permissions are set properly.  
The `techdocs` group now has write permission.
7. Confirm that users in the `techdocs` group can now create and edit files in the `/home/techdocs` directory. Users that are not in the `techdocs` group cannot edit or create files in the `/home/techdocs` directory. The `tech1` and `tech2` users are in the `techdocs` group. The `database1` user is not in that group.
8. Modify the global login scripts. Normal users should have a `umask` setting that allows the user and group to create, write and execute files and directories, while preventing other users from viewing, modifying, or executing new files and directories.

## 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 perms-review
```

## 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 perms-review
```

This concludes the section.

## ► Solution

# Control Access to Files

In this lab, you configure permissions on files and set up a directory that users in a particular group can use to conveniently share files on the local file system.

## Outcomes

- Create a directory where users can work collaboratively on files.
- Create files that are automatically assigned group ownership.
- Create files that are not accessible outside the group.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. Log in to `serverb` as the student user. Switch to the `root` user, and use `redhat` as the password.

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

2. Create a `/home/techdocs` directory.

- 2.1. Use the `mkdir` command to create a `/home/techdocs` directory.

```
[root@serverb ~]# mkdir /home/techdocs
```

3. Change the group ownership of the `/home/techdocs` directory to the `techdocs` group.

- 3.1. Use the `chown` command to change the group ownership for the `/home/techdocs` directory to the `techdocs` group.

```
[root@serverb ~]# chown :techdocs /home/techdocs
```

4. Verify that users in the `techdocs` group cannot create files in the `/home/techdocs` directory.

**Chapter 4 |** Control Access to Files

- 4.1. Use the `su` command to switch to the `tech1` user.

```
[root@serverb ~]# su - tech1  
[tech1@serverb ~]$
```

- 4.2. Create a `techdoc1.txt` file in the `/home/techdocs` directory. This step should fail.

Although the `/home/techdocs` directory is owned by the `techdocs` group and `tech1` is part of the `techdocs` group, you cannot create a file in that directory. The reason is because the `techdocs` group does not have write permission.

```
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt  
touch: cannot touch '/home/techdocs/techdoc1.txt': Permission denied
```

- 4.3. List the directory's permissions.

```
[tech1@serverb ~]$ ls -ld /home/techdocs/  
drwxr-xr-x. 2 root techdocs 6 Feb 5 16:05 /home/techdocs/
```

5. Set permissions on the `/home/techdocs` directory. On the `/home/techdocs` directory, configure `setgid` (2); read, write, and execute permissions (7) for the owner/user and group; and no permissions (0) for other users.

- 5.1. Exit from the `tech1` user shell.

```
[tech1@serverb ~]$ exit  
logout  
[root@serverb ~]#
```

- 5.2. Set the group permission for the `/home/techdocs` directory. Configure `setgid`; read, write, and execute permissions for the owner and group; and no permissions for others.

```
[root@serverb ~]# chmod 2770 /home/techdocs
```

6. Verify that the permissions are set properly.

```
[root@serverb ~]# ls -ld /home/techdocs  
drwxrws---. 2 root techdocs 6 Feb 4 18:12 /home/techdocs/
```

The `techdocs` group now has write permission.

7. Confirm that users in the `techdocs` group can now create and edit files in the `/home/techdocs` directory. Users that are not in the `techdocs` group cannot edit or create files in the `/home/techdocs` directory. The `tech1` and `tech2` users are in the `techdocs` group. The `database1` user is not in that group.

- 7.1. Switch to the `tech1` user. Create a `techdoc1.txt` file in the `/home/techdocs` directory. Exit from the `tech1` user shell.

```
[root@serverb ~]# su - tech1
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
-rw-rw-r--. 1 tech1 techdocs 0 Feb  5 16:42 /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 7.2. Switch to the **tech2** user. Add some content to the **/home/techdocs/techdoc1.txt** file. Exit from the **tech2** user shell.

```
[root@serverb ~]# su - tech2
[tech2@serverb ~]$ cd /home/techdocs
[tech2@serverb techdocs]$ echo "This is the first tech doc." > techdoc1.txt
[tech2@serverb techdocs]$ exit
logout
[root@serverb ~]#
```

- 7.3. Switch to the **database1** user. Append some content to the **/home/techdocs/techdoc1.txt** file. You get a **Permission Denied** message. Verify that **database1** does not have access to the file. Exit from the **database1** user shell.

Enter the following long echo command on a single line:

```
[root@serverb ~]# su - database1
[database1@serverb ~]$ echo "This is the first tech doc." >> \
/home/techdocs/techdoc1.txt
-bash: /home/techdocs/techdoc1.txt: Permission denied
[database1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
ls: cannot access '/home/techdocs/techdoc1.txt': Permission denied
[database1@serverb ~]$ exit
logout
[root@serverb ~]#
```

8. Modify the global login scripts. Normal users should have a umask setting that allows the user and group to create, write and execute files and directories, while preventing other users from viewing, modifying, or executing new files and directories.

- 8.1. Determine the umask of the **student** user. Switch to the **student** login shell. When done, exit from the shell.

```
[root@serverb ~]# su - student
[student@serverb ~]$ umask
0002
[student@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 8.2. Edit the **/etc/profile** file and set the following umask properties. The **/etc/profile** file already contains a umask definition. Search the file and update with the appropriate values.

## Chapter 4 | Control Access to Files

Set a umask of 007 for users with a UID greater than 199 and with a matching username and primary group name. Set a umask of 022 for everyone else.

The following example shows the expected added content in the /etc/profile file.

```
[root@serverb ~]# cat /etc/profile
...output omitted...
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
...output omitted...
```

8.3. As the student user, verify that the global umask changes to 007.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ umask
0007
```

8.4. Return to the workstation system 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 perms-review
```

## 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 perms-review
```

This concludes the section.

# Summary

---

- The `chmod` command changes file permissions from the command line.
- The `chmod` command can use one of two methods to represent permissions: symbolic or octal.
- The `chown` command changes file ownership. The `chown` command `-R` option recursively changes the ownership of a directory tree.
- The `umask` command without arguments displays the current umask value of the shell. Every process on the system has a umask. The default umask values for Bash are defined in the `/etc/profile` and `/etc/bashrc` files.
- The `suid`, `sgid`, and `sticky` special permissions provide additional access-related features to files.

## Chapter 5

# Manage SELinux Security

### Goal

Protect and manage server security by using SELinux.

### Objectives

- Explain how SELinux protects resources, change the current SELinux mode of a system, and set the default SELinux mode of a system.
- Manage the SELinux policy rules that determine the default context for files and directories with the `semanage fcontext` command and apply the context defined by the SELinux policy to files and directories with the `restorecon` command.
- Activate and deactivate SELinux policy rules with the `setsebool` command, manage the persistent value of SELinux Booleans with the `semanage boolean -l` command, and consult man pages that end with `_selinux` to find useful information about SELinux Booleans.
- Use SELinux log analysis tools and display useful information during SELinux troubleshooting with the `sealert` command.

### Sections

- Change the SELinux Enforcement Mode (and Guided Exercise)
- Control SELinux File Contexts (and Guided Exercise)
- Adjust SELinux Policy with Booleans (and Guided Exercise)
- Investigate and Resolve SELinux Issues (and Guided Exercise)

### Lab

Manage SELinux Security

# Change the SELinux Enforcement Mode

## Objectives

Explain how SELinux protects resources, change the current SELinux mode of a system, and set the default SELinux mode of a system.

## Describe 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, which could result in 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 precisely what 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* but 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 loaded policies. This mode is the default 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. SELinux violations are not denied or logged. Disabling SELinux is strongly discouraged.

**Important**

Starting in Red Hat Enterprise Linux 9, SELinux can be fully disabled only by using the `selinux=0` kernel parameter at boot. RHEL no longer supports setting the `SELINUX=disabled` option in the `/etc/selinux/config` file.

Starting in RHEL 9, disabling SELinux in the `/etc/selinux/config` file results in SELinux starting and performing active enforcement, but without loading any policies. Because policy rules define allowed actions, if no policies are loaded then all actions are denied. This behavior is intentional, and is designed to block malicious attempts to circumvent SELinux protection.

## Basic 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, which are known as *Mandatory Access Control (MAC)* because 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 that accesses that port might try to compromise a system through an existing vulnerability. If an example 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 a label called an *SELinux context*. 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`.

<code>SELinux User</code>	<code>Role</code>	<code>Type</code>	<code>Level</code>	<code>File</code>
<code>unconfined_u:object_r:httpd_sys_content_t:s0</code>		<code>/var/www/html/file2</code>		

Figure 5.1: 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.

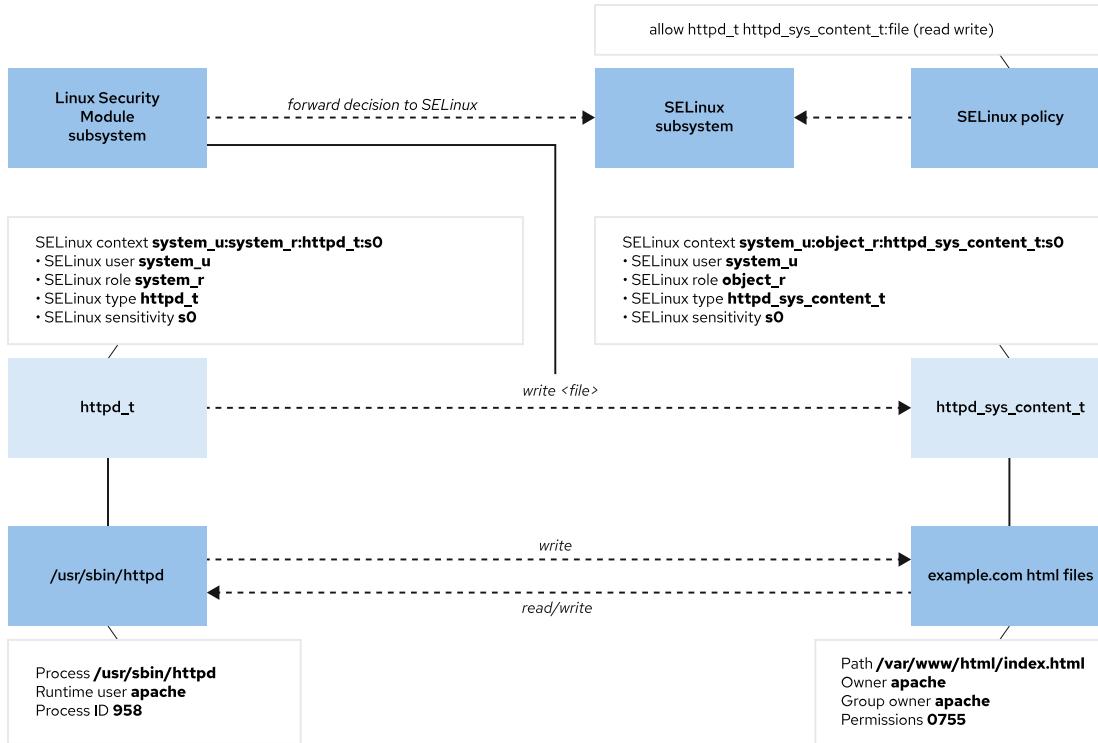


Figure 5.2: SELinux decision-making flow

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.

```
[root@host ~]# ps axZ
LABEL PID TTY STAT TIME COMMAND
system_u:system_r:kernel_t:s0 2 ? S 0:00 [kthreadd]
system_u:system_r:kernel_t:s0 3 ? I< 0:00 [rcu_gp]
system_u:system_r:kernel_t:s0 4 ? I< 0:00 [rcu_par_gp]
...output omitted...
[root@host ~]# systemctl start httpd
[root@host ~]# ps -ZC httpd
LABEL PID TTY TIME CMD
system_u:system_r:httpd_t:s0 1550 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1551 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1552 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1553 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1554 ? 00:00:00 httpd
```

```
[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. Use the `setenforce` command to change the SELinux mode.

```
[root@host ~]# getenforce
Enforcing
[root@host ~]# setenforce
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]
[root@host ~]# setenforce 0
[root@host ~]# getenforce
Permissive
[root@host ~]# setenforce Enforcing
[root@host ~]# getenforce
Enforcing
```

Alternatively, set the SELinux mode at boot time with a kernel parameter. Pass the `enforcing=0` kernel parameter to boot the system into permissive mode, or pass `enforcing=1` to boot into enforcing mode. Disable SELinux by passing the `selinux=0` kernel parameter, or pass `selinux=1` to enable SELinux.

Red Hat recommends to reboot the server when you change the SELinux mode from Permissive to Enforcing. This reboot ensures that the services started in 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 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...
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#   grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#   grubby --update-kernel ALL --remove-args selinux
#
```

```
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

## ► Guided Exercise

# Change the SELinux Enforcement Mode

In this lab, you manage SELinux modes, both temporarily and persistently.

## Outcomes

- View and set the current SELinux mode.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start selinux-opsmode
```

## Instructions

- 1. On the workstation machine, use the `ssh` command to log in to the `servera` machine as the `student` user and then switch to the `root` user.

```
[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 permissive.

- 2.1. Use the `getenforce` command to verify the current SELinux mode on the `servera` machine.

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

- 2.2. Use the `vim /etc/selinux/config` command to edit the configuration file. Change the `SELINUX` parameter from `enforcing` to `permissive` mode.

```
[root@servera ~]# vim /etc/selinux/config
```

- 2.3. Use the `grep` command to confirm that the `SELINUX` parameter displays the `permissive` mode.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config  
SELINUX=permissive  
SELINUXTYPE=targeted
```

- 2.4. Use the **getenforce** command to confirm that the SELINUX parameter displays the **enforcing** mode.

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

- 2.5. Use the **setenforce** command to change the SELINUX mode to **permissive** mode and verify the change.

```
[root@servera ~]# setenforce 0  
[root@servera ~]# getenforce  
Permissive
```

- 3. Change the default SELinux mode back to the **enforcing** mode in the configuration file.

- 3.1. Use the **vim /etc/selinux/config** command to edit the configuration file. Change the SELINUX parameter from **permissive** to **enforcing** mode.

```
[root@servera ~]# vim /etc/selinux/config
```

- 3.2. Use the **grep** command to confirm that the SELINUX parameter sets the **enforcing** mode on booting.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config  
SELINUX=enforcing  
SELINUXTYPE=targeted
```

- 4. Set the SELinux mode to **enforcing** in the command line. Reboot the **servera** machine and verify the SELinux mode.

- 4.1. Use the **setenforce** command to set the current SELinux mode to the **enforcing** mode. Use the **getenforce** command to confirm that SELinux is set to the **enforcing** mode.

```
[root@servera ~]# setenforce 1  
[root@servera ~]# getenforce  
Enforcing
```

- 4.2. Reboot the **servera** machine to implement the persistent configuration.

```
[root@servera ~]# systemctl reboot  
Connection to servera closed by remote host.  
Connection to servera closed.  
[student@workstation ~]$
```

- 4.3. Log in to **servera** machine and verify the SELinux mode.

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

- ▶ 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, 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 selinux-opsmode
```

This concludes the section.

# Control SELinux File Contexts

## Objectives

Manage the SELinux policy rules that determine the default context for files and directories with the `semanage fcontext` command and apply the context defined by the SELinux policy to files and directories with the `restorecon` command.

## 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, then new files are 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, with the new context 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 a copy to retain the context label that was determined for the file's original location. For example, the `cp -p` command preserves all file attributes where possible, and the `cp -c` command preserves only SELinux contexts, during a copy.



### Note

Copying a file always creates a new 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 a new 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 `mv` 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. Both files receive the `user_tmp_t` context type.

Move the first file, and copy the second file, to the `/var/www/html` directory.

- For use by Rajkamal J rajkamal@redhat.com veerammarish89@gmail.com
- The moved file retains the file context that was labeled from the original /tmp directory.
  - The copied file has a new inode and inherits the SELinux context from the destination /var/www/html directory.

The ls -Z command displays the SELinux context of a file. Observe the label of the files that are created in the /tmp directory.

```
[root@host ~]# touch /tmp/file1 /tmp/file2
[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 ls -Zd command displays the SELinux context of the specified directory. Note the label on the /var/www/html directory and the files inside it.

```
[root@host ~]# ls -Zd /var/www/html/
system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
[root@host ~]# ls -Z /var/www/html/index.html
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

Move one file from the /tmp directory to the /var/www/html directory. Copy the other file to the same directory. Note the resulting label on each file.

```
[root@host ~]# mv /tmp/file1 /var/www/html/
[root@host ~]# cp /tmp/file2 /var/www/html/
[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
```

The moved file retained its original label and the copied file inherited the destination directory label. Although not important to this discussion, unconfined\_u is the SELinux user, object\_r is the SELinux role, and s0 is the (lowest possible) sensitivity level. Advanced SELinux configurations and features use these values.

## Change the SELinux Context

You change the SELinux context on files with the semanage fcontext, restorecon, and chcon commands.

The recommended method to set the context for a file is to create a file context policy by using the semanage fcontext command, and then apply the specified context in the policy to the file by using the restorecon command. This method ensures that you can easily 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 easily correct the context on a set of files.

The chcon command sets SELinux context directly on files, but without referencing the system's SELinux policy. Although chcon is useful for testing and debugging, setting contexts manually with this method is temporary. File contexts that are set manually survive a reboot, but might be replaced if you run restorecon 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 `restorecon` on a file, any context that you set manually on the file is replaced if it does not match the rules in the SELinux policy.

The following example creates a directory with a `default_t` SELinux context, which it inherited from the `/` parent directory.

```
[root@host ~]# mkdir /virtual
[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
[root@host ~]# ls -Zd /virtual
unconfined_u:object_r:httpd_sys_content_t:s0 /virtual
```

Running 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
[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 called *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 beginning with a slash and followed by any number of characters, where the set can either exist or not exist". Stated more simply, this syntax matches the directory itself, even when empty, but 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 their subdirectories, and so on), should 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
```

## 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.

### `semanage fcontext commands`

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, check 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 will have the `httpd_sys_content_t` context by default. Running `restorecon` command on the wildcarded folder 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
[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 with an `index.html` file inside it. View the SELinux context for the file and the directory.

```
[root@host ~]# mkdir /virtual  
[root@host ~]# touch /virtual/index.html  
[root@host ~]# ls -Zd /virtual/  
unconfined_u:object_r:default_t:s0 /virtual  
[root@host ~]# ls -Z /virtual/  
unconfined_u:object_r:default_t:s0 index.html
```

Next, 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  
[root@host ~]# ls -Zd /virtual/  
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/  
[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 -l -C` command 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

## ► Guided Exercise

# Control SELinux File Contexts

In this lab, you persistently change the SELinux context of a directory and its contents.

## Outcomes

- Configure the Apache HTTP server to publish web content from a non-standard document root.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start selinux-filecontexts
```

## Instructions

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

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

- 2. Configure Apache to use a document directory in a non-standard location.

- 2.1. Create the `/custom` directory.

```
[root@servera ~]# mkdir /custom
```

- 2.2. Create the `index.html` file in the `/custom` directory. The `index.html` file should contain 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. You can use the `vim /etc/httpd/conf/httpd.conf` command to do so. The following example shows the expected content of the `/etc/httpd/conf/httpd.conf` file.

```
[root@servera ~]# cat /etc/httpd/conf/httpd.conf
...output omitted...
DocumentRoot "/custom"
...output omitted...
<Directory "/custom">
...output omitted...
```

- 3. Start and enable the Apache web service and 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; vendor
  preset: disabled)
    Active: active (running) since Wed 2022-04-06 05:21:19 EDT; 22s ago
      Docs: man:httpd.service(8)
     Main PID: 1676 (httpd)
...output omitted...
Apr 06 05:21:19 servera.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
Apr 06 05:21:19 servera.lab.example.com systemd[1]: Started The Apache HTTP
Server.
Apr 06 05:21:19 servera.lab.example.com httpd[1676]: Server configured, listening
on: port 80
```

- 4. Open a web browser on `workstation` and 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.
- 5. To permit access to the `index.html` file on `servera`, you must configure the SELinux context. Define an SELinux file context rule that sets the context type to `httpd_sys_content_t` for the `/custom` directory and all the files under it.

```
[root@servera ~]# semanage fcontext -a \
-t httpd_sys_content_t '/custom(/.*)?'
```

- 6. Correct the file contexts in 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
```

- ▶ 7. Try to view `http://servera/index.html` again in the web browser on the **workstation** machine. You should 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, 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 selinux-filecontexts
```

This concludes the section.

# Adjust SELinux Policy with Booleans

## Objectives

Activate and deactivate SELinux policy rules with the `setsebool` command, manage the persistent value of SELinux Booleans with the `semanage boolean -l` command, and consult `man` pages that end with `_selinux` to find useful information about 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 in 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 option makes the setting persistent by writing to the policy file. Only privileged users can set SELinux Booleans.

```
[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 `httpd`. 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 `https`, by default, and are not available to the user through a browser.

```
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
```

You can enable sharing and allow 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 will 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.

```
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (off , off)  Allow httpd to enable homedirs
[root@host ~]# setsebool httpd_enable_homedirs on
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on , off)  Allow httpd to enable homedirs
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

To list only Booleans with a current setting that is different from the default setting at boot, use the `semanage boolean -l -C` command. This example has the same result as the previous example, without requiring the `grep` filtering.

```
[root@host ~]# semanage boolean -l -C
SELinux boolean           State  Default Description
httpd_enable_homedirs      (on , off)  Allow httpd to enable homedirs
```

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. The following example sets a persistent value, and then views the Boolean's information from the policy file.

```
[root@host ~]# setsebool -P httpd_enable_homedirs on
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs      (on , on)  Allow httpd to enable homedirs
```

Use the `semanage boolean -l -C` command again. The Boolean is displayed despite the appearance that the current and default settings are the same. However, the `-C` option matches when the current setting is different from the default setting from the last boot. For this `httpd_enable_homedirs` example, the original default boot setting was `off`.

```
[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

## ► Guided Exercise

# Adjust SELinux Policy with Booleans

In this exercise, you configure Apache to publish web content from users' home directories.

### Outcomes

- Configure Apache web service to publish web content from the user's home directory.

### Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start selinux-booleans
```

### Instructions

- 1. On the **workstation** machine, use the **ssh** command to log in to the **servera** machine as the **student** user and then switch to the **root** user.

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

- 2. 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.

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

- ▶ 4. Open another terminal window, and use the `ssh` command to log in to the `servera` machine as the `student` user. Create the `index.html` web content file in the `~/public_html` directory.
- 4.1. In another terminal window, use the `ssh` command to log in to the `servera` machine as the `student` user.

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

- 4.2. Use the `mkdir` command to create the `~/public_html` directory.

```
[student@servera ~]$ mkdir ~/public_html
```

- 4.3. Create the `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 with permissions that 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 ~
[student@servera ~]$ ls -ld ~
drwx--x--x. 16 student student 4096 Nov  3 09:28 /home/student
```

- ▶ 5. Open a web browser on the `workstation` machine and enter the `http://servera/~student/index.html` address. An error message states that you do not have permission to access the file.
- ▶ 6. Switch to the other terminal and use the `getsebool` command to see if 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...
```

- ▶ 7. Use the `setsebool` command to enable persistent access to the home directory for the `httpd` service.

```
[root@servera ~]# setsebool -P httpd_enable_homedirs on
```

- ▶ **8.** Verify that you can now see the This is student content on SERVERA. message in the web browser after entering the `http://servera/~student/index.html` address.
- ▶ **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, 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 selinux-booleans
```

This concludes the section.

# Investigate and Resolve SELinux Issues

## Objectives

Use SELinux log analysis tools and display useful information during SELinux troubleshooting with the `sealert` command.

## 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 will interact.
- The policy states the process type, and 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 useful 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 properly 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 are easily 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 check 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 troubleshoot 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 troubleshoot 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.

## Chapter 5 | Manage SELinux Security

Consider the following example sequence of commands on a standard Apache web server. You create /root/mypage and move it to the default Apache content folder (/var/www/html). Then, after starting the Apache service, you try to retrieve the file content.

```
[root@host ~]# touch /root/mypage
[root@host ~]# mv /root/mypage /var/www/html
[root@host ~]# systemctl start httpd
[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.

```
[root@host ~]# tail /var/log/audit/audit.log
...output omitted...
type=AVC msg=audit(1649249057.067:212): avc: denied { setattr } for pid=2332 comm="httpd" path="/var/www/html/mypage" dev="vda4" ino=9322502 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
...output omitted
[root@host ~]# tail /var/log/messages
...output omitted...
Apr 6 08:44:19 host setroubleshoot[2547]: SELinux is preventing /usr/sbin/httpd from setattr access on the file /var/www/html/mypage. For complete SELinux messages run: sealert -l 95f41f98-6b56-45bc-95da-ce67ec9a9ab7
...output omitted...
```

The `sealert` output describes the event including 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` output includes a confidence rating, which indicates the level of confidence that the given advice will mitigate the denial. However, that advice might not be appropriate for your scenario.

For example, if the AVC denial is because the denied file is in the wrong location, then advice that states either to adjust the file's context label, or to create a new policy for this location and action, is technically accurate, but 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 95f41f98-6b56-45bc-95da-ce67ec9a9ab7
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(1649249057.67:212): avc: denied { getattr }
for pid=2332 comm="httpd" path="/var/www/html/mypage"
dev="vda4" ino=9322502 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(1649249057.67:212): arch=x86_64 syscall=newfstatat
success=no exit=EACCES a0=fffffff9c a1=7fe9c00048f8 a2=7fe9ccfc8830 a3=100
items=0 ppid=2329 pid=2332 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,admin_home_t,file,getattr
```

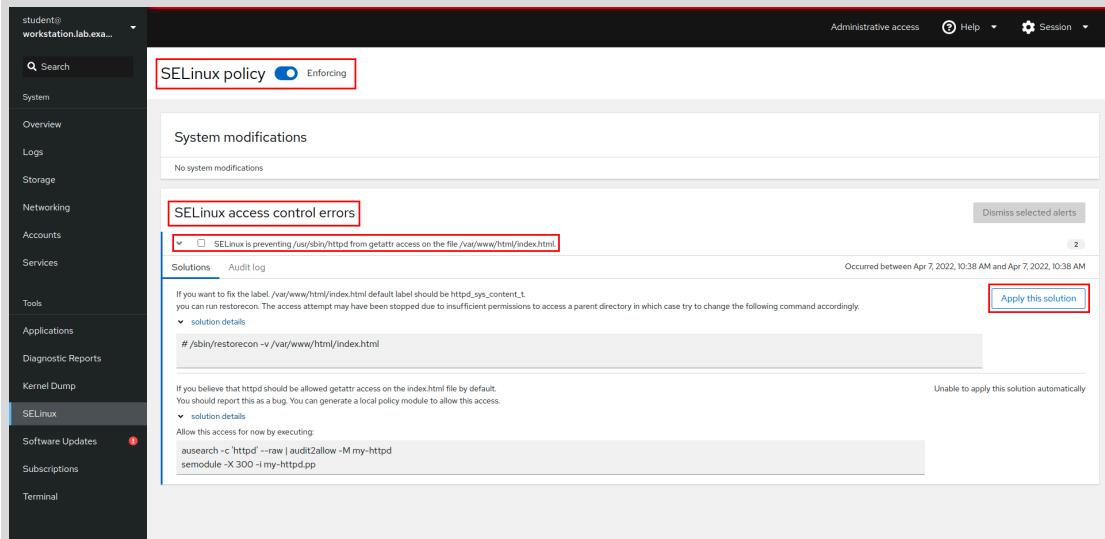
In this example, the accessed file is in the correct location, but does not have the correct SELinux file context. The Raw Audit Messages section displays information from the /var/log/audit.log event entry. Use the `restorecon /var/www/html/mypage` command to set the correct context label. To correct 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.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->Tue Apr  6 13:13:07 2019
type=PROCTITLE msg=audit(1554808387.778:4002):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1554808387.778:4002): arch=c000003e syscall=49
    success=no exit=-13 a0=3 a1=55620b8c9280 a2=10 a3=7ffed967661c items=0
    ppid=1 pid=9340 auid=4294967295 uid=0 gid=0 euid=0 suid=0 egid=0
    sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
    subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1554808387.778:4002): avc:  denied { name_bind } for
    pid=9340 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
```

## Troubleshoot SELinux Issues with the Web Console

The RHEL web console includes tools for troubleshooting SELinux issues. Select **SELinux** from the menu on the left. The SELinux policy window displays the current enforcing state. The **SELinux access control errors** section lists current SELinux issues.



**Figure 5.3: 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 the solution** to apply the suggested advice.

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

## ► Guided Exercise

# Investigate and Resolve SELinux Issues

In this lab, you learn how to troubleshoot SELinux security denials.

## Outcomes

- Gain experience with SELinux troubleshooting tools.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures 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.
- Use the `ssh` command to log in to `servera` as the `student` user. Use the `sudo -i` command to switch to the `root` user.

```
[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...
Apr  7 04:52:18 servera setroubleshoot[20715]: SELinux is preventing /usr/sbin/
httpd from getattr access on the file /custom/index.html. For complete SELinux
messages run: sealert -l 9a96294a-239b-4568-8f1e-9f35b5fb472b
...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 9a96294a-239b-4568-8f1e-9f35b5fb472b
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: NetworkManager_exec_t,
NetworkManager_log_t, NetworkManager_tmp_t, 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,
alsa_tmp_t, amanda_exec_t, amanda_log_t, amanda_recover_exec_t, amanda_tmp_t,
amtu_exec_t, anacron_exec_t, anon_inodefs_t
...output omitted...

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.lab.example.com
Source RPM Packages    httpd-2.4.51-7.el9_0.x86_64
Target RPM Packages    selinux-policy-targeted-34.1.27-1.el9.noarch
SELinux Policy RPM     selinux-policy-targeted-34.1.27-1.el9.noarch
Local Policy RPM       selinux-policy-targeted-34.1.27-1.el9.noarch
Selinux Enabled         True
Policy Type            targeted
Enforcing Mode         Enforcing
Host Name              servera.lab.example.com
Platform               Linux servera.lab.example.com
                        5.14.0-70.2.1.el9_0.x86_64 #1 SMP PREEMPT Wed Mar
                        16 18:15:38 EDT 2022 x86_64 x86_64
Alert Count             4
First Seen              2022-04-07 04:51:38 EDT
Last Seen               2022-04-07 04:52:13 EDT
Local ID                9a96294a-239b-4568-8f1e-9f35b5fb472b

Raw Audit Messages
```

```
type=AVC msg=audit(1649321533.406:1024): avc: denied { getattr } for
pid=20464 comm="httpd" path="/custom/index.html" dev="vda4" ino=25571802
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
tclass=file permissive=0

...output omitted...
```

- 4.2. Check the SELinux context for the directory from where the `httpd` service serves the content by default, `/var/www/html`. 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 Mar 21 11:47 /
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 the `/var/log/audit/audit.log` file. The `-m` option searches on the message type. The `-ts` option searches based on time. 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->Thu Apr 7 04:52:13 2022
type=PROCTITLE msg=audit(1649321533.406:1024):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1649321533.406:1024): arch=c000003e syscall=262 success=no
    exit=-13 a0=fffffff9c a1=7fefc403d850 a2=7fefc89bc830 a3=100 items=0 ppid=20461
    pid=20464 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(1649321533.406:1024): avc: denied
    { getattr } for pid=20464 comm="httpd" path="/custom/index.html"
    dev="vda4" ino=25571802 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.

```
[root@servera ~]# semanage fcontext -a \
-t httpd_sys_content_t '/custom(/.*)?'
[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. Again, attempt to view `http://servera/index.html`. The `This is SERVERA` message is displayed.
- ▶ 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, 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 selinux-issues
```

This concludes the section.

## ▶ Lab

# Manage SELinux Security

In this lab, you identify issues in system log files and adjust the SELinux configuration.

## Outcomes

- Identify issues in system log files.
- Adjust the SELinux configuration.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures 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.
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.
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, 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 selinux-review
```

This concludes the section.

## ► Solution

# Manage SELinux Security

In this lab, you identify issues in system log files and adjust the SELinux configuration.

### Outcomes

- Identify issues in system log files.
- Adjust the SELinux configuration.

### Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures 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.

```
[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...
Apr  7 06:16:15 serverb setroubleshoot[26509]: failed to retrieve rpm info for /
lab-content/la
b.html
Apr  7 06:16:17 serverb setroubleshoot[26509]: SELinux is preventing /usr/sbin/
httpd from setattr access on the file /lab-content/lab.html. For complete SELinux
messages run: sealert -l 35c9e452-2552-4ca3-8217-493b72ba6d0b
Apr  7 06:16:17 serverb setroubleshoot[26509]: SELinux is preventing /usr/sbin/
httpd from setattr 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 35c9e452-2552-4ca3-8217-493b72ba6d0b
SELinux is preventing /usr/sbin/httpd from setattr access on the file /lab-
content/lab.html.

***** Plugin catchall_labels (83.8 confidence) suggests *****

If you want to allow httpd to have setattr 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
Port                  <Unknown>
Host                  serverb.lab.example.com
Source RPM Packages    httpd-2.4.51-7.el9_0.x86_64
Target RPM Packages    selinux-policy-targeted-34.1.27-1.el9.noarch
SELinux Policy RPM     selinux-policy-targeted-34.1.27-1.el9.noarch
Local Policy RPM       selinux-policy-targeted-34.1.27-1.el9.noarch
Selinux Enabled         True
Policy Type            targeted
Enforcing Mode         Enforcing
Host Name              serverb.lab.example.com
Platform               Linux serverb.lab.example.com
                        5.14.0-70.2.1.el9_0.x86_64 #1 SMP PREEMPT Wed Mar
                        16 18:15:38 EDT 2022 x86_64 x86_64
Alert Count             8
First Seen              2022-04-07 06:14:45 EDT
Last Seen               2022-04-07 06:16:12 EDT
Local ID                35c9e452-2552-4ca3-8217-493b72ba6d0b

Raw Audit Messages
```

```

type=AVC msg=audit(1649326572.86:407): avc: denied { getattr } for
pid=10731 comm="httpd" path="/lab-content/lab.html" dev="vda4" ino=18192752
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
tclass=file permissive=0

type=SYSCALL msg=audit(1649326572.86:407): arch=x86_64 syscall=newfstatat
success=no exit=EACCES a0=fffffff9c a1=7f7c8c0457c0 a2=7f7c887f7830 a3=100 items=0
ppid=10641 pid=10731 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 the `/var/log/audit/audit.log` file. The `-m` option searches on the message type. The `-ts` option searches based on time. The following entry identifies the relevant process and file that cause 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 Apr  7 06:16:12 2022
type=PROCTITLE msg=audit(1649326572.086:407):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1649326572.086:407): arch=c000003e syscall=262 success=no
    exit=-13 a0=fffffff9c a1=7f7c8c0457c0 a2=7f7c887f7830 a3=100 items=0 ppid=10641
    pid=10731 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(1649326572.086:407): avc: denied { getattr } for
    pid=10731 comm="httpd" path="/lab-content/lab.html" dev="vda4" ino=18192752
    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 under 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 -R /lab-content/
```

5. Verify that the Apache server can now serve web content.

5.1. Use your web browser to refresh the `http://serverb/lab.html` link. If the content is displayed, then your 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, 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 selinux-review
```

This concludes the section.

# Summary

---

- Use the `getenforce` and `setenforce` commands to manage the SELinux mode of a system.
- The `semanage` command manages SELinux policy rules. The `restorecon` command applies the context that the policy defines.
- Booleans are switches that change the behavior of the SELinux policy. You can enable or disable them to tune the policy.
- The `sealert` command displays useful information to help with SELinux troubleshooting.



## Chapter 6

# Tune System Performance

### Goal

Evaluate and control processes, set tuning parameters and adjust process scheduling priorities on a Red Hat Enterprise Linux system.

### Objectives

- Use commands to kill and communicate with processes, define the characteristics of a daemon process, and stop user sessions and processes.
- Define load average and determine resource-intensive server processes.
- Optimize system performance by selecting a tuning profile that the tuned daemon manages.
- Prioritize or deprioritize specific processes, with the nice and renice commands.

### Sections

- Kill Processes (and Guided Exercise)
- Monitor Process Activity (and Guided Exercise)
- Adjust Tuning Profiles (and Guided Exercise)
- Influence Process Scheduling (and Guided Exercise)

### Lab

- Tune System Performance

# Kill Processes

---

## Objectives

Use commands to kill and communicate with processes, define the characteristics of a daemon process, and stop user sessions and processes.

## Process Control with Signals

A *signal* is a software interrupt that is delivered to a process. Signals report events to an executing program. Events that generate a signal can be an error, an external event (an I/O request or an expired timer), or by the explicit use of a signal-sending command or keyboard sequence.

The following table lists the fundamental signals that system administrators use for routine process management. Refer to signals by their short (HUP) or proper (SIGHUP) name.

### Fundamental process management signals

Signal	Name	Definition
1	HUP	Hangup : Reports termination of the controlling process of a terminal. Also requests process re-initialization (configuration reload) without termination.
2	INT	Keyboard interrupt : Causes program termination. It can be blocked or handled. Sent by pressing the INTR (Interrupt) key sequence ( <b>Ctrl+c</b> ).
3	QUIT	Keyboard quit : Similar to SIGINT; adds a process dump at termination. Sent by pressing the QUIT key sequence ( <b>kbd:[Ctrl+\]</b> ).
9	KILL	Kill, unblockable : Causes abrupt program termination. It cannot be blocked, ignored, or handled; consistently fatal.
15 <i>default</i>	TERM	Terminate : Causes program termination. Unlike SIGKILL, it can be blocked, ignored, or handled. The "clean" way to ask a program to terminate; it allows the program to complete essential operations and self-clean up before termination.
18	CONT	Continue : Sent to a process to resume if stopped. It cannot be blocked. Even if handled, it always resumes the process.
19	STOP	Stop, unblockable : Suspends the process. It cannot be blocked or handled.
20	TSTP	Keyboard stop : Unlike SIGSTOP, it can be blocked, ignored, or handled. Sent by pressing the suspend key sequence ( <b>Ctrl+z</b> ).

**Note**

Signal numbers vary between Linux hardware platforms, but signal names and meanings are standard. It is advised to use signal names rather than numbers when signaling. The numbers that are discussed in this section are for x86\_64 architecture systems.

Each signal has a *default action*, which is usually one of the following actions:

- **Term** : Terminate a program (exit) at once.
- **Core** : Save a program's memory image (core dump), then terminate.
- **Stop** : Stop a running program (suspend) and wait to continue (resume).

Programs react to the expected event signals by implementing handler routines to ignore, replace, or extend a signal's default action.

## Send Signals by Explicit Request

You can signal the current foreground process by pressing a keyboard control sequence to suspend (**Ctrl+z**), kill (**Ctrl+c**), or core dump (**Ctrl+\**) the process. However, you might use signal-sending commands to send signals to background processes in a different session.

You can specify signals either by name (for example, **-HUP** or **-SIGHUP** options) or by number (the related **-1** option). Users can kill their processes, but root privilege is required to kill processes that other users own.

The **kill** command uses a PID number to send a signal to a process. Despite its name, you can use the **kill** command to send any signal, not just those signals for terminating programs. You can use the **kill** command **-l** option to list the names and numbers of all available signals.

```
[user@host ~]$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
 11) SIGSEGV    12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
 16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
 ...output omitted...
[user@host ~]$ ps aux | grep job
5194 0.0 0.1 222448 2980 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/control job1
5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/control job2
5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/control job3
5430 0.0 0.0 221860 1096 pts/1 S+ 16:41 0:00 grep --color=auto job
[user@host ~]$ kill 5194
[user@host ~]$ ps aux | grep job
user 5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/
user/bin/control job2
user 5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/
user/bin/control job3
user 5783 0.0 0.0 221860 964 pts/1 S+ 16:43 0:00 grep --color=auto
job
[1] Terminated          control job1
[user@host ~]$ kill -9 5199
[user@host ~]$ ps aux | grep job
user 5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/
user/bin/control job3
```

```

user  5930  0.0  0.0 221860  1048 pts/1    S+   16:44   0:00 grep --color=auto
      job
[2]- Killed                  control job2
[user@host ~]$ kill -SIGTERM 5205
user  5986  0.0  0.0 221860  1048 pts/1    S+   16:45   0:00 grep --color=auto job
[3]+ Terminated              control job3

```

The `killall` command can signal multiple processes, based on their command name.

```

[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1    S     16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1    S     16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1    S     16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1    S+   16:41   0:00 grep --color=auto job
[user@host ~]$ killall control
[1]  Terminated              control job1
[2]- Terminated              control job2
[3]+ Terminated              control job3
[user@host ~]$ 

```

Use the `pkill` command to signal one or more processes that match selection criteria. Selection criteria can be a command name, a process owned by a specific user, or all system-wide processes. The `pkill` command includes advanced selection criteria:

- **Command**: Processes with a pattern-matched command name.
- **UID**: Processes owned by a Linux user account, effective or real.
- **GID**: Processes owned by a Linux group account, effective or real.
- **Parent**: Child processes of a specific parent process.
- **Terminal**: Processes that run on a specific controlling terminal.

```

[user@host ~]$ ps aux | grep pkill
user  5992  0.0  0.1 222448  3040 pts/1    S     16:59   0:00 /bin/bash /home/
user/bin/control pkill1
user  5996  0.0  0.1 222448  3048 pts/1    S     16:59   0:00 /bin/bash /home/
user/bin/control pkill2
user  6004  0.0  0.1 222448  3048 pts/1    S     16:59   0:00 /bin/bash /home/
user/bin/control pkill3
[user@host ~]$ pkill control
[1]  Terminated              control pkill1
[2]- Terminated              control pkill2
[user@host ~]$ ps aux | grep pkill
user  6219  0.0  0.0 221860  1052 pts/1    S+   17:00   0:00 grep --color=auto
pkill
[3]+ Terminated              control pkill3
[user@host ~]$ ps aux | grep test
user  6281  0.0  0.1 222448  3012 pts/1    S     17:04   0:00 /bin/bash /home/
user/bin/control test1
user  6285  0.0  0.1 222448  3128 pts/1    S     17:04   0:00 /bin/bash /home/
user/bin/control test2
user  6292  0.0  0.1 222448  3064 pts/1    S     17:04   0:00 /bin/bash /home/
user/bin/control test3

```

```

user 6318 0.0 0.0 221860 1080 pts/1 S+ 17:04 0:00 grep --color=auto
test
[user@host ~]$ pkill -U user
[user@host ~]$ ps aux | grep test
user 6870 0.0 0.0 221860 1048 pts/0 S+ 17:07 0:00 grep --color=auto
test

```

## Administratively Log Out Users

You might need to log out other users for various reasons. Some possible scenarios: the user committed a security violation; the user might have overused resources; the user has an unresponsive system; or the user has improper access to materials. In these cases, you must terminate their session by using signals administratively.

First, to log off a user, identify the login session to be terminated. Use the `w` command to list user logins and currently running processes. Note the TTY and FROM columns to determine the closing sessions.

All user login sessions are associated with a terminal device (TTY). If the device name is `pts/N`, then it is a *pseudo-terminal* that is associated with a graphical terminal window or remote login session. If it is `ttyN`, then the user is on a system console, alternative console, or another directly connected terminal device.

```

[user@host ~]$ w
12:43:06 up 27 min, 5 users, load average: 0.03, 0.17, 0.66
USER      TTY      FROM          LOGIN@    IDLE     JCPU     PCPU WHAT
root      tty2          12:26   14:58   0.04s   0.04s -bash
bob       tty3          12:28   14:42   0.02s   0.02s -bash
user      pts/1  desktop.example.com 12:41   2.00s   0.03s   0.03s w
[user@host ~]$

```

Discover how long a user has been on the system by viewing the session login time. CPU resources that current jobs consume, including background tasks and child processes, are in the JCPU column for each session. Current foreground process CPU consumption is in the PCPU column.

Processes and sessions can be individually or collectively signaled. To terminate all processes for one user, use the `pkill` command. Because the initial process in a login session (*session leader*) is designed to handle session termination requests and to ignore unintended keyboard signals, killing all of a user's processes and login shells requires the SIGKILL signal.



### Important

Administrators commonly use SIGKILL.

It is always fatal, because the SIGKILL signal cannot be handled or ignored. However, it forces termination without allowing the killed process to run self-cleanup routines. Red Hat recommends to send SIGTERM first, and then to try SIGINT; and only if both fail, to try again with SIGKILL.

First, use the `pgrep` command to identify the PID numbers to kill. This command operates similarly to the `pkill` command, including the same options, except that the `pgrep` command lists processes rather than killing them.

```
[root@host ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@host ~]# pkill -SIGKILL -u bob
[root@host ~]# pgrep -l -u bob
```

When processes that require attention are in the same login session, killing all of a user's processes might not be needed. Use the w command to determine the controlling terminal for the session, and then kill only the processes that reference the same terminal ID. Unless SIGKILL is specified, the session leader (here, the Bash login shell) successfully handles and survives the termination request but terminates all other session processes.

```
[root@host ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@host ~]# w -h -u bob
bob      tty3      18:37      5:04    0.03s  0.03s -bash
[root@host ~]# pkill -t tty3
[root@host ~]# pgrep -l -u bob
7391 bash
[root@host ~]# pkill -SIGKILL -t tty3
[root@host ~]# pgrep -l -u bob
[root@host ~]#
```

You can apply the same selective process termination with parent and child process relationships. Use the pstree command to view a process tree for the system or a single user. Use the parent process's PID to kill all children that they created. The parent Bash login shell survives this time because the signal is directed only at its child processes.

```
[root@host ~]# pstree -p bob
bash(8391)─sleep(8425)
             ├ sleep(8426)
             └ sleep(8427)
[root@host ~]# pkill -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
[root@host ~]# pkill -SIGKILL -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
```



## References

`kill(1)`, `killall(1)`, `pgrep(1)`, `pkill(1)`, `pstree(1)`, `signal(7)`, and `w(1)` man pages

For further reading, refer to *Signal Handling* at

<https://www.gnu.org/software/libc/manual/pdf/libc.pdf#Signal%20Handling>

For further reading, refer to *Processes* at

<https://www.gnu.org/software/libc/manual/pdf/libc.pdf#Processes>

## ► Guided Exercise

# Kill Processes

In this exercise, you use signals to manage and stop processes.

### Outcomes

- Start and stop multiple shell processes.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start processes-kill
```

### Instructions

- 1. On the workstation machine, open two terminal windows side by side. In this section, these terminals are referred to as *left* and *right*. In each terminal, use the `ssh` command to log in to the `servera` machine as the `student` user.

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

- 2. In the left terminal shell, create the `/home/student/bin` directory. Create the `instance` shell script in the new directory. Change the script permissions so that it is executable.

- 2.1. Create the `/home/student/bin` directory.

```
[student@servera ~]$ mkdir /home/student/bin
```

- 2.2. Create the `instance` script file in the `/home/student/bin` directory. Press the `i` key to enter Vim interactive mode. The file must have the following content as shown. Use the `:wq` command to save the file.

```
[student@servera ~]$ cd /home/student/bin  
[student@servera bin]$ vim /home/student/bin/instance  
#!/bin/bash  
while true; do  
    echo -n "$@" >> ~/instance_outfile  
    sleep 5  
done
```

**Note**

The `instance` script runs until the process is terminated. It appends command-line arguments to the `~/instance_outfile` file once every 5 seconds.

- 2.3. Make the `instance` script file executable.

```
[student@servera ~]$ chmod +x /home/student/bin/instance
```

- ▶ 3. In the left terminal shell, change into the `/home/student/bin/` directory. Start three processes with the `instance` script file, by passing the `network`, `interface`, and `connection` arguments. Start the processes in the background.

```
[student@servera bin]$ instance network &
[1] 3460
[student@servera bin]$ instance interface &
[2] 3482
[student@servera bin]$ instance connection &
[3] 3516
```

- ▶ 4. In the right terminal shell, verify that all three processes are appending content to the `~/home/student/instance_outfile` file.

```
[student@servera ~]$ tail -f ~/instance_outfile
network interface network connection interface network connection interface
network
...output omitted...
```

- ▶ 5. In the left terminal shell, List existing jobs.

```
[student@servera bin]$ jobs
[1]  Running           instance network &
[2]- Running           instance interface &
[3]+ Running           instance connection &
```

- ▶ 6. Use signals to suspend the `instance network` process. Verify that the `instance network` process is set to `Stopped`. Verify that the `network` process is no longer appending content to the `~/instance_output` file.

- 6.1. Stop the `instance network` process. Verify that the process is stopped.

```
[student@servera bin]$ kill -SIGSTOP %1
[1]+ Stopped           instance network
[student@servera bin]$ jobs
[1]+ Stopped           instance network
[2]  Running           instance interface &
[3]- Running           instance connection &
```

- 6.2. In the right terminal shell, view the `tail` command output. Verify that the word `network` is no longer appended to the `~/instance_outfile` file.

```
...output omitted...
interface connection interface connection interface connection interface
```

- 7. In the left terminal shell, terminate the `instance interface` process. Verify that the `instance interface` process disappeared. Verify that the `instance interface` process output is no longer appended to the `~/instance_outfile` file.

- 7.1. Terminate the `instance interface` process. Verify that the process is terminated.

```
[student@servera bin]$ kill -SIGTERM %2
[student@servera bin]$ jobs
[1]+  Stopped                  instance network
[2]  Terminated                instance interface
[3]-  Running                  instance connection &
```

- 7.2. In the right terminal shell, view the `tail` command output. Verify that the word `interface` is no longer appended to the `~/instance_outfile` file.

```
...output omitted...
connection connection connection connection connection connection
connection
```

- 8. In the left terminal shell, resume the `instance network` process. Verify that the `instance network` process is set to `Running`. Verify that the `instance network` process output is appended to the `~/instance_outfile` file.

- 8.1. Resume the `instance network` process. Verify that the process is in the `Running` state.

```
[student@servera bin]$ kill -SIGCONT %1
[student@servera bin]$ jobs
[1]+  Running                  instance network &
[3]-  Running                  instance connection &
```

- 8.2. In the right terminal shell, view the `tail` command output. Verify that the word `network` is appended to the `~/instance_outfile` file.

```
...output omitted...
network connection network connection network connection network connection
network connection
```

- 9. In the left terminal shell, terminate the remaining two jobs. Verify that no jobs remain and that output is stopped.

- 9.1. Terminate the `instance network` process. Next, terminate the `instance connection` process.

```
[student@servera bin]$ kill -SIGTERM %1
[student@servera bin]$ kill -SIGTERM %3
[1]+ Terminated instance network
[student@servera bin]$ jobs
[3]+ Terminated instance connection
```

- 10. In the left terminal shell, list the current running processes in all open terminal shells. Terminate the `tail` processes. Verify that the processes are no longer running.

10.1. List all current running processes. Refine the search to view only `tail` lines.

```
[student@servera bin]$ ps -ef | grep tail
student 4581 31358 0 10:02 pts/0    00:00:00 tail -f instance_outfile
student 4869 2252 0 10:33 pts/1    00:00:00 grep --color=auto tail
```

10.2. Terminate the `tail` process. Verify that the process is no longer running.

```
[student@servera bin]$ pkill -SIGTERM tail
[student@servera bin]$ ps -ef | grep tail
student 4874 2252 0 10:36 pts/1    00:00:00 grep --color=auto tail
```

10.3. In the right terminal shell, verify that the `tail` command is no longer running.

```
...output omitted...
network connection network connection network connection Terminated
[student@servera ~]$
```

- 11. Close the extra terminal. Return to the `workstation` system as the `student` user.

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

## 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 processes-kill
```

This concludes the section.

# Monitor Process Activity

## Objectives

Define load average and determine resource-intensive server processes.

### Describe Load Average

*Load average* is a measurement that Linux kernel provides, to represent the perceived system load for a period of time. It can be used as a rough gauge of how many system resource requests are pending, to determine whether system load increases or decreases.

The kernel collects the current load number every five seconds based on the number of processes in runnable and uninterruptible states. This number is accumulated and reported as an exponential moving average over the most recent 1, 5, and 15 minutes.

### Load Average Calculation

The load average represents the perceived system load for a period of time. Linux determines load average by reporting how many processes are ready to run on a CPU and how many processes are waiting for disk or network I/O to complete.

- The load number is a running average of the number of processes that are ready to run (in process state R) or are waiting for I/O to complete (in process state D).
- Some UNIX systems consider only CPU utilization or run queue length to indicate system load. Linux also includes disk or network utilization, because the high usage of these resources can significantly impact system performance as CPU load. For high load averages with minimal CPU activity, examine disk and network activity.

Load average is a rough measurement of how many processes are currently waiting for a request to complete before they do anything else. The request might be for CPU time to run the process. Alternatively, the request might be for a critical disk I/O operation to complete, and the process cannot be run on the CPU until the request completes, even though the CPU is idle. Either way, system load is impacted, and the system appears to run more slowly because processes are waiting to run.

### Interpret Load Average Values

The `uptime` command is one way to display the current load average. It prints the current time, how long the machine has been up, how many user sessions are running, and the current load average.

```
[user@host ~]$ uptime  
15:29:03 up 14 min,  2 users,  load average: 2.92, 4.48, 5.20
```

The three values for the load average represent the load over the last 1, 5, and 15 minutes. It indicates whether the system load appears to be increasing or decreasing.

If the main contribution to load average is from processes that are waiting for the CPU, then you can calculate the approximate per CPU load value to determine whether the system is experiencing significant waiting.

Use the `lscpu` command to determine the number of CPUs that are present on a system.

In the following example, the system is a dual-core single-socket system with two hyper threads per core. Roughly speaking, Linux treats this CPU configuration as a four-CPU system for scheduling purposes.

```
[user@host ~]$ lscpu
Architecture:           x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:  0-3
Thread(s) per core:   2
Core(s) per socket:   2
Socket(s):             1
NUMA node(s):          1
...output omitted...
```

For a moment, imagine that the only contribution to the load number is from processes that need CPU time. Then you can divide the displayed load average values by the number of logical CPUs in the system. A value below 1 indicates adequate resource utilization and minimal wait times. A value above 1 indicates resource saturation and some processing delay.

```
# From lscpu, the system has four logical CPUs, so divide by 4:
#                               load average: 2.92, 4.48, 5.20
#       divide by number of logical CPUs:    4    4    4
#                                         -----
#                               per-CPU load average: 0.73  1.12  1.30
#
# This system's load average appears to be decreasing.
# With a load average of 2.92 on four CPUs, all CPUs were in use ~73% of the time.
# During the last 5 minutes, the system was overloaded by ~12%.
# During the last 15 minutes, the system was overloaded by ~30%.
```

An idle CPU queue has a load number of 0. Each process that waits for a CPU adds a count of 1 to the load number. If one process is running on a CPU, then the load number is 1, the resource (the CPU) is in use, but no requests are waiting. If that process runs for an entire minute, then its contribution to the one-minute load average is 1.

However, processes that are uninterruptibly sleeping for critical I/O due to a busy disk or network resource are also included in the count and increase the load average. While not indicating CPU utilization, these processes are added to the queue count because they wait for resources and cannot run on a CPU until they get the resources. This metric is still considered as system load due to resource limitations that cause processes not to run.

Until resource saturation, a load average remains below 1 since tasks are seldom found waiting in the queue. Load average increases only when resource saturation causes requests to remain queued and are counted by the load calculation routine. When resource utilization approaches 100%, each additional request starts experiencing service wait time.

## Real-time Process Monitoring

The `top` command displays a dynamic view of the system's processes and a summary header followed by a process or thread list. Unlike the static `ps` command output, the `top` command continuously refreshes at a configurable interval and provides column reordering, sorting, and highlighting. You can make persistent changes to the `top` settings. The default `top` output columns are as follows:

- The process ID (PID).
- The process owner user name (USER).
- Virtual memory (VIRT) is all the memory that the process uses, including the resident set, shared libraries, and any mapped or swapped memory pages. (labeled VSZ in the `ps` command.)
- Resident memory (RES) is the physical memory that the process uses, including any resident, shared objects. (labeled RSS in the `ps` command.)
- Process state (S) can be one of the following states:
  - D = Uninterruptible Sleeping
  - R = Running or Runnable
  - S = Sleeping
  - T = Stopped or Traced
  - Z = Zombie
- CPU time (TIME) is the total processing time since the process started. It can be toggled to include a cumulative time of all previous children.
- The process command name (COMMAND).

### Fundamental Keystrokes in `top` Command

Key	Purpose
? or h	Help for interactive keystrokes.
l, t, m	Toggles for load, threads, and memory header lines.
1	Toggle for individual CPUs or a summary for all CPUs in the header.
s	Change the refresh (screen) rate, in decimal seconds (such as 0.5, 1, 5).
b	Toggle reverse highlighting for Running processes; default is bold only.
Shift+b	Enables bold use in display, in the header, and for <i>Running</i> processes.
Shift+h	Toggle threads; show process summary or individual threads.
u, Shift+u	Filter for any user name (effective, real).
Shift+m	Sort process listing by memory usage, in descending order.
Shift+p	Sort process listing by processor utilization, in descending order.
k	Kill a process. When prompted, enter PID, and then signal.
r	Renice a process. When prompted, enter PID, and then nice_value.

Key	Purpose
Shift+w	Write (save) the current display configuration for use at the next top restart.
q	Quit.
f	Manage the columns by enabling or disabling fields. You can also set the sort field for top.

**Note**

The s, k, and r keystrokes are not available when the top command is started in a secure mode.

**References**

ps(1), top(1), uptime(1), and w(1) man pages

## ► Guided Exercise

# Monitor Process Activity

In this exercise, you use the `top` command to examine running processes and control them dynamically.

### Outcomes

- Manage processes in real time.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start processes-monitor
```

### Instructions

- 1. On the `workstation` machine, open two terminal windows side by side. In this section, these terminals are referred to as *left* and *right*. In each terminal, log in to the `servera` machine as the `student` user.

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

- 2. In the left terminal shell, create the `/home/student/bin` directory. Create a shell script called `monitor` in the new directory to generate an artificial CPU load. Make the `monitor` script file executable.

- 2.1. Create the `/home/student/bin` directory.

```
[student@servera ~]$ mkdir /home/student/bin
```

- 2.2. Create the script file in the `/home/student/bin` directory with the content shown.

```
[student@servera ~]$ vim /home/student/bin/monitor  
#!/bin/bash  
while true; do  
    var=1  
    while [[ var -lt 60000 ]]; do  
        var=$($var+1)
```

```
done
sleep 1
done
```

**Note**

The `monitor` script runs until the process is terminated. It generates an artificial CPU load by performing sixty thousand addition calculations. After generating the CPU load, it then sleeps for one second, resets the variable, and repeats.

- 2.3. Make the `monitor` file executable.

```
[student@servera ~]$ chmod a+x /home/student/bin/monitor
```

- 3. In the right terminal shell, run the `top` command. Resize the window to view the contents of the command.

```
[student@servera ~]$ top
top - 12:13:03 up 11 days, 58 min, 3 users, load average: 0.00, 0.00, 0.00
Tasks: 113 total, 2 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.0 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1829.4 total, 1377.3 free, 193.9 used, 258.2 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1476.1 avail Mem

PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
5861 root      20   0        0      0      0 I  0.3  0.0  0:00.71 kworker/1:3-
events
6068 student   20   0  273564  4300  3688 R  0.3  0.2  0:00.01 top
  1 root      20   0  178680 13424  8924 S  0.0  0.7  0:04.03 systemd
  2 root      20   0        0      0      0 S  0.0  0.0  0:00.03 kthreadd
  3 root      0 -20        0      0      0 I  0.0  0.0  0:00.00 rcu_gp
...output omitted...
```

- 4. In the left terminal shell, verify the number of logical CPUs on this virtual machine.

```
[student@servera ~]$ lscpu
Architecture:           x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                2
...output omitted...
```

- 5. In the left terminal shell, run a single instance of the `monitor` script file in the background.

```
[student@servera ~]$ monitor &
[1] 6071
```

- 6. In the right terminal shell, monitor the `top` command. Use the single keystrokes `l`, `t`, and `m` to toggle the load, threads, and memory header lines. After observing this behavior, ensure that all headers are displayed.

- 7. Note the process ID (PID) for the `monitor` process. View the CPU percentage for the process, which is expected to hover around 15% to 25%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
071 student    20   0 222448  2964  2716 S 18.7  0.2  0:27.35 monitor
...output omitted...
```

View the load averages. The one-minute load average is currently less than a value of 1. The observed value might be affected by resource contention from another virtual machine or from the virtual host.

```
top - 12:23:45 up 11 days,  1:09,  3 users,  load average: 0.21, 0.14, 0.05
```

- 8. In the left terminal shell, run a second instance of the `monitor` script file in the background.

```
[student@servera ~]$ monitor &
[2] 6498
```

- 9. In the right terminal shell, note the process ID (PID) for the second `monitor` process. View the CPU percentage for the process, which is also expected to hover between 15% and 25%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
6071 student    20   0 222448  2964  2716 S 19.0  0.2  1:36.53 monitor
6498 student    20   0 222448  2996  2748 R 15.7  0.2  0:16.34 monitor
...output omitted...
```

Again view the one-minute load average, which remains less than 1. Wait at least one minute for the calculation to adjust to the new workload.

```
top - 12:27:39 up 11 days,  1:13,  3 users,  load average: 0.36, 0.25, 0.11
```

- 10. In the left terminal shell, run a third instance of the `monitor` script file in the background.

```
[student@servera ~]$ monitor &
[3] 6881
```

- 11. In the right terminal shell, note the process ID (PID) for the third `monitor` process. View the CPU percentage for the process, which is again expected to hover between 15% and 25%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
6881 student    20   0 222448  3032  2784 S 18.6  0.2  0:11.48 monitor
6498 student    20   0 222448  2996  2748 S 15.6  0.2  0:47.86 monitor
6071 student    20   0 222448  2964  2716 S 18.1  0.2  2:07.86 monitor
```

To push the load average above 1, you must start more `monitor` processes. The classroom setup has two CPUs, so only three processes are not enough to stress it. Start three more

monitor processes in the background. View again the one-minute load average, which is now expected to be above 1. Wait at least one minute for the calculation to adjust to the new workload.

```
[student@servera ~]$ monitor &
[4] 10708
[student@servera ~]$ monitor &
[5] 11122
[student@servera ~]$ monitor &
[6] 11338
```

```
top - 12:42:32 up 11 days, 1:28, 3 users, load average: 1.23, 2.50, 1.54
```

- ▶ 12. When you are finished observing the load average values, terminate each of the monitor processes from within the top command.

- 12.1. In the right terminal shell, press k to observe the prompt below the headers and above the columns.

```
...output omitted...
PID to signal/kill [default pid = 11338]
```

- 12.2. The prompt chooses the monitor processes at the top of the list. Press Enter to kill the process.

```
...output omitted...
Send pid 11338 signal [15/sigterm]
```

- 12.3. Press Enter again to confirm the SIGTERM signal 15.

Verify that the selected process is no longer present in the top command. If the PID exists, then repeat these steps to terminate the processes, and substitute SIGKILL signal 9 when prompted.

PID	User	Time	Process ID	State	Processor	Load Average	Memory Usage	Filesystem	Command
6498	student	20	0	222448	2996	2748 R	22.9	0.2	5:31.47 monitor
6881	student	20	0	222448	3032	2784 R	21.3	0.2	4:54.47 monitor
11122	student	20	0	222448	2984	2736 R	15.3	0.2	2:32.48 monitor
6071	student	20	0	222448	2964	2716 S	15.0	0.2	6:50.90 monitor
10708	student	20	0	222448	3032	2784 S	14.6	0.2	2:53.46 monitor

- ▶ 13. Repeat the previous step for each remaining monitor process. Verify that no monitor processes remain in the top command.

- ▶ 14. In the right terminal shell, press q to exit the top command. Close the right terminal.

- ▶ 15. Return to the workstation machine as the student user.

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

## 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 processes-monitor
```

This concludes the section.

# Adjust Tuning Profiles

## Objectives

Optimize system performance by selecting a tuning profile that the tuned daemon manages.

## Tune Systems

System administrators optimize the performance of a system by adjusting device settings based on various use case workloads. The tuned daemon applies tuning adjustments both statically and dynamically by using tuning profiles that reflect particular workload requirements.

### Configure Static Tuning

The tuned daemon applies system settings when the service starts or on selecting a new tuning profile. Static tuning configures predefined kernel parameters in profiles that the tuned daemon applies at runtime. With static tuning, the tuned daemon sets kernel parameters for overall performance expectations, without adjusting these parameters as activity levels change.

### Configure Dynamic Tuning

With dynamic tuning, the tuned daemon monitors system activity and adjusts settings according to runtime behavior changes. Dynamic tuning continuously adjusts tuning to fit the current workload, starting with the initial declared settings in your selected tuning profile.

For example, storage devices experience high use during startup and log in, but have minimal activity when user workloads consist of using web browsers and email clients. Similarly, CPU and network devices experience activity increases during peak usage throughout a workday. The tuned daemon monitors the activity of these components and adjusts parameter settings to maximize performance during high-activity times and reduce settings during low activity. Predefined tuning profiles provide performance parameters that the tuned daemon uses.

To monitor and adjust parameter settings, the tuned daemon uses modules called monitor and tuning *plug-ins*, respectively.

Monitor plug-ins analyze the system and obtain information from it, so the tuning plug-ins use this information for dynamic tuning. At this moment, the tuned daemon ships with three different monitor plug-ins:

- **disk**: Monitors the disk load based on the number of IO operations for every disk device.
- **net**: Monitors the network load based on the number of transferred packets per network card.
- **load**: Monitors the CPU load for every CPU.

Tuning plug-ins tune the individual subsystems. They use the data obtained by the monitor plug-ins and the performance parameters provided by the predefined tuning profiles. Among others, the tuned daemon ships with the following tuning plug-ins:

- **disk**: Sets different disk parameters, for example, the disk scheduler, the spin-down timeout, or the advanced power management.
- **net**: Configures the interface speed and the Wake-on-LAN (WoL) functionality.
- **cpu**: Sets different CPU parameters, for example, the CPU governor or the latency.

By default, dynamic tuning is disabled. You can enable it by setting the `dynamic_tuning` variable to 1 in the `/etc/tuned/tuned-main.conf` configuration file. If you enable dynamic tuning, then the `tuned` daemon monitors periodically the system and adjust the tuning settings to runtime behavior changes. You can set the time in seconds between updates by using the `update_interval` variable in the `/etc/tuned/tuned-main.conf` configuration file.

```
[root@host ~]$ cat /etc/tuned/tuned-main.conf
...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...
```

## The tuned Utility

A minimal Red Hat Enterprise Linux installation includes the `tuned` package by default. You can install and enable the package manually by using the following commands:

```
[root@host ~]$ dnf install tuned
...output omitted...
[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 include profiles that focus on the following aspects:

- Low latency for storage and network
- High throughput for storage and network
- Virtual machine performance
- Virtualization host performance

The next table shows a list of the tuning profiles distributed with Red Hat Enterprise Linux 9.

**Tuning Profiles Distributed with Red Hat Enterprise Linux 9**

Tuned Profile	Purpose
balanced	Ideal for systems that require a compromise between power saving and performance.
powersave	Tunes the system for maximum power saving.
throughput-performance	Tunes the system for maximum throughput.
accelerator-performance	Tunes the same as <code>throughput-performance</code> , and also reduces the latency to less than 100 µs.

Tuned Profile	Purpose
latency-performance	Ideal for server systems that require low latency at the expense of power consumption.
network-throughput	Derived from the throughput-performance profile. Additional network tuning parameters are applied for maximum network throughput.
network-latency	Derived from the latency-performance profile. Enables additional network tuning parameters to provide low network latency.
desktop	Derived from the balanced profile. Provides faster response of interactive applications.
hpc-compute	Derived from the latency-performance profile. Ideal for high-performance computing.
virtual-guest	Tunes the system for maximum performance if it runs on a virtual machine.
virtual-host	Tunes the system for maximum performance if it acts as a host for virtual machines.
intel-sst	Optimized for systems with Intel Speed Select Technology configurations. Use it as an overlay on other profiles.
optimize-serial-console	Increases responsiveness of the serial console. Use it as an overlay on other profiles.

The tuned application stores the tuning profiles under the /usr/lib/tuned and /etc/tuned directories. Every profile has a separate directory, and inside the directory the tuned.conf main configuration file and, optionally, other files.

```
[root@host ~]# cd /usr/lib/tuned
[root@host tuned]# ls
accelerator-performance  hpc-compute          network-throughput      throughput-
performance              intel-sst             optimize-serial-console virtual-
balanced                 guest                latency-performance   virtual-
functions               host                powersave            virtual-
host                   functions            network-latency       recommend.d
[root@host tuned]$ ls virtual-guest
tuned.conf
```

A typical tuned.conf configuration file looks as follows:

```
[root@host tuned]# cat virtual-guest/tuned.conf
#
# tuned configuration
#
```

```
[main]
summary=Optimize for running inside a virtual guest
include=throughput-performance

[sysctl]
# 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%)
vm.dirty_ratio = 30

# 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 might include a summary of the tuning profile. This section also accepts the `include` parameter, that you can use to make the profile inherit all the settings from the referenced profile.

This is very useful when creating new tuning profiles, because you can use one of the provided profiles as a basis and then add or modify the parameters that you want to configure. To create or modify tuning profiles, copy the tuning profile files from the `/usr/lib/tuned` directory to the `/etc/tuned` directory and then modify them. In case there are profile directories with the same name under the `/usr/lib/tuned` and `/etc/tuned` directories, the latter always take precedence. Thus, never directly modify files in the `/usr/lib/tuned` system directory.

The rest of the sections in the tuned.conf file use the tuning plug-ins to modify parameters in the system. In the previous example, the [sysctl] section modifies the `vm.dirty_ratio` and `vm.swappiness` kernel parameters through the `sysctl` plug-in.

## Manage Profiles from the Command Line

Use the `tuned-adm` command to change the settings of the tuned daemon. The `tuned-adm` command queries current settings, lists available profiles, recommends a tuning profile for the system, changes profiles directly, or turns off tuning.

You can identify the currently active tuning profile with the `tuned-adm active` command.

```
[root@host ~]# tuned-adm active
Current active profile: virtual-guest
```

The `tuned-adm list` command lists all available tuning profiles, including both built-in profiles and the custom-created tuning profiles.

```
[root@host ~]# tuned-adm list
Available profiles:
- accelerator-performance      - Throughput performance based tuning with ...
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- hpc-compute                 - Optimize for HPC compute workloads
```

**Chapter 6 |** Tune System Performance

```
- intel-sst           - Configure for Intel Speed Select Base Frequency
- latency-performance - Optimize for deterministic performance at ...
- network-latency    - Optimize for deterministic performance at ...
...output omitted...
Current active profile: virtual-guest
```

Use the `tuned-adm profile profilename` command to switch to a different active profile that better matches the system's current tuning requirements.

```
[root@host ~]$ tuned-adm profile throughput-performance
[root@host ~]$ tuned-adm active
Current active profile: throughput-performance
```

The `tuned-adm recommend` command can recommend a tuning profile for the system. The system uses this mechanism to determine the default profile after its installation.

```
[root@host ~]$ tuned-adm recommend
virtual-guest
```

**Note**

The `tuned-adm recommend` command bases its recommendation on various system characteristics, including whether the system is a virtual machine and other predefined selected categories during system installation.

To revert the setting changes that the current profile applied, either switch to another profile or deactivate the tuned daemon. Turn off the tuned application tuning activity by using the `tuned-adm off` command.

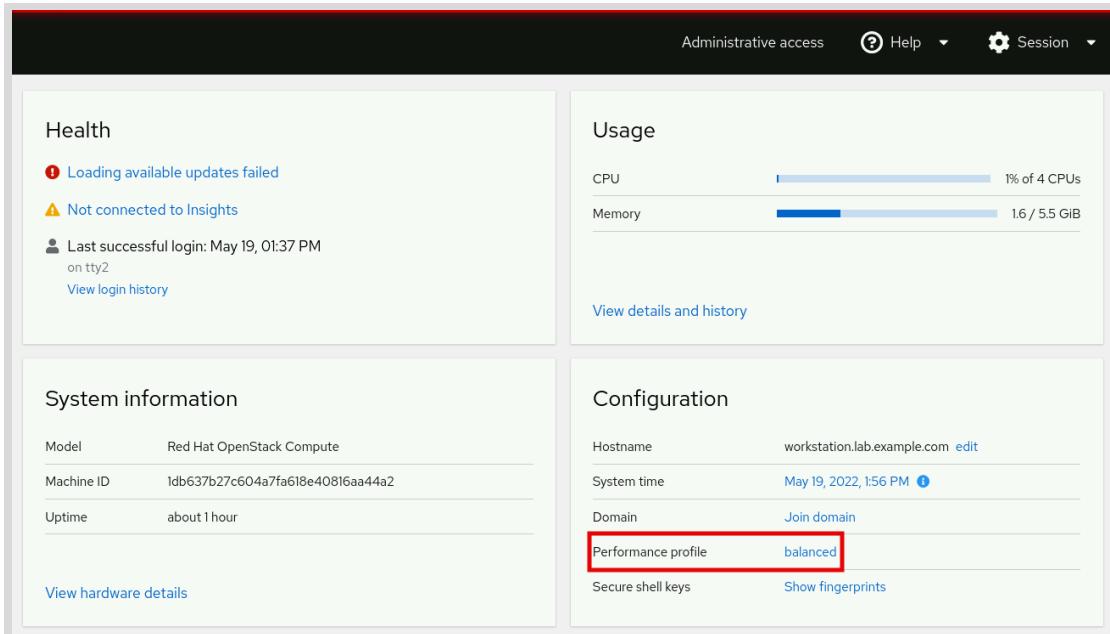
```
[root@host ~]$ tuned-adm off
[root@host ~]$ tuned-adm active
No current active profile.
```

## Manage Profiles with the Web Console

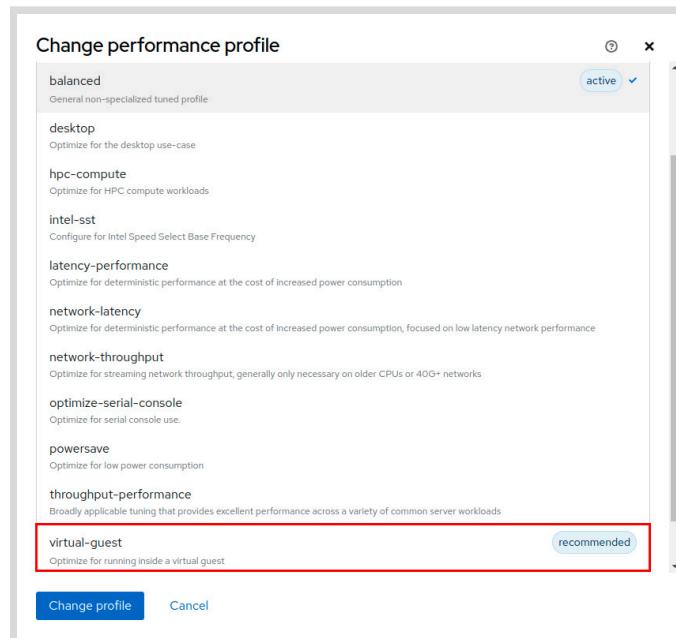
To manage system performance profiles with the web console, you need to log in and escalate privileges. Privilege escalation mode permits the user to execute commands, with administrative privileges, that modify system performance profiles. Because changing tuning profiles modifies some system parameters, you need to do it with administrative privileges.

You can switch to the administrative access mode in the web console by clicking the **Limited access** or **Turn on administrative access** buttons. Then, enter your password when prompted. After you escalate privileges, the **Limited access** button changes to **Administrative access**. As a security reminder, remember to always toggle back to limited access mode once you perform on your system the task that requires administrative privileges.

As a privileged user, click the **Overview** menu option in the left navigation bar. The **Performance profile** field displays the current active profile.

**Figure 6.1: 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 to select one that best suits the system purpose and click the **Change profile** button.

**Figure 6.2: Select a preferred performance profile**

To verify changes, return to the main **Overview** page and confirm that it displays the active profile in the **Performance profile** field.



## References

tuned(8), tuned.conf(5), tuned-main.conf(5), and tuned-adm(1) man pages

For more information, refer to the *Monitoring and Managing System Status and Performance* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/monitoring\\_and\\_managing\\_system\\_status\\_and\\_performance/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/monitoring_and_managing_system_status_and_performance/index)

## ► Guided Exercise

# Adjust Tuning Profiles

In this exercise, you tune server performance by activating the tuned service and applying a tuning profile.

### Outcomes

- Configure a system to use a tuning profile.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start tuning-profiles
```

### Instructions

- 1. Log in to `servera` as the student user.

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

- 2. Verify that the tuned package is installed, enabled, and started.

- 2.1. Verify that the tuned package is installed.

```
[student@servera ~]$ dnf list tuned  
...output omitted...  
Installed Packages  
tuned.noarch           2.18.0-1.el9          @System
```

- 2.2. Verify that the service is enabled.

```
[student@servera ~]$ systemctl is-enabled tuned  
enabled
```

- 2.3. Verify that the service is currently running.

```
[student@servera ~]$ systemctl is-active tuned  
active
```

- 3. List the available tuning profiles and identify the active profile.

```
[student@servera ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- accelerator-performance      - Throughput performance based tuning with disabled
                                higher latency STOP states
- balanced                   - General non-specialized tuned profile
- 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
```

- 4. Review the `tuned.conf` configuration file for the current active profile, `virtual-guest`. You can find it in the `/usr/lib/tuned/virtual-guest` directory. The `virtual-guest` tuning profile is based on the `throughput-performance` profile, but it sets different values for the `vm.dirty_ratio` and `vm.swappiness` parameters. Verify that the `virtual-guest` tuning profile applies these values on your system.
- 4.1. Review the `virtual-guest` configuration file in the `/usr/lib/tuned/virtual-guest` directory. Check the values for the `vm.dirty_ratio` and `vm.swappiness` parameters.

```
[student@servera ~]$ cat /usr/lib/tuned/virtual-guest/tuned.conf
#
# tuned configuration
#
[main]
summary=Optimize for running inside a virtual guest
include=throughput-performance

[sysctl]
# 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%)
vm.dirty_ratio = 30
```

```
# 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
```

4.2. Verify that the tuning profile applies these values on your system.

```
[student@servera ~]$ sysctl vm.dirty_ratio
vm.dirty_ratio = 30
[student@servera ~]$ sysctl vm.swappiness
vm.swappiness = 30
```

- ▶ 5. Review the `tuned.conf` configuration file for the `virtual-guest` parent's tuning profile, `throughput-performance`. You can find it in the `/usr/lib/tuned/throughput-performance` directory. Notice that the `throughput-performance` tuning profile sets a different value for the `vm.dirty_ratio` and `vm.swappiness` parameters, although the `virtual-guest` profile overwrites them. Verify that the `virtual-guest` tuning profile applies the value for the `vm.dirty_background_ratio` parameter, which inherits from the `throughput-performance` profile.
  - 5.1. Review the `throughput-performance` configuration file in the `/usr/lib/tuned/throughput-performance` directory. Check the values for the `vm.dirty_ratio`, `vm.swappiness`, and `vm.dirty_background_ratio` parameters.

```
[student@servera ~]$ cat /usr/lib/tuned/throughput-performance/tuned.conf
#
# tuned configuration
#
[main]
summary=Broadly applicable tuning that provides excellent performance across a
variety of common server workloads
...output omitted...

[sysctl]
# 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%)
vm.dirty_ratio = 40

# Start background writeback (via writeback threads) at this percentage (system
# default is 10%)
vm.dirty_background_ratio = 10

# PID allocation wrap value. When the kernel's next PID value
# reaches this value, it wraps back to a minimum PID value.
# PIDs of value pid_max or larger are not allocated.
#
```

**Chapter 6 |** Tune System Performance

```
# A suggested value for pid_max is 1024 * <# of cpu cores/threads in system>
# e.g., a box with 32 cpus, the default of 32768 is reasonable, for 64 cpus,
# 65536, for 4096 cpus, 4194304 (which is the upper limit possible).
#kernel.pid_max = 65536

# The swappiness parameter controls the tendency of the kernel to move
# processes out of physical memory and onto the swap disk.
# 0 tells the kernel to avoid swapping processes out of physical memory
# for as long as possible
# 100 tells the kernel to aggressively swap processes out of physical memory
# and move them to swap cache
vm.swappiness=10

...output omitted...
```

- 5.2. Verify that the `virtual-guest` tuning profile applies the inherited `vm.dirty_background_ratio` parameter.

```
[student@servera ~]$ sysctl vm.dirty_background_ratio
vm.dirty_background_ratio = 10
```

- 6. Change the current active tuning profile to `throughput-performance`, and then confirm the results. Verify that the `vm.dirty_ratio` and `vm.swappiness` parameters change to the values in the `throughput-performance` configuration file.

- 6.1. Change the current active tuning profile.

```
[student@servera ~]$ sudo tuned-adm profile throughput-performance
```

- 6.2. Confirm that `throughput-performance` is the active tuning profile.

```
[student@servera ~]$ sudo tuned-adm active
Current active profile: throughput-performance
```

- 6.3. Verify the values for the `vm.dirty_ratio` and `vm.swappiness` parameters.

```
[student@servera ~]$ sysctl vm.dirty_ratio
vm.dirty_ratio = 40
[student@servera ~]$ sysctl vm.swappiness
vm.swappiness = 10
```

- 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, 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 tuning-profiles
```

This concludes the section.

# Influence Process Scheduling

---

## Objectives

Prioritize or deprioritize specific processes, with the `nice` and `renice` commands.

## Linux Process Scheduling

Modern computer systems use multi-core, multi-thread CPUs that can execute many instruction threads simultaneously. The largest high-performing supercomputers can have hundreds or thousands of CPUs with hundreds of processing cores and thread structures per CPU, and can process millions of instruction threads in parallel. Although a single user running many applications can saturate the typical desktop system or personal workstation with CPU activity, a properly sized and configured workstation is designed to match the user's intended workload. However, the typical enterprise or Internet server handles many hundreds or thousands of users and application requests each second, which can easily result in CPU saturation. All systems under CPU load will experience scenarios that require handling more process threads than the system has the CPU processing units needed to immediately schedule the threads.

Linux and other operating systems use a technique called *time-slicing* or *multitasking* for process management. The operating system *process scheduler* rapidly switches between process threads on each available CPU core. This behavior gives the impression that a significant number of processes are running at the same time.

## Process Priorities

Each process has a varying measure of importance, known historically as a process *priority*. Linux implements *scheduling policies* that define the rules by which processes are organized and prioritized to obtain CPU processing time. Linux has different *scheduling policies* designed for handling interactive application requests, non-interactive batch application processing, and real-time application requirements. Real-time scheduling policies still use process priorities and queues, but current, non-real-time (*normal*) scheduling policies use the Completely Fair Scheduler (CFS), which instead organizes processes that are awaiting CPU time into a binary search tree. This process priority introduction describes the default scheduling policy called `SCHED_NORMAL` or `SCHED_OTHER`.

Processes running under the `SCHED_NORMAL` policy are assigned a *static* real-time priority of 0, to ensure that all system real-time processes have a higher priority than normal processes. However, this static priority value is not included when organizing normal process threads for CPU scheduling. Instead, the CFS scheduling algorithm arranges normal process threads into a time-weighted binary tree, with the first item having the least amount of previously spent CPU time to the last item that has the most cumulative CPU time.

## Nice Value

The order of the binary tree is additionally influenced by a user-modifiable, per-process *nice* value, which ranges from -20 (increased priority) to 19 (decreased priority), with a default of 0. Processes inherit their starting nice value from their parent process.

A higher nice value indicates a decrease in the process priority from the default, which can be remembered as *making the process nicer* to other processes. A lower nice value indicates an

increase in the process priority from the default, which can be remembered as *making the process less nice* to other processes.

Modifying the nice value on a process will either raise or lower the process thread's position in the binary tree. Increasing the nice value will lower the thread's position, and decreasing the value will raise the thread's position.



### Important

Generally, priorities only indirectly determine the amount of CPU time a process thread receives. On a non-saturated system with available CPU capacity, every process is scheduled for immediate CPU time, for as much as each process wants. Relative process importance, as managed in the binary tree, determines only which threads are selected and placed on CPUs first.

On a CPU-saturated system, where there are more waiting threads than CPU processing units, higher priority (lower nice) process threads are placed first, until all CPU units are busy, while the lowest priority (higher nice) threads initially must wait in the binary tree. However, the Completely Fair Scheduler is designed to balance process importance, nice values, and previous cumulative CPU time, and dynamically adjusts the binary tree such that all processes obtain fair CPU time.

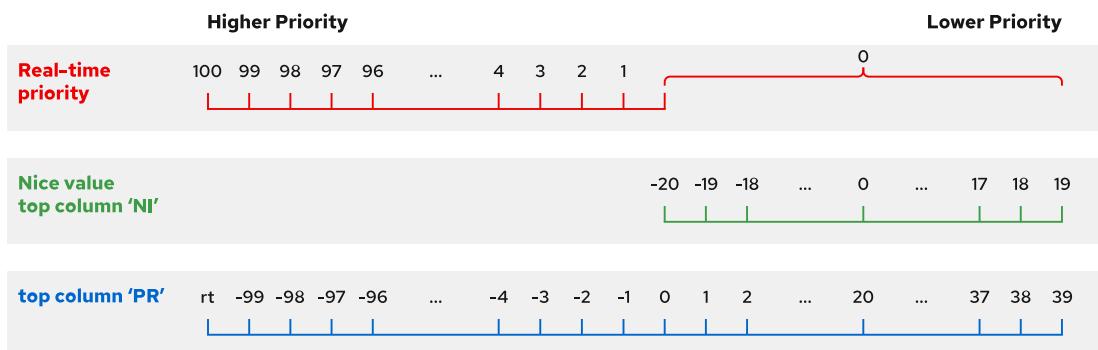
## Permission to Modify Nice Values

Privileged users can decrease the nice value of a process, to make a process less nice, which will cause a process to be repetitively placed higher in the binary tree, and therefore be scheduled more often. On a saturated system, the overall CPU time available to other processes is reduced.

Unprivileged users can only increase the nice value on their own processes, which will make their own processes nicer, and therefore lower their placement in the binary tree. Unprivileged users cannot decrease their process' nice values to raise their importance, nor can they adjust the nice values for another user's process.

## Viewing Nice Values

Nice values map to a priority value, and both values are available for viewing in process listing commands. A nice value of -20 maps to a 0 priority in the `top` command. A nice value of 19 maps to a 39 priority in the `top` command.



**Figure 6.3: Priorities and nice values as reported by the top command**

In Figure 6.3, the nice values are aligned with the priority values that are used by the `top` command. The `top` command displays the process priority in the PR column, and the nice value in the NI column. The `top` priority numbering scheme, which displays real-time process priorities as negative numbers, is a legacy of internal priority algorithms.

The following output is the summary and a partial process listing in the `top` command:

```
Tasks: 192 total, 1 running, 191 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 1.6 sy, 0.0 ni, 96.9 id, 0.0 wa, 0.0 hi, 1.6 si, 0.0 st
MiB Mem : 5668.6 total, 4655.6 free, 470.1 used, 542.9 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 4942.6 avail Mem

 PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
  1 root      20   0  172180  16232  10328 S  0.0  0.3  0:01.49 systemd
  2 root      20   0          0          0      0 S  0.0  0.0  0:00.01 kthreadd
  3 root      0 -20          0          0      0 I  0.0  0.0  0:00.00 rcu_gp
  4 root      0 -20          0          0      0 I  0.0  0.0  0:00.00 rcu_par_gp
```

The `ps` command displays process nice values, when using the default formatting options.

The following `ps` command lists all processes with their process ID, process name, nice value, and scheduling class, sorted in descending order by nice value. In the CLS scheduling class column, TS stands for *time sharing*, which is another name for the normal scheduling policies, including `SCHED_NORMAL`. Other CLS values, such as FF for *first in first out* and RR for *round robin*, indicate real-time processes. Real-time processes are not assigned nice values, as indicated by the dash (-) in the NI column. Advanced scheduling policies are taught in the *Red Hat Performance Tuning: Linux in Physical, Virtual, and Cloud (RH442)* course.

```
[user@host ~]$ ps axo pid,comm,nice,cls --sort=-nice
PID COMMAND      NI CLS
 33 khugepaged    19 TS
 32 ksmd         5 TS
 814 rtkit-daemon 1 TS
  1 systemd        0 TS
  2 kthreadd       0 TS
  5 kworker/0:0:cgr 0 TS
  7 kworker/0:1:rcu 0 TS
  8 kworker/u4:0:ev 0 TS
 15 migration/0    - FF
...output omitted...
```

## Start Processes with User-set Nice Values

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 process from the shell, and displays the process's nice value. Note the use of the `PID` option in the `ps` command to specify the requested output.



### Note

The example command is used here purely for demonstrating nice values, and was chosen for its low resource consumption.

```
[user@host ~]$ sleep 60 &
[1] 2667
[user@host ~]$ ps -o pid,comm,nice 2667
  PID COMMAND      NI
 2667 sleep        0
```

All users can use the `nice` command to start commands with a default or higher nice value. Without options, the `nice` command starts a process with a default nice value of 10. Setting a higher value by default ensures that the new process is a lower priority than your current working shell, and less likely to affect your current interactive session.

The following example starts the same command as a background job with the default nice value and displays the process's nice value:

```
[user@host ~]$ nice sleep 60 &
[1] 2736
[user@host ~]$ ps -o pid,comm,nice 2736
  PID COMMAND      NI
 2736 sleep        10
```

Use the `nice` command `-n` option to apply a user-defined nice value to the starting process. The default is to add 10 to the process's current nice value. The following example starts a background job with a user-defined nice value of 15 and displays the result:

```
[user@host ~]$ nice -n 15 sleep 60 &
[1] 2673
[user@host ~]$ ps -o pid,comm,nice 2740
  PID COMMAND      NI
 2740 sleep        15
```



### Important

Unprivileged users may only increase the nice value from its current value, to a maximum of 19. If the value is increased, then unprivileged users cannot reduce the value to revert to the previous nice value. However, a privileged user may reduce the nice value from any current value, to a minimum of -20.

## Change the Nice Value of an Existing Process

You can change the nice value of an existing process with the `renice` command. This example uses the process ID from the previous example to change from the current nice value of 15 to a new nice value of 19.

```
[user@host ~]$ renice -n 19 2740
2740 (process ID) old priority 15, new priority 19
```

You can also use the `top` command to change the nice value on an existing process. From the `top` interactive interface, press the `r` option to access the `renice` command. Enter the process ID, and then the new nice value.



## References

`nice(1)`, `renice(1)`, `top(1)`, and `sched_setscheduler(2)` man pages

## ► Guided Exercise

# Influence Process Scheduling

In this exercise, you adjust the scheduling priority of processes with the `nice` and `renice` commands and observe the effects on process execution.

### Outcomes

- Adjust scheduling priorities for processes.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start tuning-procscheduling
```



#### Important

This exercise uses commands that perform an endless checksum on a device file while intentionally using significant CPU resources. Verify that you have terminated all exercise processes before leaving this exercise.

## Instructions

- 1. Use the `ssh` command to log in to the `servera` machine as the `student` user.

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

- 2. Determine the number of CPU cores on the `servera` machine and then start two instances of the `sha1sum /dev/zero &` command for each core.

- 2.1. Use the `grep` command to parse the number of existing virtual processors (CPU cores) from the `/proc/cpuinfo` file.

```
[student@servera ~]$ grep -c '^processor' /proc/cpuinfo  
2
```

- 2.2. Use a looping command to start multiple instances of the `sha1sum /dev/zero &` command. Start two instances for each virtual processor that was indicated in the previous step. In this example, a `for` loop creates four instances. The PID values in your output might vary from the example.

```
[student@servera ~]$ for i in {1..4}; do sha1sum /dev/zero & done
[1] 1132
[2] 1133
[3] 1134
[4] 1135
```

- 3. Verify that the background jobs are running for each of the sha1sum processes.

```
[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 and pgrep commands to display the percentage of CPU usage for each sha1sum process.

```
[student@servera ~]$ ps u $(pgrep sha1sum)
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
student   1132 49.6  0.1 225336  2288 pts/0    R    11:40  2:40 sha1sum /dev/zero
student   1133 49.6  0.1 225336  2296 pts/0    R    11:40  2:40 sha1sum /dev/zero
student   1134 49.6  0.1 225336  2264 pts/0    R    11:40  2:40 sha1sum /dev/zero
student   1135 49.6  0.1 225336  2280 pts/0    R    11:40  2:40 sha1sum /dev/zero
```

- 5. Terminate all sha1sum processes, and then verify that no jobs are running.

- 5.1. Use the pkill command to terminate all running processes with the name pattern sha1sum.

```
[student@servera ~]$ pkill sha1sum
[2]  Terminated      sha1sum /dev/zero
[4]+  Terminated      sha1sum /dev/zero
[1]-  Terminated      sha1sum /dev/zero
[3]+  Terminated      sha1sum /dev/zero
```

- 5.2. Verify that no jobs are running.

```
[student@servera ~]$ jobs
[student@servera ~]$
```

- 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 level of 10. Start at least as many instances as the number of system virtual processors. In this example, three regular instances are started, plus another with a higher nice level.

- 6.1. Use looping to start three instances of the sha1sum /dev/zero & command.

```
[student@servera ~]$ for i in {1..3}; do sha1sum /dev/zero & done
[1] 1207
[2] 1208
[3] 1209
```

- 6.2. Use the nice command to start the fourth instance with a nice level of 10.

```
[student@servera ~]$ nice -n 10 sha1sum /dev/zero &
[4] 1210
```

- 7. Use the ps and pgrep commands to display the PID, percentage of CPU usage, nice value, and executable name for each process. The instance with the nice value of 10 should display a lower percentage of CPU usage than the other instances.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
 PID %CPU NI COMMAND
 1207 64.2 0 sha1sum
 1208 65.0 0 sha1sum
 1209 63.9 0 sha1sum
 1210 8.2 10 sha1sum
```

- 8. Use the sudo renice command to lower the nice level of a process from the previous step. Use the PID value of the process instance with the nice level of 10 to lower its nice level to 5.

```
[student@servera ~]$ sudo renice -n 5 1210
[sudo] password for student:
1210 (process ID) old priority 10, new priority 5
```

- 9. Repeat the ps and pgrep commands to display the CPU percentage and nice level.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
 PID %CPU NI COMMAND
 1207 62.9 0 sha1sum
 1208 63.2 0 sha1sum
 1209 63.2 0 sha1sum
 1210 10.9 5 sha1sum
```

- 10. Use the pkill command to terminate all running processes with the sha1sum name pattern.

```
[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, 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 tuning-procscheduling
```

This concludes the section.

## ► Lab

# Tune System Performance

In this lab, you apply a specific tuning profile and adjust the scheduling priority of an existing process with high CPU usage.

## Outcomes

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures 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 while intentionally using significant CPU resources. Verify that you have terminated all lab processes before leaving this lab.

## Instructions

1. Change the current tuning profile for the `serverb` machine to the `balanced` profile, a general non-specialized tuned profile. List the information for the `balanced` tuning profile when it is the current tuning profile.
2. Two processes on `serverb` are consuming a high percentage of CPU usage. Adjust each process's `nice` level to 10 to allow more CPU time for other processes.

## 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, 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 tuning-review
```

This concludes the section.

## ► Solution

# Tune System Performance

In this lab, you apply a specific tuning profile and adjust the scheduling priority of an existing process with high CPU usage.

### Outcomes

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures 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 while intentionally using significant CPU resources. Verify that you have terminated all lab processes before leaving this lab.

### Instructions

1. Change the current tuning profile for the `serverb` machine to the `balanced` profile, a general non-specialized tuned profile. List the information for the `balanced` tuning profile when it is the current 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           2.18.0-1.el9          @System
```

- 1.3. Verify the `tuned` service state.

```
[student@serverb ~]$ systemctl is-active tuned
active
```

- 1.4. List all available tuning profiles and their descriptions. Note that the current active profile is **virtual-guest**.

```
[student@serverb ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- accelerator-performance      - Throughput performance based tuning with disabled
                                higher latency STOP states
- balanced                     - General non-specialized tuned profile
- 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.

```
[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

Profile summary:
General non-specialized tuned profile
...output omitted...
```

2. Two processes on **serverb** are consuming a high percentage of CPU usage. Adjust each process's nice level to 10 to allow more CPU time for other processes.
- 2.1. Determine the top two CPU consumers on the **serverb** machine. The **ps** command lists the top CPU consumers at the bottom of the output. CPU percentage values 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 level for each of the top two CPU consumers.

```
[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 level for each process to 10. Use the correct PID values for your processes from the previous command output.

```
[student@serverb ~]$ sudo renice -n 10 1079 1095
[sudo] password for student:
1079 (process ID) old priority 2, new priority 10
1095 (process ID) old priority -2, new priority 10
```

- 2.4. Verify that the current nice level for each process is 10.

```
[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
```

- 2.5. 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, 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 tuning-review
```

This concludes the section.

# Summary

---

- A signal is a software interrupt that reports events to an executing program. The `kill`, `pkill`, and `killall` commands use signals to control processes.
- Load average is an estimate of how busy the system is. To display load average values, you can use the `top`, `uptime`, or `w` command.
- The `tuned` service automatically modifies device settings to meet specific system needs based on a predefined selected tuning profile.
- To revert all changes of the selected profile to the system settings, either switch to another profile or deactivate the `tuned` service.
- The system assigns a relative priority to a process to determine its CPU access. This priority is called the `nice` value of a process.
- The `nice` command assigns a priority to a process when it starts.
- The `renice` command modifies the priority of a running process.

## Chapter 7

# Schedule Future Tasks

### Goal

Schedule tasks to automatically execute in the future.

### Objectives

- Schedule commands to run on a repeating schedule with a user's crontab file.
- Schedule commands to run on a repeating schedule with the system crontab file and directories.
- Enable and disable systemd timers, and configure a timer that manages temporary files.

### Sections

- Schedule Recurring User Jobs (and Guided Exercise)
- Schedule Recurring System Jobs (and Guided Exercise)
- Manage Temporary Files (and Guided Exercise)

# Schedule Recurring User Jobs

---

## Objectives

Schedule commands to run on a repeating schedule with a user's crontab file.

## Describe Recurring User Jobs

*Recurring* jobs are scheduled to run repeatedly. Red Hat Enterprise Linux systems provide the `cron`d daemon, which is enabled and started by default. The `cron`d daemon reads multiple configuration files: one per user and a set of system-wide files. Each user has a personal file which they edit with the `crontab -e` command. 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 `cron`d 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 local user can use to manage their jobs:

### 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 those read from <i>filename</i> . 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` commands as the `root` user is not recommended due to the ability to exploit personal jobs configured to run as `root`. Such privileged jobs should be configured as described in the later section describing recurring system jobs.

## Describe User Job Format

The `crontab -e` command 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 is used for interpreting 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 the month* or *Day of the 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. 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 using 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.

## Examples of Recurring User Jobs

The following job executes the `/usr/local/bin/yearly_backup` command at exactly 9:00 AM on 3 February, every year. February is represented as the number 2 in the example, as it is the second month of the year.

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

The following job sends an email containing the Chime word to the owner of this job every five minutes in between and including 9 a.m. and 16 p.m., but only on each Friday in July.

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

The preceding 9-16 range of hours means that the job timer starts at the ninth hour (09:00) and continues until the end of the sixteenth hour (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 specific for the hours instead of a single value, then all hours within the range will 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 crond recognizes that the job allowed output to go to the STDIO channel without redirection. Because cron jobs run in a background environment without an output device (known as a *controlling terminal*), crond buffers the output and creates an email to send it to the user specified in the configuration. For system jobs, the email will be sent to the root account.

The following job runs the /usr/local/bin/daily\_report command every working day (Monday to Friday) 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 working day (Monday to Friday), at 9 AM.

```
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

## ► Guided Exercise

# Schedule Recurring User Jobs

In this exercise, you schedule commands to run on a repeating schedule as a non-privileged user, with the `cron` command.

## Outcomes

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

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures 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. The job must run only from Monday to Friday, not on Saturday or Sunday.



### Important

If you are working on this exercise outside the specified days in the preceding instruction, then adjust your system time and date accordingly so that the job runs while you are working.

- 2.1. Open the `crontab` file with the default text editor.

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

- 2.2. Insert the following line:

```
*/2 * * * Mon-Fri /usr/bin/date >> /home/student/my_first_cron_job.txt
```

- 2.3. Press Esc and type :wq to save the changes and exit the editor. When the editor exits, you should see the following output:

```
...output omitted...
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 * * * Mon-Fri /usr/bin/date >> /home/student/my_first_cron_job.txt
```

- 4. Have your shell prompt sleep until the `/home/student/my_first_cron_job.txt` file is created as a result 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
Mon Apr  4 03:04:01 AM EDT 2022
```

- 6. Remove all the scheduled recurring jobs for the `student` user.

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

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

- 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, 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 scheduling-cron
```

This concludes the section.

# Schedule Recurring System Jobs

---

## Objectives

Schedule commands to run on a repeating schedule with the system crontab file and 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 helpful 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,wed,thu,fri,sat
# | | | | |
# * * * * * 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, and month. These repositories are placed under the `/etc/cron.hourly/`, `/etc/cron.daily/`, `/etc/cron.weekly/`, and `/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. If a script is not executable, then it does not run.

## Run Periodic Commands with Anacron

The `run-parts` command also runs the daily, weekly, and monthly jobs from the `/etc/anacrontab` configuration file.

The `/etc/anacrontab` file 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, and monthly jobs. When the `crond` 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 follows.

### 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 macro `@daily` is equivalent to the `1` integer, which executes the job daily. Similarly, the macro `@weekly` 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

This field 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. When a job does not run within this time interval on a particular day, then the job must wait until the next day for execution.

## Systemd Timer

The `systemd` timer unit activates another unit of a different type (such as a service) whose unit name matches the timer unit name. The timer unit allows timer-based activation of other units. The `systemd` timer unit logs timer events in system journals for easier debugging.

## Sample Timer Unit

The `sysstat` package provides the `systemd` timer unit, called 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 `OnCalendar=*:00/10` option signifies that this timer unit activates the corresponding `sysstat-collect.service` unit every 10 minutes. You might specify more complex time intervals.

For example, a `2022-04-* 12:35,37,39:16` value against the `OnCalendar` option causes the timer unit to activate the corresponding service unit at the `12:35:16`, `12:37:16`, and `12:39:16` times, every day during April 2022. You might also specify relative timers with the `OnUnitActiveSec` option. For example, with the `OnUnitActiveSec=15min` option, the timer unit triggers the corresponding unit to start 15 minutes after the last time that the timer unit activated its corresponding unit.



### Important

Do not modify any units in the configuration files under the `/usr/lib/systemd/system` directory, because the `systemd` unit overrides the configuration changes in that file. Create a copy of the configuration file in 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 activate the timer unit.

```
[root@host ~]# systemctl enable --now <unitname>.timer
```



### References

`crontab(5)`, `anacron(8)`, `anacrontab(5)`, `systemd.time(7)`, `systemd.timer(5)`, and `crond(8)` man pages

## ► Guided Exercise

# Schedule Recurring System Jobs

In this exercise, you schedule commands to run on various schedules by adding configuration files to the system crontab directories.

## Outcomes

- Schedule a recurring system job to count the number of active users.
- Update the `systemd` timer unit that gathers system activity data.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start scheduling-system
```

## 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. Schedule a recurring system job that generates a log message that indicates the number of active users in the system. This job must run daily and use the `w -h | wc -l` command to retrieve the number of active users in the system. Use the `logger` command to generate the log message of currently active users.

- 2.1. Create the `/etc/cron.daily/usercount` script file with the following content:

```
#!/bin/bash
USERCOUNT=$(w -h | wc -l)
logger "There are currently ${USERCOUNT} active users"
```

- 2.2. Make the script file executable.

```
[root@servera ~]# chmod +x /etc/cron.daily/usercount
```

- 3. Install the sysstat package. The timer unit must trigger the service unit every ten minutes to collect system activity data with the /usr/lib64/sa/sa1 shell script. Change the timer unit configuration file to collect the system activity data every two minutes.

3.1. Install the sysstat package.

```
[root@servera ~]# dnf install sysstat
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

3.2. 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
```

3.3. Edit the /etc/systemd/system/sysstat-collect.timer file for the timer unit to run every two minutes. Replace any occurrence of the 10 minutes string with 2 minutes throughout the unit configuration file, including the occurrences in the commented lines. Use the vim /etc/systemd/system/sysstat-collect.timer command to edit the configuration file.

From these changes, the sysstat-collect.timer unit triggers the sysstat-collect.service unit every two minutes and collects the system activity data in a binary file in the /var/log/sa directory.

```
...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
```

3.4. Make the systemd daemon aware of the changes.

```
[root@servera ~]# systemctl daemon-reload
```

3.5. Activate the sysstat-collect.timer unit.

```
[root@servera ~]# systemctl enable --now sysstat-collect.timer
...output omitted...
```

3.6. 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 is not present. The while loop exits when the file is present.

```
[root@servera ~]# while [ $(ls /var/log/sa | wc -l) -eq 0 ]; \
do sleep 1s; done
```

- 3.7. Verify that the binary file in the `/var/log/sa` directory was 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
[root@servera ~]# date
Tue Apr  5 04:08:29 AM EDT 2022
```

- 3.8. 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, 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 scheduling-system
```

This concludes the section.

# Manage Temporary Files

## Objectives

Enable and disable `systemd` timers, and configure a timer that manages temporary files.

## Manage Temporary Files

Most critical applications and services use temporary files and directories. Some applications and users use the `/tmp` directory to hold transient working data, while other applications use task-specific locations such as daemon- and user-specific volatile directories under `/run`, which exist only in memory. When the system reboots or loses power, memory-based file systems are self-cleaning.

Commonly, daemons and scripts operate properly only when their expected temporary files and directories exist. Additionally, purging temporary files located 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 launched is the `systemd-tmpfiles-setup` service. This service runs the `systemd-tmpfiles` command `--create --remove` options, 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 called `systemd-tmpfiles-clean.timer` at a regular interval triggers `systemd-tmpfiles-clean.service`, which executes the `systemd-tmpfiles --clean` command.

A `systemd` timer unit configuration has a `[Timer]` section for indicating 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.
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation; either version 2.1 of the License, or
# (at your option) any later version.
```

```
[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` timer unit configuration file to meet your requirements. For example, a `30min` value for the `OnUnitActiveSec` parameter triggers the `systemd-tmpfiles-clean.service` service unit 30 minutes after the service unit is last activated. As a result, `systemd-tmpfiles-clean.service` gets triggered every 30 minutes after the changes are recognized.

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

```
[root@host ~]# systemctl daemon-reload
```

After reloading the `systemd` manager configuration, use the following `systemctl` command to activate the `systemd-tmpfiles-clean.timer` unit.

```
[root@host ~]# systemctl enable --now systemd-tmpfiles-clean.timer
```

## Clean Temporary Files Manually

The `systemd-tmpfiles --clean` command parses the same configuration files as the `systemd-tmpfiles --create` command, but instead of creating files and directories, it purges all files that were not accessed, changed, or modified more recently than the maximum age as defined in the configuration file.

Find detailed information about the format of the configuration files for the `systemd-tmpfiles` service in the `tmpfiles.d(5)` man page. The basic syntax consists of the following columns: Type, Path, Mode, UID, GID, Age, and Argument. Type refers to the action that the `systemd-tmpfiles` service should take; for example, `d` to create a directory if it does not exist, or `Z` to recursively restore SELinux contexts, file permissions, and ownership.

The following are examples of purge configuration with explanations:

```
d /run/systemd/seats 0755 root root -
```

When you create files and directories, create the `/run/systemd/seats` directory if it does not exist, with the `root` user and the `root` group as owners, and with permissions of `rwxr-xr-x`. If this directory does exist, then take no action. The `systemd-tmpfiles` service does not purge this directory automatically.

```
D /home/student 0700 student student 1d
```

Create the `/home/student` directory if it does not exist. If it does exist, then empty it of all contents. When the system runs the `systemd-tmpfiles --clean` command, it removes all files in the directory that you did not access, change, or modify for more than one day.

```
L /run/fstablink - root root - /etc/fstab
```

Create the `/run/fstablink` symbolic link, to point to the `/etc/fstab` folder. Never automatically purge this line.

## 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 normally daemons 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 easily override vendor-provided settings by copying the relevant file to the `/etc/tmpfiles.d/` directory and then editing it. By using these configuration locations properly, you can manage administrator-configured settings from a central configuration management system, and package updates will not overwrite your configured settings.



### Note

When testing new or modified configurations, it is useful to 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

## ► Guided Exercise

# Manage Temporary Files

In this exercise, you configure `systemd-tmpfiles` to change how quickly it removes temporary files from `/tmp`, and also to periodically purge files from another directory.

## Outcomes

- Configure `systemd-tmpfiles` to remove unused temporary files from `/tmp`.
- Configure `systemd-tmpfiles` to periodically purge files from another directory.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start scheduling-tempfiles
```

## Instructions

- 1. Log in to the `servera` system 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. 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.

- 2.1. 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
```

- 2.2. Search for the configuration line in the `/etc/tmpfiles.d/tmp.conf` file that applies to the `/tmp` directory. Replace the existing age of the temporary files in that configuration line with the new age of 5 days. Remove from the file all the other lines, including the commented lines. You can use the `vim /etc/tmpfiles.d/tmp.conf` command to edit the configuration file.

In the configuration, the `q` type is identical to the `d` type and instructs the `systemd-tmpfiles` service to create 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 root. The /tmp directory must not contain the unused temporary files from the last five days.

The /etc/tmpfiles.d/tmp.conf file should appear as follows:

```
q /tmp 1777 root root 5d
```

- 2.3. Verify the /etc/tmpfiles.d/tmp.conf file configuration.

Because the command does not return any errors, it confirms that the configuration settings are correct.

```
[root@servera ~]# systemctl-tmpfiles --clean /etc/tmpfiles.d/tmp.conf
```

- 3. Add a new configuration that ensures that the /run/momentary directory exists with user and group ownership set to root. The octal permissions for the directory must be 0700. The configuration must purge any file in this directory that remains unused in the last 30 seconds.

- 3.1. Create the /etc/tmpfiles.d/momentary.conf file with the following content.

With the configuration, the `systemd-tmpfiles` service ensures that the /run/momentary directory exists and that its octal permissions are set to 0700. The ownership of the /run/momentary directory must be the root user and group. The service purges any file in this directory if it remains unused for 30 seconds.

```
[root@servera ~]# vim /etc/tmpfiles.d/momentary.conf
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.

Because the command does not return any errors, it confirms that the configuration settings are correct.

```
[root@servera ~]# systemctl-tmpfiles --create \
/etc/tmpfiles.d/momentary.conf
```

- 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 under the /run/momentary directory that is unused in the last 30 seconds, based on the `systemd-tmpfiles` configuration for the directory.

- 4.1. Create the /run/momentary/test file.

```
[root@servera ~]# touch /run/momentary/test
```

- 4.2. Configure your shell prompt to not to return for 30 seconds.

```
[root@servera ~]# sleep 30
```

- 4.3. After your shell prompt returns, clean stale files from 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
```

- 4.5. 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, 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 scheduling-tempfiles
```

This concludes the section.

## ► Quiz

# Schedule Future 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 /usr/local/bin/daily\_backup 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
  
- ▶ 5. Which directory contains the shell scripts to run daily?
  - a. /etc/cron.d
  - b. /etc/cron.hourly
  - c. /etc/cron.daily
  - d. /etc/cron.weekly

- ▶ **6. 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
  
- ▶ **7. 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

## ► Solution

# Schedule Future 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 /usr/local/bin/daily\_backup 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
  
- ▶ 5. Which directory contains the shell scripts to run daily?
  - a. /etc/cron.d
  - b. /etc/cron.hourly
  - c. /etc/cron.daily
  - d. /etc/cron.weekly

- ▶ **6. 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
  
- ▶ **7. 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

# Summary

---

- Deferred jobs or tasks are scheduled to run once in the future.
- Recurring user jobs execute the user's tasks on a repeating schedule.
- Recurring system jobs accomplish, on a repeating schedule, administrative tasks with system-wide impact.
- The systemd timer units can execute both the deferred and recurring jobs.

## Chapter 8

# Install and Update Software Packages

### Goal

Download, install, update, and manage software packages from Red Hat and DNF package repositories.

### Objectives

- Register a system to your Red Hat account and assign it entitlements for software updates and support services with Red Hat Subscription Management.
- Find, install, and update software packages with the `dnf` command.
- Enable and disable server use of Red Hat or third-party DNF repositories.

### Sections

- Register Systems for Red Hat Support (and Quiz)
- Install and Update Software Packages with DNF (and Guided Exercise)
- Enable DNF Software Repositories (and Guided Exercise)

### Lab

- Install and Update Software Packages

# Register Systems for Red Hat Support

## Objectives

Register a system to your Red Hat account and assign it entitlements for software updates and support services with Red Hat Subscription Management.

## Red Hat Subscription Management

Red Hat Subscription Management provides tools to entitle machines to product subscriptions, for administrators to get updates to software packages and to track information about support contracts and subscriptions that the systems use. Standard tools such as the `dnf` command obtain software packages and updates through a content distribution network that the Red Hat Content Delivery Network provides.

You can perform the following main tasks with the Red Hat Subscription Management tools:

- *Register* a system to associate it with the Red Hat account with an active subscription. With the Subscription Manager, the system can register uniquely in the subscription service inventory. You can unregister the system when not in use.
- *Subscribe* a system to entitle it to updates for the selected Red Hat products. Subscriptions have specific levels of support, expiration dates, and default repositories. The tools help to either auto-attach or select a specific entitlement.
- *Enable repositories* to provide software packages. By default, each subscription enables multiple repositories; other repositories such as updates or source code are enabled or disabled.
- *Review and track* available or consumed entitlements. In the Red Hat Customer Portal, you might view the subscription information locally on a specific system or for a Red Hat account.

## Simple Content Access

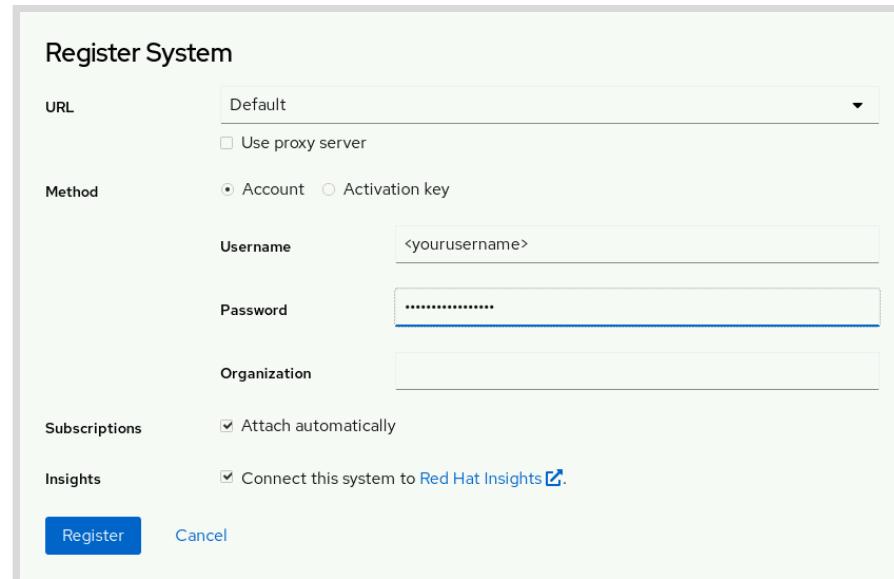
Simple Content Access (SCA) is a Red Hat subscription management capability. When you enable SCA for your organization, the entitlement process is simplified. SCA eliminates the requirement to attach subscriptions at a per-system level. You register your systems, enable the repositories that each system needs, and begin installing software packages.

Simple Content Access is an optional feature of Red Hat Satellite Server and Red Hat Subscription Management. This course includes the subscription commands, as needed, if you have not yet enabled SCA.

## Subscribe a System with RHEL Web Console

Different options exist to register a system with the Red Hat Customer Portal. For example, you can access a graphical interface by using a GNOME application or through the RHEL web console, or you can register your system by using a command-line tool.

To register a system with the RHEL web console, launch the Red Hat Subscription Manager application from the **Activities** menu. Type *subscription* in the **Type to search** field and click the **Red Hat Subscription Manager** application. When prompted, enter the appropriate password to authenticate. In the **Subscriptions** window, click **Register** to open the **Register System** dialog box.



**Figure 8.1: The Register System dialog box**

By default, systems register to the Red Hat Customer Portal. Provide the login and the password for your Red Hat Customer Portal account and click **Register** to register the system. When registered, the system automatically attaches an available subscription.

Close the **Subscriptions** window after registering and assigning the system to a subscription. The system is now subscribed and ready to receive updates or to install new software according to the subscription that is attached to the Red Hat Content Delivery Network.

## Subscribe a System with the Command Line

Use the `subscription-manager` command to register a system without using a graphical environment. The `subscription-manager` command automatically attaches a system to the best-matched compatible subscriptions for the system.

Register a system by using the credentials of the Red Hat Customer Portal as the `root` user:

```
[root@host ~]# subscription-manager register --username <yourusername>
Registering to: subscription.rhsm.redhat.com:443/subscription
Password: yourpassword
The system has been registered with ID: 1457f7e9-f37e-4e93-960a-c94fe08e1b4f
The registered system name is: host.example.com
```

View available subscriptions for your Red Hat account:

```
[root@host ~]# subscription-manager list --available
-----
 Available Subscriptions
-----
 ...output omitted...
```

Auto-attach a subscription:

```
[root@host ~]# subscription-manager attach --auto  
...output omitted...
```

Alternatively, attach a subscription from a specific pool from the list of available subscriptions:

```
[root@host ~]# subscription-manager attach --pool=poolID  
...output omitted...
```

View consumed subscriptions:

```
[root@host ~]# subscription-manager list --consumed  
...output omitted...
```

Unregister a system:

```
[root@host ~]# subscription-manager unregister  
Unregistering from: subscription.rhsm.redhat.com:443/subscription  
System has been unregistered.
```

## Activation Keys

An *activation key* is a preconfigured subscription management file that available for use with both Red Hat Satellite Server and subscription management through the Red Hat Customer Portal.

Use the `subscription-manager` command with activation keys to simplify the registration and assignment of predefined subscriptions. This method of registration is beneficial for automating installations and deployments. For organizations that enable Simple Content Access, activation keys can register systems and enable repositories without needing to attach subscriptions.

## Entitlement Certificates

Digital certificates store current entitlement information on the local system. The registered system stores the entitlement certificates under the `/etc/pki` directory.

- `/etc/pki/product` certificates indicates installed Red Hat products.
- `/etc/pki/consumer` certificates identifies the Red Hat account for registration.
- `/etc/pki/entitlement` certificates indicate which subscriptions are attached.

The `rct` command inspects the certificates and the `subscription-manager` command examines the attached subscriptions on the system.



### References

`subscription-manager(8)` and `rct(8)` man pages

For further information, refer to *Registering the System and Managing Subscriptions* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/assembly\\_registering-the-system-and-managing-subscriptions\\_configuring-basic-system-settings](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/assembly_registering-the-system-and-managing-subscriptions_configuring-basic-system-settings)

## ► Quiz

# Register Systems for Red Hat Support

Choose the correct answer to the following questions:

- ▶ 1. **Which item helps to register the system to Red Hat Subscription Management without a username and password?**
  - a. Organization ID
  - b. Proxy URL
  - c. Activation keys
  - d. dnf
  
- ▶ 2. **Which GUI tool is used to register and subscribe a system?**
  - a. PackageKit
  - b. gpk-application
  - c. Red Hat Subscription Manager
  - d. gnome-software
  
- ▶ 3. **Which directory stores the certificates for Red Hat products when using entitlement certificates?**
  - a. /etc/pki/entitlement
  - b. /etc/subscription/product
  - c. /etc/pki/product
  - d. /etc/certs/pki
  - e. None of the previous options.

## ► Solution

# Register Systems for Red Hat Support

Choose the correct answer to the following questions:

- ▶ 1. **Which item helps to register the system to Red Hat Subscription Management without a username and password?**
  - a. Organization ID
  - b. Proxy URL
  - c. Activation keys
  - d. dnf
  
- ▶ 2. **Which GUI tool is used to register and subscribe a system?**
  - a. PackageKit
  - b. gpk-application
  - c. Red Hat Subscription Manager
  - d. gnome-software
  
- ▶ 3. **Which directory stores the certificates for Red Hat products when using entitlement certificates?**
  - a. /etc/pki/entitlement
  - b. /etc/subscription/product
  - c. /etc/pki/product
  - d. /etc/certs/pki
  - e. None of the previous options.

# Install and Update Software Packages with DNF

## Objectives

Find, install, and update software packages with the `dnf` command.

## Manage Software Packages with DNF

DNF (Dandified YUM) replaced YUM as the package manager in Red Hat Enterprise Linux 9. DNF commands are functionally the same as YUM commands. For compatibility, YUM commands still exist as symbolic links to DNF:

```
[user@host ~]$ ls -l /bin/ | grep yum | awk '{print $9 " " $10 " " $11}'  
yum -> dnf-3  
yum-builddep -> /usr/libexec/dnf-utils  
yum-config-manager -> /usr/libexec/dnf-utils  
yum-debug-dump -> /usr/libexec/dnf-utils  
yum-debug-restore -> /usr/libexec/dnf-utils  
yumdownloader -> /usr/libexec/dnf-utils  
yum-groups-manager -> /usr/libexec/dnf-utils
```

In this course, you work with the `dnf` command. Some documentation might still refer to the `yum` command, but the files are the same linked command.

The low-level `rpm` command can be used to install packages, but it is not designed to work with package repositories or to resolve dependencies from multiple sources automatically.

DNF improves RPM-based software installation and updates. With the `dnf` command, you can install, update, remove, and get information about software packages and their dependencies. You can get a history of transactions and work with multiple Red Hat and third-party software repositories.

## Find Software with DNF

The `dnf help` command displays usage information. The `dnf list` command displays installed and available packages.

```
[user@host ~]$ dnf list 'http*'  
Available Packages  
http-parser.i686          2.9.4-6.el9      rhel-9.0-for-x86_64-appstream-rpms  
http-parser.x86_64          2.9.4-6.el9      rhel-9.0-for-x86_64-appstream-rpms  
httpcomponents-client.noarch 4.5.13-2.el9    rhel-9.0-for-x86_64-appstream-rpms  
httpcomponents-core.noarch   4.4.13-6.el9    rhel-9.0-for-x86_64-appstream-rpms  
httpd.x86_64                 2.4.51-5.el9    rhel-9.0-for-x86_64-appstream-rpms  
httpd-devel.x86_64           2.4.51-5.el9    rhel-9.0-for-x86_64-appstream-rpms  
httpd-filesystem.noarch       2.4.51-5.el9    rhel-9.0-for-x86_64-appstream-rpms  
httpd-manual.noarch          2.4.51-5.el9    rhel-9.0-for-x86_64-appstream-rpms  
httpd-tools.x86_64            2.4.51-5.el9    rhel-9.0-for-x86_64-appstream-rpms
```

The `dnf search KEYWORD` command lists packages by keywords found in the name and summary fields only. To search for packages with "web server" in their name, summary, and description fields, use `search all`:

```
[user@host ~]$ dnf search all 'web server'
=====
Summary & Description Matched: web server =====
nginx.x86_64 : A high performance web server and reverse proxy server
pcp-pmda-weblog.x86_64 : Performance Co-Pilot (PCP) metrics from web server logs
=====
Summary Matched: web server =====
libcurl.x86_64 : A library for getting files from web servers
libcurl.i686 : A library for getting files from web servers
=====
Description Matched: web server =====
freeradius.x86_64 : High-performance and highly configurable free RADIUS server
git-instaweb.noarch : Repository browser in gitweb
http-parser.i686 : HTTP request/response parser for C
http-parser.x86_64 : HTTP request/response parser for C
httpd.x86_64 : Apache HTTP Server
mod_auth_openidc.x86_64 : OpenID Connect auth module for Apache HTTP Server
mod_jk.x86_64 : Tomcat mod_jk connector for Apache
mod_security.x86_64 : Security module for the Apache HTTP Server
varnish.i686 : High-performance HTTP accelerator
varnish.x86_64 : High-performance HTTP accelerator
...output omitted...
```

The `dnf info PACKAGE_NAME` command returns detailed information about a package, including the needed disk space for installation. For example, the following command retrieves information about the `httpd` package:

```
[user@host ~]$ dnf info httpd
Available Packages
Name        : httpd
Version     : 2.4.51
Release     : 5.el9
Architecture: x86_64
Size        : 1.5 M
Source      : httpd-2.4.51-5.el9.src.rpm
Repository   : rhel-9.0-for-x86_64-appstream-rpms
Summary     : Apache HTTP Server
URL         : https://httpd.apache.org/
License      : ASL 2.0
Description  : The Apache HTTP Server is a powerful, efficient, and extensible
               : web server.
```

The `dnf provides PATHNAME` command displays packages that match the specified path name (the path names often include wildcard characters). For example, the following command finds packages that provide the `/var/www/html` directory:

```
[user@host ~]$ dnf provides /var/www/html
httpd-filesystem-2.4.51-5.el9.noarch : The basic directory layout for the Apache
                                         HTTP Server
Repo        : rhel-9.0-for-x86_64-appstream-rpms
Matched from:
Filename    : /var/www/html
```

## Install and Remove Software with DNF

The `dnf install PACKAGE_NAME` command obtains and installs a software package, including any dependencies.

```
[root@host ~]# dnf install httpd
Dependencies resolved.
=====
 Package      Arch    Version       Repository      Size
=====
Installing:
 httpd        x86_64  2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 1.5 M
Installing dependencies:
 apr          x86_64  1.7.0-11.el9   rhel-9.0-for-x86_64-appstream-rpms 127 k
 apr-util     x86_64  1.6.1-20.el9   rhel-9.0-for-x86_64-appstream-rpms 98 k
 apr-util-bdb x86_64  1.6.1-20.el9   rhel-9.0-for-x86_64-appstream-rpms 15 k
 httpd-filesystem noarch 2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 17 k
 httpd-tools   x86_64  2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 88 k
 redhat-logos-httpd
                  noarch 90.4-1.el9   rhel-9.0-for-x86_64-appstream-rpms 18 k
Installing weak dependencies:
 apr-util-openssl x86_64 1.6.1-20.el9   rhel-9.0-for-x86_64-appstream-rpms 17 k
 mod_http2       x86_64  1.15.19-2.el9   rhel-9.0-for-x86_64-appstream-rpms 153 k
 mod_lua         x86_64  2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 63 k

Transaction Summary
=====
Install 10 Packages

Total download size: 2.1 M
Installed size: 5.9 M
Is this ok [y/N]: y
Downloading Packages:
(1/10): apr-1.7.0-11.el9.x86_64.rpm           6.4 MB/s | 127 kB    00:00
(2/10): apr-util-bdb-1.6.1-20.el9.x86_64.rpm   625 kB/s | 15 kB    00:00
(3/10): apr-util-openssl-1.6.1-20.el9.x86_64.r p 1.9 MB/s | 17 kB    00:00
...output omitted...
Total                                         24 MB/s | 2.1 MB    00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing      :                                     1/1
  Installing    : apr-1.7.0-11.el9.x86_64           1/10
  Installing    : apr-util-bdb-1.6.1-20.el9.x86_64   2/10
  Installing    : apr-util-openssl-1.6.1-20.el9.x86_64 3/10
...output omitted...
Installed:
  apr-1.7.0-11.el9.x86_64                      apr-util-1.6.1-20.el9.x86_64
  apr-util-bdb-1.6.1-20.el9.x86_64            apr-util-openssl-1.6.1-20.el9.x86_64
...output omitted...
Complete!
```

The `dnf update PACKAGE` command obtains and installs a newer version of the specified package, including any dependencies. Generally the process tries to preserve configuration files in place, but in some cases, those files might be renamed if the packager considers that the old name will not work after the update. With no PACKAGE specified, it installs all relevant updates.

```
[root@host ~]# dnf update
```

Because a new kernel can be tested only by booting to that kernel, the package specifically supports the installation of multiple versions at once. If the new kernel fails to boot, then the old kernel is still available. Running the `dnf update kernel` command installs the new kernel. The configuration files hold a list of packages to always install even if the administrator requests an update.

Use the `dnf list kernel` command to list all installed and available kernels. To view the currently running kernel, use the `uname` command. The `uname` command `-r` option shows only the kernel version and release, and the `uname` command `-a` option shows the kernel release and additional information.

```
[user@host ~]$ dnf list kernel
Installed Packages
kernel.x86_64                  5.14.0-70.el9          @System
[user@host ~]$ uname -r
5.14.0-70.el9.x86_64
[user@host ~]$ uname -a
Linux workstation.lab.example.com 5.14.0-70.el9.x86_64 #1 SMP PREEMPT Thu Feb 24
19:11:22 EST 2022 x86_64 x86_64 x86_64 GNU/Linux
```

The `dnf remove PACKAGE` command removes an installed software package, including any supported packages.

```
[root@host ~]# dnf remove httpd
```



### Warning

The `dnf remove` command removes the listed packages *and any package that requires the packages to be removed* (and packages which require those packages, and so on). This command can lead to unexpected removal of packages, so carefully review the list of packages to be removed.

## Install and Remove Groups of Software with DNF

The `dnf` command also has the concept of *groups*, which are collections of related software that are installed together for a particular purpose.

In Red Hat Enterprise Linux 9, the `dnf` command can install two kinds of package groups. Regular groups are collections of packages. Environment groups are collections of regular groups. The packages or groups that these collections provide might be listed as `mandatory` (they must be installed if the group is installed), `default` (normally installed if the group is installed), or `optional` (not installed when the group is installed, unless specifically requested).

Similar to the `dnf list` command, the `dnf group list` command shows the names of installed and available groups.

```
[user@host ~]$ dnf group list
Available Environment Groups:
  Server with GUI
  Server
  Minimal Install
...output omitted...
Available Groups:
  Legacy UNIX Compatibility
  Console Internet Tools
  Container Management
...output omitted...
```

Some groups are normally installed through environment groups and are hidden by default. List these hidden groups with the `dnf group list hidden` command.

The `dnf group info` command displays information about a group. It includes a list of mandatory, default, and optional package names.

```
[user@host ~]$ dnf group info "RPM Development Tools"
Group: RPM Development Tools
Description: Tools used for building RPMs, such as rpmbuild.
Mandatory Packages:
  redhat-rpm-config
  rpm-build
Default Packages:
  rpmdevtools
Optional Packages:
  rpmlint
```

The `dnf group install` command installs a group that installs its mandatory and default packages and their dependent packages.

```
[root@host ~]# dnf group install "RPM Development Tools"
...output omitted...
Installing Groups:
  RPM Development Tools

Transaction Summary
=====
Install 19 Packages

Total download size: 4.7 M
Installed size: 15 M
Is this ok [y/N]: y
...output omitted...
```

**Important**

Starting in Red Hat Enterprise Linux 7, the behavior of Yum groups changed, to be treated as objects and tracked by the system. If an installed group is updated, and if the Yum repository added new mandatory or default packages to the group, then those new packages are installed at update.

RHEL 6 and earlier versions consider a group to be installed if all its mandatory packages are installed, or if it had no mandatory packages, or if any default or optional packages in the group are installed. Starting in RHEL 7, a group is considered to be installed *only* if `yum group install` was used to install it. You can use the `yum group mark install GROUPNAME` command to mark a group as installed, and any missing packages and their dependencies are installed at the next update.

RHEL 6 and earlier versions did not have the two-word form of the `yum group` commands. In other words, in RHEL 6 the command `yum grouplist` existed, but the equivalent RHEL 7 and RHEL 8 `yum group list` command did not.

## View Transaction History

All installation and removal transactions are logged in the `/var/log/dnf.rpm.log` file.

```
[user@host ~]$ tail -5 /var/log/dnf.rpm.log
2022-03-23T16:46:43-0400 SUBDEBUG Installed: python-srpm-macros-3.9-52.el9.noarch
2022-03-23T16:46:43-0400 SUBDEBUG Installed: redhat-rpm-config-194-1.el9.noarch
2022-03-23T16:46:44-0400 SUBDEBUG Installed: elfutils-0.186-1.el9.x86_64
2022-03-23T16:46:44-0400 SUBDEBUG Installed: rpm-build-4.16.1.3-11.el9.x86_64
2022-03-23T16:46:44-0400 SUBDEBUG Installed: rpmdevtools-9.5-1.el9.noarch
```

The `dnf history` command displays a summary of installation and removal transactions.

[root@host ~]# dnf history				
ID	Command line	Date and time	Action(s)	Altered
7	group install RPM Develop	2022-03-23 16:46	Install	20
6	install httpd	2022-03-23 16:21	Install	10 EE
5	history undo 4	2022-03-23 15:04	Removed	20
4	group install RPM Develop	2022-03-23 15:03	Install	20
3		2022-03-04 03:36	Install	5
2		2022-03-04 03:33	Install	767 EE
1	-y install patch ansible-	2022-03-04 03:31	Install	80

The `dnf history undo` command reverses a transaction.

```
[root@host ~]# dnf history undo 6
...output omitted...
Removing:
  apr-util-openssl x86_64 1.6.1-20.el9 @rhel-9.0-for-x86_64-appstream-rpms 24 k
  httpd          x86_64 2.4.51-5.el9 @rhel-9.0-for-x86_64-appstream-rpms 4.7 M
...output omitted...
```

## Summary of DNF Commands

Packages can be located, installed, updated, and removed by name or by package groups.

Task:	Command:
List installed and available packages by name	<code>dnf list [NAME-PATTERN]</code>
List installed and available groups	<code>dnf group list</code>
Search for a package by keyword	<code>dnf search KEYWORD</code>
Show details of a package	<code>dnf info PACKAGE NAME</code>
Install a package	<code>dnf install PACKAGE NAME</code>
Install a package group	<code>dnf group install GROUP NAME</code>
Update all packages	<code>dnf update</code>
Remove a package	<code>dnf remove PACKAGE NAME</code>
Display transaction history	<code>dnf history</code>

## Manage Package Module Streams with DNF

Traditionally, managing alternative versions of an application's software package and its related packages meant maintaining different repositories for each version. For developers who wanted the latest version of an application and administrators who wanted the most stable version of the application, the resulting situation was tedious to manage. Red Hat simplifies this process by using a technology called *Modularity*. With modularity, a single repository can host multiple versions of an application's package and its dependencies.

## Introduction to BaseOS and Application Stream

Red Hat Enterprise Linux 9 distributes the content through two main software repositories: *BaseOS* and *Application Stream* (AppStream).

The *BaseOS* repository provides the core operating system content for Red Hat Enterprise Linux as RPM packages. *BaseOS* components have a lifecycle identical to that of content in previous Red Hat Enterprise Linux releases. The *Application Stream* repository provides content with varying lifecycles as both modules and traditional packages.

*Application Stream* contains necessary parts of the system, as well as a wide range of applications that were previously available as part of Red Hat Software Collections and other products and programs. Each *Application Stream* has a lifecycle that is either the same as Red Hat Enterprise Linux 9 or shorter.

Both *BaseOS* and *AppStream* are necessary parts of a Red Hat Enterprise Linux 9 system.

The *Application Stream* repository contains two types of content: modules and traditional RPM packages. A module describes a set of RPM packages that belong together. Modules can contain several streams to make multiple versions of applications available for installation. Enabling a module stream gives the system access to the RPM packages within that module stream. Typically, modules organize the RPM packages around a specific version of a software application or programming language. A typical module contains packages with an application, packages with

the application's specific dependency libraries, packages with documentation for the application, and packages with helper utilities.



### Important

Red Hat Enterprise Linux 9.0 ships without modules. Future versions of RHEL 9 might introduce additional content and later software versions as modules. Furthermore, starting with RHEL 9, you must manually specify default module streams, as they are no longer defined by default. You can define default module streams with configuration files in the `/etc/dnf/modules.defaults.d/` directory.

## Module Streams

Each module has one or more *module streams*, which hold different versions of the content. Each of the streams receives updates independently. Think of the module stream as a virtual repository in the Application Stream physical repository.

For each module, you can enable only one of its streams, and this stream provides its packages.

## Module Profiles

Each module can have one or more profiles. A profile is a list of packages that you can install together for a particular use-case such as for a server, client, development, minimal installation, or other.

Installing a module profile installs a particular set of packages from the module stream. You can subsequently install or uninstall packages normally. If you do not specify a profile, then the module installs its default profile.

## Manage Modules with DNF

Red Hat Enterprise Linux 9 supports modular features of Application Stream. To handle the modular content, you can use the `dnf module` command. Otherwise, the `dnf` command works with modules similar to regular packages.

You can find some important commands when managing modules in the following list:

- **`dnf module list`** : List the available modules with the module name, stream, profiles, and a summary.
- **`dnf module list module-name`** : List the module streams for a specific module and retrieve their status.
- **`dnf module info module-name`** : Display details of a module, including the available profiles and a list of the packages that the module installs. Running the `dnf module info` command without specifying a module stream lists the packages that are installed from the default profile and stream. Use the `module-name:stream` format to view a specific module stream. Add the `--profile` option to display information about packages that each of the module's profiles installed.
- **`dnf module provides package`** : Display which module provides a specific package.



## References

`dnf(1)` and `dnf.conf(5)` man pages

For more information, refer to the *Managing Software Packages* chapter in the *Red Hat Enterprise Linux 9 Configuring Basic system Settings Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/index#managing-software-packages\\_configuring-basic-system-settings](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#managing-software-packages_configuring-basic-system-settings)

For more information, refer to the *Distribution of Content in RHEL 9* chapter in the *Red Hat Enterprise Linux 9 Managing Software with the DNF Tool Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/managing\\_software\\_with\\_the\\_dnf\\_tool/index#assembly\\_distribution-of-content-in-rhel-9\\_managing-software-with-the-dnf-tool](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_software_with_the_dnf_tool/index#assembly_distribution-of-content-in-rhel-9_managing-software-with-the-dnf-tool)

## Modularity

<https://docs.fedoraproject.org/en-US/modularity/>

## ► Guided Exercise

# Install and Update Software Packages with DNF

In this exercise, you install and remove packages and package groups.

## Outcomes

- Install and remove packages with dependencies.

## Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start software-dnf
```

## Instructions

- 1. From **workstation**, open an SSH session to the **servera** machine as the **student** user. Use the **sudo -i** command to switch to the **root** user.

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

- 2. Search for a specific package.

2.1. Attempt to run the **nmap** command. You should find that it is not installed.

```
[root@servera ~]# nmap  
-bash: nmap: command not found
```

2.2. Use the **dnf search** command to search for packages with **nmap** as part of their name or summary.

```
[root@servera ~]# dnf search nmap  
...output omitted...  
===== Name Exactly Matched: nmap =====  
nmap.x86_64 : Network exploration tool and security scanner  
===== Name & Summary Matched: nmap =====  
nmap-ncat.x86_64 : Nmap's Netcat replacement
```

2.3. Use the **dnf info** command to obtain more information about the **nmap** package.

```
[root@servera ~]# dnf info nmap
...output omitted...
Available Packages
Name        : nmap
Epoch       : 3
Version     : 7.91
Release    : 10.el9
...output omitted...
```

► 3. Use the `dnf install` command to install the `nmap` package.

```
[root@servera ~]# dnf install nmap
...output omitted...
Dependencies resolved.
=====
Package          Arch      Version       Repository      Size
=====
Installing:
  nmap    x86_64    3:7.91-10.el9    rhel-9.0-for-x86_64-appstream-rpms  5.6 M

Transaction Summary
=====
Install 1 Package

Total download size: 5.6 M
Installed size: 24 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

► 4. Remove packages.

- 4.1. Use the `dnf remove` command to remove the `nmap` package, but respond with no when prompted. How many packages are removed?

```
[root@servera ~]# dnf remove nmap
Dependencies resolved.
=====
Package          Arch      Version       Repository      Size
=====
Removing:
  nmap    x86_64    3:7.91-10.el9    @rhel-9.0-for-x86_64-appstream-rpms  24 M

Transaction Summary
=====
Remove 1 Package
```

```
Freed space: 24 M
Is this ok [y/N]: n
Operation aborted.
```

- 4.2. Use the `dnf remove` command to remove the `tar` package, but respond with no when prompted. How many packages are removed?

```
[root@servera ~]# dnf remove tar
...output omitted...
Dependencies resolved.

=====
Package      Arch    Version       Repository      Size
=====
Removing:
tar          x86_64  2:1.34-3.el9   @System           3.0 M
Removing dependent packages:
cockpit      x86_64  264-1.el9     @rhel-9.1-for-x86_64-baseos-rpms  57 k
cockpit-system noarch 264-1.el9   @System           3.3 M
...output omitted...

Transaction Summary
=====
Remove 12 Packages

Freed space: 48 M
Is this ok [y/N]: n
Operation aborted.
```

- 5. Gather information about the "Security Tools" component group and install it on servera.

- 5.1. Use the `dnf group list` command to list all available component groups.

```
[root@servera ~]# dnf group list
```

- 5.2. Use the `dnf group info` command to obtain more information about the Security Tools component group, including a list of included packages.

```
[root@servera ~]# dnf group info "Security Tools"
...output omitted...
Group: Security Tools
Description: Security tools for integrity and trust verification.
Default Packages:
  scap-security-guide
Optional Packages:
  aide
  hmaccalc
  openscap
  openscap-engine-sce
  openscap-utils
  scap-security-guide-doc
  scap-workbench
```

```
tpm2-tools  
tss2  
udica
```

- 5.3. Use the `dnf group install` command to install the Security Tools component group.

```
[root@servera ~]# dnf group install "Security Tools"  
...output omitted...  
Dependencies resolved.  
=====  
 Package      Arch    Version       Repository          Size  
=====  
 Installing group/module packages:  
   scap-security-guide  
         noarch 0.1.60-5.el9  rhel-9.0-for-x86_64-appstream-rpms 683 k  
 Installing dependencies:  
   openscap      x86_64 1:1.3.6-3.el9  rhel-9.0-for-x86_64-appstream-rpms 2.0 M  
 ...output omitted...  
  
Transaction Summary  
=====  
Install 5 Packages  
  
Total download size: 3.0 M  
Installed size: 94 M  
Is this ok [y/N]: y  
...output omitted...  
Installed:  
  openscap-1:1.3.6-3.el9.x86_64  
  openscap-scanner-1:1.3.6-3.el9.x86_64  
  scap-security-guide-0.1.60-5.el9.noarch  
  xmlsec1-1.2.29-9.el9.x86_64  
  xmlsec1-openssl-1.2.29-9.el9.x86_64  
  
Complete!
```

- 6. Explore the history and undo options of the `dnf` command.

- 6.1. Use the `dnf history` command to display recent `dnf` history.

```
[root@servera ~]# dnf history  
ID      | Command line           | Date and time     | Action(s)      | Altered  
----+-----+-----+-----+-----+-----+  
 3 | group install Security T | 2022-03-24 15:23 | Install        | 6  
 2 | install nmap            | 2022-03-24 15:12 | Install        | 1  
 1 | -y install @base firewal | 2022-03-03 04:47 | Install        | 156 EE
```

On your system, the history is probably different.

- 6.2. Use the `dnf history info` command to confirm that the last transaction is the group installation. In the following command, replace the transaction ID with the one from the preceding step.

```
[root@servera ~]# dnf history info 3
Transaction ID : 3
Begin time      : Thu 24 Mar 2022 03:23:56 PM EDT
Begin rpmdb     : 7743aed72ac79f632442c9028aafdf2499a1591f92a660b3f09219b422ca95f02
End time        : Thu 24 Mar 2022 03:23:58 PM EDT (2 seconds)
End rpmdb       : 20c4f0215388b7dca9a874260784b1e5cf9bc142da869967269e3d84dd0f789d
User           : Student User <student>
Return-Code     : Success
Releasever     : 9
Command Line   : group install Security Tools
Comment        :
Packages Altered:
    Install openscap-1:1.3.6-3.el9.x86_64          @rhel-9.0-for-x86_64-
appstream-rpms
    Install openscap-scanner-1:1.3.6-3.el9.x86_64  @rhel-9.0-for-x86_64-
appstream-rpms
...output omitted...
```

- 6.3. Use the `dnf history undo` command to remove the set of packages that were installed when the `nmap` package was installed. On your system, find the correct transaction ID from the output of the `dnf history` command, and then use that ID in the following command.

```
[root@servera ~]# dnf history undo 2
```

- 7. Return to the `workstation` system as the `student` user.

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

## 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 software-dnf
```

This concludes the section.

# Enable DNF Software Repositories

## Objectives

Enable and disable server use of Red Hat or third-party DNF repositories.

## Enable Red Hat Software Repositories

In many cases, systems have access to numerous Red Hat repositories. The `dnf repolist all` command lists all available repositories and their statuses:

```
[user@host ~]$ dnf repolist all
repo id                                repo name          status
rhel-9.0-for-x86_64-appstream-rpms      RHEL 9.0 AppStream  enabled
rhel-9.0-for-x86_64-baseos-rpms         RHEL 9.0 BaseOS    enabled
```



### Note

Red Hat subscriptions grant access to specific repositories. In the past, administrators needed to attach subscriptions on a per-system basis. Simple Content Access (SCA) simplifies how systems access repositories. With SCA, systems can access any repository from any subscription that you purchase, without attaching a subscription. You can enable SCA on the Red Hat Customer Portal within **My Subscriptions > Subscription Allocations**, or on your Red Hat Satellite server.

The `dnf config-manager` command can enable and disable repositories. For example, the following command enables the `rhel-9-server-debug-rpms` repository:

```
[user@host ~]$ dnf config-manager --enable rhel-9-server-debug-rpms
```

Non-Red Hat sources provide software through third-party repositories. For example, Adobe provides some of its software for Linux through DNF repositories. In a Red Hat classroom, the `content.example.com` server hosts DNF repositories. The `dnf` command can access repositories from a website, an FTP server, or the local file system.

You can add a third-party repository in one of two ways. You can either create a `.repo` file in the `/etc/yum.repos.d/` directory, or you can add a `[repository]` section to the `/etc/dnf/dnf.conf` file. Red Hat recommends using `.repo` files, and reserving the `dnf.conf` file for additional repository configurations. The `dnf` command searches both locations by default; however, the `.repo` files take precedence. A `.repo` file contains the URL of the repository, a name, whether to use GPG to check the package signatures, and if so for the latter, the URL to point to the trusted GPG key.

## Add DNF Repositories

The `dnf config-manager` command can also add repositories to the machine. The following command creates a `.repo` file by using an existing repository's URL.

```
[user@host ~]$ dnf config-manager \
--add-repo="https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/"
Adding repo from: https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/
```

The corresponding .repo file is visible in the /etc/yum.repos.d/ directory:

```
[user@host ~]$ cd /etc/yum.repos.d
[user@host yum.repos.d]$ cat \
dl.fedoraproject.org_pub_epel_9_Everything_x86_64_.repo
[dl.fedoraproject.org_pub_epel_9_Everything_x86_64_]
name=created by dnf config-manager from https://dl.fedoraproject.org/pub/epel/9/
Everything/x86_64/
baseurl=https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/
enabled=1
```

Modify this file to customize parameters and to specify the location of a GPG key. Keys are stored in various locations on the remote repository site, such as http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-9. Administrators should download the key to a local file rather than allowing the dnf command to retrieve the key from an external source. For example, the following .repo file uses the gpgkey parameter to reference a local key:

```
[EPEL]
name=EPEL 9
baseurl=https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL - 9
```

## RPM Configuration Packages for Local Repositories

Some repositories provide a configuration file and GPG public key as part of an RPM package to simplify their installation. The dnf install command can download and install these RPM packages.

For example, the following command installs the RHEL9 Extra Packages for Enterprise Linux (EPEL) repository RPM:

```
[user@host ~]$ rpm --import \
http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL - 9
[user@host ~]$ dnf install \
https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

The .repo files often list multiple repository references in a single file. Each repository reference begins with a single-word name in square brackets.

```
[user@host ~]$ cat /etc/yum.repos.d/epel.repo
[epel]
name=Extra Packages for Enterprise Linux $releasever - $basearch
#baseurl=https://download.example/pub/epel/$releasever/Everything/$basearch/
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-$releasever&arch=
$basearch&infra=$infra&content=$contentdir
enabled=1
```

```
gpgcheck=1
countme=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever
...output omitted...
[epel-source]
name=Extra Packages for Enterprise Linux $releasever - $basearch - Source
#baseurl=https://download.example/pub/epel/$releasever/Everything/source/tree/
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-source-
$releasever&arch=$basearch&infra=$infra&content=$contentdir
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever
gpgcheck=1
```

To define a repository, but not to search it by default, insert the `enabled=0` parameter. Although the `dnf config-manager` command persistently enables and disables repositories, the `dnf` command `--enablerepo= PATTERN` and `--disablerepo= PATTERN` options are temporary for the duration of the command.



### Warning

Install the RPM GPG key before installing signed packages, to ensure that packages come from a trusted source. If the RPM GPG key is not installed, then the `dnf` command fails to install signed packages. The `dnf` command `--nogpgcheck` option ignores missing GPG keys, but might result in installing compromised or forged packages.



### References

`dnf(8)`, `dnf.conf(5)`, and `dnf-config-manager(8)` man pages

For more information, refer to the *Managing Software with the DNF Tool* chapter in the Red Hat Enterprise Linux 9 product documentation at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/managing\\_software\\_with\\_the\\_dnf\\_tool](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_software_with_the_dnf_tool)

## ► Guided Exercise

# Enable DNF Software Repositories

In this exercise, you configure your server to get packages from a remote DNF repository, and then update or install a package from that repository.

## Outcomes

- Configure a system to obtain software updates from a classroom server and update the system to use the latest packages.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start software-repo
```

## Instructions

- 1. Use the `ssh` command to log in to the `servera` system as the `student` user. Use the `sudo -i` command to switch to the `root` user.

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

- 2. Configure the software repositories on `servera` to obtain custom packages and updates from the following URL:

- Custom packages provided at [http://content.example.com/rhel9.0/x86\\_64/rhcsa-practice/rht](http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht)
- Updates of the custom packages provided at [http://content.example.com/rhel9.0/x86\\_64/rhcsa-practice/errata](http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata)

- 2.1. Use the `dnf config-manager` command to add the custom packages repository.

```
[root@servera ~]# dnf config-manager \
--add-repo "http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht"
Adding repo from: http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht
```

**Chapter 8 |** Install and Update Software Packages

- 2.2. Examine the software repository file that the previous command created in the /etc/yum.repos.d directory. Use the vim command to edit the file and add the gpgcheck=0 parameter to disable the GPG key check for the repository.

```
[root@servera ~]# vim \
/etc/yum.repos.d/content.example.com_rhel9.0_x86_64_rhcsa-practice_rht.repo
[content.example.com_rhel9.0_x86_64_rhcsa-practice_rht]
name=created by dnf config-manager from http://content.example.com/rhel9.0/x86_64/
rhcsa-practice/rht
baseurl=http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht
enabled=1
gpgcheck=0
```

- 2.3. Create the /etc/yum.repos.d/errata.repo file to enable the updates repository with the following content:

```
[rht-updates]
name=rht updates
baseurl=http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata
enabled=1
gpgcheck=0
```

- 2.4. Use the dnf repolist all command to list all repositories on the system.

```
[root@servera ~]# dnf repolist all
repo id                                repo name      status
content.example.com_rhel9.0_x86_64_rhcsa-practice_rht  created by .... enabled
...output omitted...
rht-updates                               rht updates    enabled
```

- 3. Disable the rht-updates software repository and install the rht-system package.

- 3.1. Use the dnf config-manager --disable command to disable the rht-updates repository.

```
[root@servera ~]# dnf config-manager --disable rht-updates
```

- 3.2. List, and then install, the rht-system package.

```
[root@servera ~]# dnf list rht-system
Available Packages
rht-system.noarch 1.0.0-1 content.example.com_rhel9.0_x86_64_rhcsa-practice_rht
[root@servera ~]# dnf install rht-system
Dependencies resolved.
=====
Package          Arch      Version       Repository      Size
=====
Installing:
rht-system      noarch   1.0.0-1     content..._rht   3.7 k
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

```
Installed:
  rht-system-1.0.0-1.noarch
Complete!
```

- 3.3. Verify that the `rht-system` package is installed, and note the version number of the package.

```
[root@servera ~]# dnf list rht-system
Installed Packages
rht-system.noarch  1.0.0-1 @content.example.com_rhel9.0_x86_64_rhcsa-practice_rht
```

- 4. Enable the `rht-updates` software repository and update all relevant software packages.

- 4.1. Use `dnf config-manager --enable` to enable the `rht-updates` repository.

```
[root@servera ~]# dnf config-manager --enable rht-updates
```

- 4.2. Use the `dnf update` command to update all software packages on `servera`.

```
[root@servera ~]# dnf update
Dependencies resolved.
=====
Package           Arch      Version       Repository      Size
=====
Upgrading:
  rht-system      noarch   1.0.0-2      rht-updates    7.5 k
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 4.3. Verify that the `rht-system` package is upgraded, and note the version number of the package.

```
[root@servera ~]# dnf list rht-system
Installed Packages
rht-system.noarch          1.0.0-2           @rht-updates
```

- 5. Exit from `servera`.

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

## 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 software-repo
```

This concludes the section.

## ► Lab

# Install and Update Software Packages

In this lab, you manage software repositories, and install and upgrade packages from those repositories.

## Outcomes

- Manage software repositories.
- Install and upgrade packages from repositories.
- Install an RPM package.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. On the `serverb` machine, configure a software repository to obtain updates. Name the repository `errata` and configure the repository in the `/etc/yum.repos.d/errata.repo` file. Configure the `errata.repo` file to use the `http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata` repository. Do not check GPG signatures.
2. On `serverb`, install the `rht-system` package.
3. For security reasons, the `serverb` machine must not be able to connect to a paper printer. You can achieve this effect by removing the `cups` package. When finished, exit from the `root` shell.
4. The start script downloads the `rhcsa-script-1.0.0-1.noarch.rpm` package in the `/home/student` directory on the `serverb` machine.  
Confirm that the `rhcsa-script-1.0.0-1.noarch.rpm` package is available on `serverb` and install it using `root` privileges. Verify that the package is installed. Exit from the `serverb` 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 software-review
```

## 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 software-review
```

This concludes the section.

## ► Solution

# Install and Update Software Packages

In this lab, you manage software repositories, and install and upgrade packages from those repositories.

## Outcomes

- Manage software repositories.
- Install and upgrade packages from repositories.
- Install an RPM package.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. On the `serverb` machine, configure a software repository to obtain updates. Name the repository `errata` and configure the repository in the `/etc/yum.repos.d/errata.repo` file. Configure the `errata.repo` file to use the `http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata` repository. Do not check GPG signatures.

- 1.1. Log in to the `serverb` machine as the `student` user and switch to the `root` user.

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

- 1.2. Create the `/etc/yum.repos.d/errata.repo` file with the following content:

```
[errata]
name=Red Hat Updates
baseurl=http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata
enabled=1
gpgcheck=0
```

2. On `serverb`, install the `rht-system` package.

- 2.1. List the available packages for the `rht-system` package.

```
[root@serverb ~]# dnf list rht-system
Last metadata expiration check: 0:05:27 ago on Wed 27 Apr 2022 05:01:59 AM EDT.
Available Packages
rht-system.noarch      1.0.0-2                  errata
```

2.2. Install the latest version of the `rht-system` package.

```
[root@serverb ~]# dnf install rht-system
...output omitted...
Total download size: 7.5 k
Installed size: 300
Is this ok [y/N]: y
...output omitted...
Complete!
[root@serverb ~]#
```

3. For security reasons, the `serverb` machine must not be able to connect to a paper printer. You can achieve this effect by removing the `cups` package. When finished, exit from the `root` shell.

3.1. List the installed `cups` package.

```
[root@serverb ~]# dnf list cups
Last metadata expiration check: 0:08:02 ago on Wed 27 Apr 2022 05:01:59 AM EDT.
Installed Packages
cups.x86_64      1:2.3.3op2-13.el9      @rhel-9.0-for-x86_64-appstream-rpms
[root@serverb ~]#
```

3.2. Remove the `cups` package.

```
[root@serverb ~]# dnf remove cups.x86_64
...output omitted...
Remove 46 Packages

Freed space: 94 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

3.3. Exit from the `root` shell.

```
[root@serverb ~]# exit
[student@serverb ~]$
```

4. The start script downloads the `rhcsa-script-1.0.0-1.noarch.rpm` package in the `/home/student` directory on the `serverb` machine. Confirm that the `rhcsa-script-1.0.0-1.noarch.rpm` package is available on `serverb` and install it using `root` privileges. Verify that the package is installed. Exit from the `serverb` machine.

- 4.1. Verify that the `rhcsa-script-1.0.0-1.noarch.rpm` package is available on serverb.

```
[student@serverb ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm -i
Name        : rhcsa-script
Version     : 1.0.0
Release     : 1
Architecture: noarch
Install Date: (not installed)
Group       : System
Size        : 593
License     : GPL
Signature   : (none)
Source RPM  : rhcsa-script-1.0.0-1.src.rpm
Build Date  : Wed 23 Mar 2022 08:24:21 AM EDT
Build Host  : localhost
Packager    : Bernardo Gargallo
URL         : http://example.com
Summary     : RHCSA Practice Script
Description  :
A RHCSA practice script.
The package changes the motd.
```

- 4.2. Install the `rhcsa-script-1.0.0-1.noarch.rpm` package.

```
[student@serverb ~]$ sudo dnf install \
rhcsa-script-1.0.0-1.noarch.rpm
[sudo] password for student: student
Last metadata expiration check: 0:11:06 ago on Wed 27 Apr 2022 05:01:59 AM EDT.
Dependencies resolved.
=====
Package      Architecture Version   Repository Size
=====
Installing:
rhcsa-script  noarch      1.0.0-1   @commandline 7.5 k

Transaction Summary
=====
Install 1 Package

Total size: 7.5 k
Installed size: 593
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing          : 1/1
Running scriptlet: rhcsa-script-1.0.0-1.noarch 1/1
Installing       : rhcsa-script-1.0.0-1.noarch 1/1
Running scriptlet: rhcsa-script-1.0.0-1.noarch 1/1
Verifying        : rhcsa-script-1.0.0-1.noarch 1/1
```

```
Installed:  
    rhcsa-script-1.0.0-1.noarch  
  
Complete!
```

4.3. Verify that the package is installed.

```
[student@serverb ~]$ rpm -q rhcsa-script  
rhcsa-script-1.0.0-1.noarch  
[student@serverb ~]$
```

4.4. Return to the workstation system 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 software-review
```

## 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 software-review
```

This concludes the section.

# Summary

---

- Red Hat Subscription Management provides tools to entitle machines to product subscriptions, get updates to software packages, and track information about support contracts and subscriptions that the systems use.
- The `dnf` utility is a powerful command-line tool to install, update, remove, and query software packages.
- You can use the `dnf config-manager` command to enable and disable DNF repositories.

## Chapter 9

# Manage Basic Storage

### Goal

Create and manage storage devices, partitions, file systems, and swap spaces from the command line.

### Objectives

- Access the contents of file systems by adding and removing file systems in the file-system hierarchy.
- Create storage partitions, format them with file systems, and mount them for use.
- Create and manage swap spaces to supplement physical memory.

### Sections

- Mount and Unmount File Systems (and Guided Exercise)
- Add Partitions, File Systems, and Persistent Mounts (and Guided Exercise)
- Manage Swap Space (and Guided Exercise)

### Lab

- Manage Basic Storage

# Mount and Unmount File Systems

---

## Objectives

Access the contents of file systems by adding and removing file systems in the file-system hierarchy.

## Mount File Systems Manually

To access the file system on a removable storage device, you must mount the storage device. With the `mount` command, the `root` user can mount a file system manually. The first argument of the `mount` command specifies the file system to mount. The second argument specifies the directory as the mount point in the file-system hierarchy.

You can mount the file system in one of the following ways with the `mount` command:

- With the device file name in the `/dev` directory.
- With the UUID, a universally unique identifier of the device.

Then, identify the device to mount, ensure that the mount point exists, and mount the device on the mount point.



### Note

If you mount a file system with the `mount` command, and then reboot your system, the file system is not automatically remounted. The *Red Hat System Administration II* (RH134) course explains how to persistently mount file systems with the `/etc/fstab` file.

## Identify a Block Device

A hot-pluggable storage device, whether a hard disk drive (HDD) or a solid-state device (SSD) in a server, or alternatively a USB storage device, might be plugged each time into a different port on a system. Use the `lsblk` command to list the details of a specified block device or of all the available devices.

```
[root@host ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
vda    252:0    0 10G  0 disk
└─vda1 252:1    0  1M  0 part
└─vda2 252:2    0 200M 0 part /boot/efi
└─vda3 252:3    0 500M 0 part /boot
└─vda4 252:4    0 9.3G 0 part /
vdb    252:16   0   5G  0 disk
vdc    252:32   0   5G  0 disk
vdd    252:48   0   5G  0 disk
```

The partition size helps to identify the device when the partition name is unknown. For example, considering the previous output, if the size of the identified partition is 9.3 GB, then mount the `/dev/vda4` partition.

## Mount File System with the Partition Name

The following example mounts the /dev/vda4 partition on the /mnt/data mount point.

```
[root@host ~]# mount /dev/vda4 /mnt/data
```

The mount point directory must exist before mounting the file system. The /mnt directory exists for use as a temporary mount point.



### Important

If a directory to be used as a mount point is not empty, then the existing files will be hidden and not accessible while a file system is mounted there. The original files will be accessible again after the mounted file system is unmounted.

Device detection order and storage device naming can change when devices are added or removed on a system. It is recommended to use an unchanging device identifier for mount file systems consistently.

## Mount File System with Partition UUID

One stable identifier that is associated with a file system is its universally unique identifier (UUID). This UUID is stored in the file system superblock and remains the same until the file system is re-created.

The `lsblk -fp` command lists the full path of the device, the UUIDs and mount points, and the partition's file-system type. The mount point is blank when the file system is not mounted.

```
[root@host ~]# lsblk -fp
  NAME      FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE% MOUNTPOINTS
  /dev/vda
    └─/dev/vda1
    └─/dev/vda2  vfat   FAT16    7B77-95E7          192.3M     4% /boot/efi
    └─/dev/vda3  xfs    boot    2d67e6d0-...-1f091bf1  334.9M    32% /boot
    └─/dev/vda4  xfs    root    efd314d0-...-ae98f652    7.7G    18% /
  /dev/vdb
  /dev/vdc
  /dev/vdd
```

Mount the file system by the file-system UUID.

```
[root@host ~]# mount UUID="efd314d0-b56e-45db-bbb3-3f32ae98f652" /mnt/data
```

## Automatically Mount Removable Storage Devices

When using the graphical desktop environment, the system automatically mounts removable storage media when the media presence is detected.

The removable storage device mounts at the `/run/media/USERNAME/LABEL` location. *USERNAME* is the name of the user that is logged in to the graphical environment. *LABEL* is an identifier, which is typically the label on the storage media.

To safely detach a removable device, manually unmount all file systems on the device first.

## Unmount File Systems

System shutdown and reboot procedures unmount all file systems automatically. All file-system data that is flushed to the storage device, to ensure file system data integrity.



### Warning

File-system data uses memory cache during normal operation. You must unmount a removable drive's file systems before unplugging the drive. The unmount procedure flushes data to disk before releasing the drive.

The `umount` command uses the mount point as an argument to unmount a file system.

```
[root@host ~]# umount /mnt/data
```

Unmounting is not possible when the mounted file system is in use. All processes must stop accessing data under the mount point for the `umount` command to succeed.

In the following example, the `umount` command fails because the shell uses the `/mnt/data` directory as its current working directory, and thus generates an error message.

```
[root@host ~]# cd /mnt/data
[root@host data]# umount /mnt/data
umount: /mnt/data: target is busy.
```

The `lsof` command lists all open files and the processes that are accessing the file system. The list helps to identify which processes are preventing the file system from successfully unmounting.

```
[root@host data]# lsof /mnt/data
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash    1593 root cwd DIR 253,17      6  128 /mnt/data
lsof    2532 root cwd DIR 253,17     19  128 /mnt/data
lsof    2533 root cwd DIR 253,17     19  128 /mnt/data
```

After identifying the processes, wait for the processes to complete or send the `SIGTERM` or `SIGKILL` signal to terminate them. In this case, it is sufficient to change the current working directory to a directory outside the mount point.

```
[root@host data]# cd
[root@host ~]# umount /mnt/data
```



### References

`lsblk(8)`, `mount(8)`, `umount(8)`, and `lsof(8)` man pages.

## ► Guided Exercise

# Mount and Unmount File Systems

In this exercise, you practice mounting and unmounting file systems.

## Outcomes

- Identify and mount a new file system at a specified mount point, and then unmount the file system.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start fs-mount
```

## 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. A new partition with a file system is added to the `/dev/vdb` disk on the `servera` machine. Mount the newly available partition by using the UUID at the `/mnt/part1` mount point.

- 2.1. Create the `/mnt/part1` directory.

```
[root@servera ~]# mkdir /mnt/part1
```

- 2.2. Query the UUID of the `/dev/vdb1` device.

NAME	FSTYPE	LABEL	UUID	MOUNTPOINT
<code>/dev/vdb</code>				
<code>└─/dev/vdb1</code>	xfs		<code>a04c511a-b805-4ec2-981f-42d190fc9a65</code>	

- 2.3. Mount the file system by using the UUID on the `/mnt/part1` directory. Use the `/dev/vdb1` UUID from the previous command output.

```
[root@servera ~]# mount \
UUID="a04c511a-b805-4ec2-981f-42d190fc9a65" /mnt/part1
```

2.4. Verify that the /dev/vdb1 device is mounted on the /mnt/part1 directory.

```
[root@servera ~]# lsblk -fp /dev/vdb
NAME      FSTYPE LABEL UUID                                     MOUNTPOINT
/dev/vdb
└─/dev/vdb1 xfs   a04c511a-b805-4ec2-981f-42d190fc9a65 /mnt/part1
```

- 3. Change to the /mnt/part1 directory and create the testdir subdirectory. Create the /mnt/part1/testdir/newmount file.

3.1. Change to the /mnt/part1 directory.

```
[root@servera ~]# cd /mnt/part1
```

3.2. Create the /mnt/part1/testdir directory.

```
[root@servera part1]# mkdir testdir
```

3.3. Create the /mnt/part1/testdir/newmount file.

```
[root@servera part1]# touch testdir/newmount
```

- 4. Unmount the file system that is mounted on the /mnt/part1 directory.

4.1. Unmount the /mnt/part1 directory while the shell is in the /mnt/part1 directory. The umount command fails to unmount the device.

```
[root@servera part1]# umount /mnt/part1
umount: /mnt/part1: target is busy.
```

4.2. Change the current directory on the shell to the /root directory.

```
[root@servera part1]# cd
[root@servera ~]#
```

4.3. Unmount the /mnt/part1 directory.

```
[root@servera ~]# umount /mnt/part1
```

- 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, 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 fs-mount
```

This concludes the section.

# Add Partitions, File Systems, and Persistent Mounts

---

## Objectives

Create storage partitions, format them with file systems, and mount them for use.

## Partition Disks

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 numerous 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.

## MBR Partition Scheme

The *Master Boot Record* (MBR) partitioning scheme is the standard on systems that run 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 9.1: MBR partitioning of the /dev/vdb 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.

## GPT Partition Scheme

For systems that run *Unified Extensible Firmware Interface* (UEFI) firmware, GPT is the standard for disk partitioning and addresses the limitations of the MBR scheme. A GPT 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 9.2: GPT partitioning of the /dev/vdb storage device**

GPT partitioning offers additional features and benefits over MBR. A GPT uses a *globally unique identifier* (GUID) to identify each disk and partition. A GPT makes the partition table redundant,

with the primary GPT at the head of the disk, and a backup secondary GPT at the end of the disk. A GPT uses a checksum to detect errors in the GPT header and partition table.

## Manage Partitions

An administrator can use a *partition editor* program to change a disk's partitions, such as creating and deleting partitions, and changing partition types.



### Note

Which partition editor should you use? Although IT professionals have strong opinions about feature distinctions, each listed editor here performs common disk preparation tasks successfully.

- `fdisk` is a historical favorite and has supported GPT partitions for years.
- `gdisk` and other `fdisk` variants were initially created to support GPT.
- `parted` and the `libparted` library have been the RHEL standard for years.
- The Anaconda installer continues to use the `libparted` library.
- `gnome-disk` is the default GNOME graphical tool, replacing `gparted` upstream.
- Almost all CLI editors are good for scripting, and `parted` was designed for it.

Administrators can use the `parted` partition editor for both the MBR and the GPT partitioning scheme. The `parted` command takes the whole disk device name as the first argument, followed by subcommands. The following example uses the `print` subcommand to display the partition table on the `/dev/vda` disk.

```
[root@host ~]# parted /dev/vda print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 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
```

Use the `parted` command without a subcommand to open an interactive partitioning session.

```
[root@host ~]# parted /dev/vda
GNU Parted 3.4
Using /dev/vda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 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
```

```
(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)

```
[root@host ~]# parted /dev/vda unit s print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 104857600s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Partition Flags:

Number  Start      End        Size       Type      File system  Flags
 1      2048s    20971486s  20969439s  primary   xfs          boot
 2      20971520s 104857535s  83886016s  primary   xfs
```

As shown in the previous example, you can also specify multiple subcommands (here, `unit` and `print`) on the same line.

## 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 `parted` to write an MBR disk label or a GPT disk label.

```
[root@host ~]# parted /dev/vdb mklabel msdos
[root@host ~]# parted /dev/vdb mklabel gpt
```



### Warning

The `mklabel` subcommand wipes the existing partition table. Use the `mklabel` subcommand when the intent is 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.

Run the `parted` command and specify the disk device name as an argument, to start in interactive mode. The session displays (`parted`) as a subcommand prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Use the `mkpart` subcommand to create a primary or extended partition.

```
(parted) mkpart
Partition type? primary/extended? primary
```



### Note

If you need 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.

Indicate the file-system type that you want to create on the partition, such as `xfs` or `ext4`. This value does not create the file system, but it is only a useful partition type label.

```
File system type? [ext2]? xfs
```

To list the supported file-system types, use the following command:

```
[root@host ~]# parted /dev/vdb help mkpart
...output omitted...
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, asfs, amufs5, amufs4, amufs3,
amufs2, amufs1, amufs0, amufs, affs7, affs6, affs5, affs4, affs3, affs2,
affs1, affs0, linux-swap, linux-swap(new), linux-swap(old)

'mkpart' makes a partition without creating a new file system on the
partition. FS-TYPE may be specified to set an appropriate partition
ID.
```

Specify the disk sector to start the new partition on.

```
Start? 2048s
```

The `s` suffix provides the value in sectors, or uses 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 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

## Chapter 9 | Manage Basic Storage

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 new partition should end, and exit `parted`. You can specify the end as a size or as an ending location.

```
End? 1000MB
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

When you provide the end position, the `parted` command updates the partition table on the disk with the new partition details.

Run the `udevadm settle` command. This command waits for the system to detect the new partition and to create the associated device file under the `/dev` directory. The prompt returns when the task is done.

```
[root@host ~]# udevadm settle
```

As an alternative to interactive mode, you can create a partition in a single command:

```
[root@host ~]# parted /dev/vdb 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.

As the `root` user, execute the `parted` command and specify the disk device name as an argument.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
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
```

Indicate the file-system type that you want to create on the partition, such as `xfs` or `ext4`. This value does not create the file system, but is a useful partition type label.

```
File system type? [ext2]? xfs
```

Specify the disk sector that the new partition starts on.

```
Start? 2048s
```

Specify the disk sector where the new partition should end, and exit `parted`. When you provide the end position, the `parted` command updates the GPT on the disk with the new partition details.

```
End? 1000MB
(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 under the `/dev` directory. The prompt returns when the task is done.

```
[root@host ~]# udevadm settle
```

As an alternative to interactive mode, you can create a partition in a single command:

```
[root@host ~]# parted /dev/vdb mkpart userdata xfs 2048s 1000MB
```

## Delete 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/vdb
GNU Parted 3.4
Using /dev/vdb
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: Virtio Block Device (virtblk)
Disk /dev/vdb: 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, and exit `parted`. The `rm` subcommand immediately deletes the partition from the partition table on the disk.

```
(parted) rm 1
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

As an alternative to interactive mode, you can delete a partition in a single command:

```
[root@host ~]# parted /dev/vdb rm 1
```

## Create File Systems

After a block device is created, 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 apply an XFS file system to a block device. For an ext4 file system, use the `mkfs.ext4` command.

```
[root@host ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=60992 blks
                                =          sectsz=512   attr=2, projid32bit=1
                                =          crc=1      finobt=1, sparse=1, rmapbt=0
                                =          reflink=1 bigtime=1 inobtcount=1
data     =          bsize=4096   blocks=243968, imaxpct=25
        =          sunit=0     swidth=0 blks
naming  =version 2              bsize=4096   ascii-ci=0, ftype=1
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
```

## Mount 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 onto 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 mount point, and can include file-system mount options. File-system options customize the behavior of the file system.

```
[root@host ~]# mount /dev/vdb1 /mnt
```

You also use the `mount` command to view currently mounted file systems, the mount points, and their options.

```
[root@host ~]# mount | grep vdb1
/dev/vdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

## Persistently Mount File Systems

Manually mounting a file system is a good way to verify that a formatted device is accessible and 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 white-space-delimited file with six fields per line.

```
[root@host ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Thu Apr 5 12:05:19 2022
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
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/vdb1`.

The second field is the directory mount point, from which the block device is accessible in the directory structure. The mount point must exist; if not, create it with the `mkdir` command.

The third field contains the file-system type, such as `xfs` or `ext4`.

The fourth field is the comma-separated list of options to apply to the device. `defaults` 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 the `fsck` command should be run at system boot to verify that the file systems are clean. The value in this field indicates the order in which `fsck` should run. For XFS file systems, set this field to `0`, because XFS does not use `fsck` to check its file-system status. For `ext4` file systems, set it to `1` for the root file system, and `2` for the other `ext4` file systems. By using this notation, the `fsck` utility processes the root file system first and then checks file systems on separate disks concurrently, and file systems on the same disk in sequence.

**Note**

An incorrect entry in `/etc/fstab` might render the machine non-bootable. Verify that an entry is valid by manually unmounting the new file system and then by using `mount /mountpoint` 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 --verify` command 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, because 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 --fs` command to scan the block devices that are connected to a machine and retrieve the file-system UUIDs.

```
[root@host ~]# lsblk --fs
NAME   FSTYPE  FSVER  LABEL      UUID          FSAVAIL FSUSE% MOUNTPOINTS
vda
└─vda1
└─vda2 xfs      boot    49dd...75fdf  312M    37%    /boot
└─vda3 xfs      root    8a90...ce0da  4.8G    48%    /
```

**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 *Configuring and Managing File Systems* guide at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/managing\\_file\\_systems/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_file_systems/index)

## ► Guided Exercise

# Add Partitions, File Systems, and Persistent Mounts

In this exercise, you create a partition on a new storage device, format it with an XFS file system, configure it to mount at boot, and mount it for use.

### Outcomes

- Use the `parted`, `mkfs.xfs`, and other commands to create a partition on a new disk, format it, and persistently mount it.

### Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start storage-partitions
```

### Instructions

- 1. Log in to `servera` 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. Create an `msdos` disk label on the `/dev/vdb` device.

```
[root@servera ~]# parted /dev/vdb mklabel msdos
Information: You may need to update /etc/fstab.
```

- 3. Add a 1 GB primary partition. For proper alignment, start the partition at the sector 2048. Set the partition file-system type to XFS.

- 3.1. Use `parted` interactive mode to create the partition.

```
[root@servera ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mkpart
```

```

Partition type? primary/extended? primary
File system type? [ext2]? xfs
Start? 2048s
End? 1001MB
(parted) quit
Information: You may need to update /etc/fstab.

```

Because the partition starts at the sector 2048, the previous command sets the end position to 1001 MB to get a partition size of 1000 MB (1 GB).

Alternatively, you can perform the same operation with the following non-interactive command: `parted /dev/vdb mkpart primary xfs 2048s 1001 MB`

- 3.2. Verify your work by listing the partitions on the `/dev/vdb` device.

```

[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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

```

- 3.3. Run the `udevadm settle` command. This command waits for the system to register the new partition and returns when it is done.

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

- 4. Format the new partition with the XFS file system.

```

[root@servera ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=61056 blks
                                =                      sectsz=512  attr=2, projid32bit=1
                                =                      crc=1      finobt=1, sparse=1, rmapbt=0
                                =                      reflink=1 bigtime=1 inobtcount=1
data     =                      bsize=4096   blocks=244224, imaxpct=25
                                =                      sunit=0    swidth=0 blks
naming   =version 2             bsize=4096   ascii-ci=0, ftype=1
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

```

- 5. Configure the new file system to mount onto the `/archive` directory persistently.

- 5.1. Create the `/archive` directory.

```
[root@servera ~]# mkdir /archive
```

- 5.2. Discover the UUID of the `/dev/vdb1` device. The UUID in the output is probably different on your system.

```
[root@servera ~]# lsblk --fs /dev/vdb
NAME   FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE%
MOUNTPOINTS
vdb
└─vdb1 xfs            881e856c-37b1-41e3-b009-ad526e46d987
```

- 5.3. Add an entry to the `/etc/fstab` file. Replace the UUID with the one that you discovered from the previous step.

```
...output omitted...
UUID=881e856c-37b1-41e3-b009-ad526e46d987 /archive xfs defaults 0 0
```

- 5.4. Update the `systemd` daemon for the system to register the new `/etc/fstab` file configuration.

```
[root@servera ~]# systemctl daemon-reload
```

- 5.5. Mount the new file system with the new entry in the `/etc/fstab` file.

```
[root@servera ~]# mount /archive
```

- 5.6. Verify that the new file system is mounted onto the `/archive` directory.

```
[root@servera ~]# mount | grep /archive
/dev/vdb1 on /archive type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- 6. Reboot `servera`. After the server rebooted, log in and verify that the `/dev/vdb1` device is mounted on the `/archive` directory. When done, log out from `servera`.

- 6.1. Reboot `servera`.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- 6.2. Wait for `servera` to reboot and log in as the `student` user.

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

- 6.3. Verify that the `/dev/vdb1` device is mounted on the `/archive` directory.

```
[student@servera ~]$ mount | grep /archive
/dev/vdb1 on /archive type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

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

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

## 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 storage-partitions
```

This concludes the section.

# Manage Swap Space

## Objectives

Create and manage swap spaces 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 needed page from the swap area.

Because swap areas reside 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.

### 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
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 need a complete boot. For those systems, the swap space must be greater than the amount of RAM.

The Knowledgebase article in References at the end of this section gives more guidance about sizing the swap space.

## Create Swap Space

To create a swap space, you need to 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 file-system type whether to activate the device; however, that requirement is no longer the case. Even though utilities no longer use the partition file-system type, administrators can quickly determine the partition's purpose from that type.

The following example creates a 256 MB partition.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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

(parted) mkpart
Partition name? []? swap1
File system type? [ext2]? linux-swap
Start? 1001MB
End? 1257MB
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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

(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

After creating the partition, 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. The command returns only when it is finished.

```
[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, leaving the rest of the device unformatted so that the kernel can use it for storing memory pages.

```
[root@host ~]# mkswap /dev/vdb2
Setting up swap space version 1, size = 244 MiB (255848448 bytes)
no label, UUID=39e2667a-9458-42fe-9665-c5c854605881
```

## Activate Swap Space

You can use the `swapon` command to activate a formatted swap space.

Use `swapon` with the device as a parameter, or use `swapon -a` to activate all the listed swap spaces in the `/etc/fstab` file. Use the `swapon --show` and `free` commands to inspect the available swap spaces.

```
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:       1873036     134688     1536436        16748      201912     1576044
Swap:          0          0          0
[root@host ~]# swapon /dev/vdb2
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:       1873036     135044     1536040        16748      201952     1575680
Swap:    249852          0     249852
```

You can deactivate a swap space with 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 an active swap space 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 UUID as the first field. When you format the device, the `mkswap` command displays that UUID. If you lost the output of `mkswap`, then use the `lsblk --fs` command. As an alternative, you can use the device name in the first field.

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 a `none` placeholder value; however, a `swap` value gives 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 `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 `systemd` to register the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

## Set Swap Space Priority

By default, the system uses swap spaces in series, meaning that the kernel uses the first activated swap space until it is full, and then it starts using the second swap space. You can instead define a priority for each swap space to force a particular order.

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 --show` command to display the swap space priorities.

When swap spaces have the same priority, the kernel writes to them in a round-robin fashion.



### References

`mkswap(8)`, `swapon(8)`, `swapoff(8)`, `mount(8)`, and `parted(8)` man pages

### Knowledgebase: What Is the Recommended Swap Size for Red Hat Platforms?

<https://access.redhat.com/solutions/15244>

## ► Guided Exercise

# Manage Swap Space

In this exercise, you create and format a partition for use as swap space, format it as swap, and activate it persistently.

## Outcomes

- Create a partition and a swap space on a disk by using the GPT partitioning scheme.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start storage-swap
```

## Instructions

- 1. Log in to `servera` 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. Inspect the `/dev/vdb` disk. The disk already has a partition table and uses the GPT partitioning scheme. Also, it has an existing 1 GB partition.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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. Add a new partition of 500 MB for use as swap space. Set the partition type to `linux-swap`.

- 3.1. Create the `myswap` partition. Because the disk uses the GPT partitioning scheme, you must give a name to the partition. Notice that the start position, 1001 MB, is the end of the existing first partition. The `parted` command ensures that the

## Chapter 9 | Manage Basic Storage

new partition immediately follows the previous one, without any gap. Because the partition starts at the 1001 MB position, the command sets the end position to 1501 MB to get a partition size of 500 MB.

```
[root@servera ~]# parted /dev/vdb mkpart myswap linux-swap \
1001MB 1501MB
Information: You may need to update /etc/fstab.
```

- 3.2. Verify your work by listing the partitions on the /dev/vdb 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/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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. Run the `udevadm settle` command. This command waits for the system to register the new partition and returns when it is done.

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

- 4. Initialize the new partition as swap space.

```
[root@servera ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 476 MiB (499118080 bytes)
no label, UUID=cb7f71ca-ee82-430e-ad4b-7dda12632328
```

- 5. Enable the new swap space.

- 5.1. Verify that creating and initializing the swap space does not yet enable it for use.

```
[root@servera ~]# swapon --show
```

- 5.2. Enable the new swap space.

```
[root@servera ~]# swapon /dev/vdb2
```

- 5.3. Verify that the new swap space is now available.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

- 5.4. Disable the swap space.

```
[root@servera ~]# swapoff /dev/vdb2
```

- 5.5. Confirm that the swap space is disabled.

```
[root@servera ~]# swapon --show
```

- 6. Enable the new swap space at system boot.

- 6.1. Use the `lsblk` command with the `--fs` option to discover the UUID of the `/dev/vdb2` device. The UUID in the output will be different on your system.

```
[root@servera ~]# lsblk --fs /dev/vdb2
NAME FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE%
MOUNTPOINTS
vdb2 swap    1          762735cb-a52a-4345-9ed0-e3a68aa8bb97
```

- 6.2. Add an entry to the `/etc/fstab` file. In the following command, replace the UUID with the one that you discovered from the previous step.

```
...output omitted...
UUID=762735cb-a52a-4345-9ed0-e3a68aa8bb97  swap  swap  defaults  0 0
```

- 6.3. Update the `systemd` daemon for the system to register the new `/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/vdb2 partition 476M   0B   -2
```

- 7. Reboot the `servera` machine. After the server reboots, log in and verify that the swap space is enabled. When done, log out from `servera`.

- 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 `servera` 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/vdb2 partition 476M   0B    -2
```

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

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

## 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 storage-swap
```

This concludes the section.

## ▶ Lab

# Manage Basic Storage

In this lab, you create several partitions on a new disk, formatting some with file systems and mounting them, and activating others as swap spaces.

## Outcomes

- Display and create partitions with the `parted` command.
- Create file systems on partitions and persistently mount them.
- Create swap spaces and activate them at boot.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. The `serverb` machine has several unused disks. On the first unused disk, create a 2 GB GPT backup partition. Because it is difficult to set an exact size, a size between 1.8 GB and 2.2 GB is acceptable. Configure the backup partition to host an XFS file system.
2. Format the 2 GB backup partition with an XFS file system and persistently mount it to the `/backup` directory.
3. On the same disk, create two 512 MB GPT partitions called `swap1` and `swap2`. A size between 460 MB and 564 MB is acceptable. Configure the file-system types of the partitions to host swap spaces.
4. Initialize the two 512 MiB partitions as swap spaces and configure them to activate at boot. Set the swap space on the `swap2` partition to be preferred over the other.
5. To verify your work, reboot the `serverb` machine. Confirm that the system automatically mounts the first partition onto the `/backup` directory. Also, confirm that the system activates the two swap spaces.

## 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, 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 storage-review
```

This concludes the section.

## ► Solution

# Manage Basic Storage

In this lab, you create several partitions on a new disk, formatting some with file systems and mounting them, and activating others as swap spaces.

## Outcomes

- Display and create partitions with the `parted` command.
- Create file systems on partitions and persistently mount them.
- Create swap spaces and activate them at boot.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. The `serverb` machine has several unused disks. On the first unused disk, create a 2 GB GPT backup partition. Because it is difficult to set an exact size, a size between 1.8 GB and 2.2 GB is acceptable. Configure the backup partition to host an XFS file system.

- 1.1. Log in to `serverb` as the `student` user and switch to the `root` user.

```
[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 first unused disk, `/dev/vdb`, does not have any partitions.

```
[root@serverb ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
vda    252:0    0   10G  0 disk
└─vda1 252:1    0    1M  0 part
└─vda2 252:2    0  200M  0 part /boot/efi
└─vda3 252:3    0  500M  0 part /boot
└─vda4 252:4    0  9.3G  0 part /
vdb    252:16   0    5G  0 disk
vdc    252:32   0    5G  0 disk
vdd    252:48   0    5G  0 disk
```

- 1.3. Confirm that the /dev/vdb disk has no label.

```
[root@serverb ~]# parted /dev/vdb print
Error: /dev/vdb: unrecognised disk label
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

- 1.4. Define the GPT partitioning scheme.

```
[root@serverb ~]# parted /dev/vdb mklabel gpt
Information: You may need to update /etc/fstab.
```

- 1.5. Create the 2 GB backup partition with an xfs file-system type. Start the partition at sector 2048.

```
[root@serverb ~]# parted /dev/vdb 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/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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. Run the udevadm settle command. This command waits for the system to detect the new partition and to create the /dev/vdb1 device file.

```
[root@serverb ~]# udevadm settle
```

2. Format the 2 GB backup partition with an XFS file system and persistently mount it to the /backup directory.

- 2.1. Format the /dev/vbd1 partition.

```
[root@serverb ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=121984 blks
                                =           sectsz=512  attr=2, projid32bit=1
                                =           crc=1      finobt=1, sparse=1, rmapbt=0
                                =           reflink=1 bigtime=1 inobtcount=1
data     =           bsize=4096   blocks=487936, imaxpct=25
                                =           sunit=0    swidth=0 blks
naming   =version 2             bsize=4096   ascii-ci=0, ftype=1
```

```
log      =internal log          bsize=4096  blocks=2560, version=2
        =
realtime =none                sectsz=512   sunit=0 blks, lazy-count=1
                                extsz=4096  blocks=0, rtextents=0
```

- 2.2. Create the /backup mount point.

```
[root@serverb ~]# mkdir /backup
```

- 2.3. Before adding the new file system to the /etc/fstab file, retrieve its UUID. The UUID on your system might be different.

```
[root@serverb ~]# lsblk --fs /dev/vdb1
NAME FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE%
MOUNTPOINTS
vdb1 xfs            f74ed805-b1fc-401a-a5ee-140f97c6757d
```

- 2.4. Edit the /etc/fstab file and define the new file system.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=f74ed805-b1fc-401a-a5ee-140f97c6757d  /backup  xfs  defaults  0 0
```

- 2.5. Force the systemd daemon to reread the /etc/fstab file.

```
[root@serverb ~]# systemctl daemon-reload
```

- 2.6. Manually mount the /backup directory to verify your work. Confirm that the mount is successful.

```
[root@serverb ~]# mount /backup
[root@serverb ~]# mount | grep /backup
/dev/vdb1 on /backup type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

3. On the same disk, create two 512 MB GPT partitions called swap1 and swap2. A size between 460 MB and 564 MB is acceptable. Configure the file-system types of the partitions to host swap spaces.

- 3.1. Retrieve the end position of the first partition by displaying the current partition table on the /dev/vdb disk. In the next step, you use that value as the start of the swap1 partition.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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 512 MB swap1 partition. Set its type to `linux-swap`. Use the end position of the first partition as the starting point. The end position is 2000 MB + 512 MB = 2512 MB

```
[root@serverb ~]# parted /dev/vdb mkpart swap1 linux-swap 2000M 2512M
Information: You may need to update /etc/fstab.
```

- 3.3. Create the second 512 MB swap2 partition. Set its type to `linux-swap`. Use the end position of the previous partition as the starting point: 2512M. The end position is 2512 MB + 512 MB = 3024 MB

```
[root@serverb ~]# parted /dev/vdb mkpart swap2 linux-swap 2512M 3024M
Information: You may need to update /etc/fstab.
```

- 3.4. Display the partition table to verify your work.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 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. Run the `udevadm settle` command. The command waits for the system to register the new partitions and to create the device files.

```
[root@serverb ~]# udevadm settle
```

4. Initialize the two 512 MiB partitions as swap spaces and configure them to activate at boot. Set the swap space on the swap2 partition to be preferred over the other.

- 4.1. Use the `mkswap` command to initialize the swap partitions. Note the UUIDs of the two swap spaces, because you will use that information in the next step. If you clear the `mkswap` output, then use the `lsblk --fs` command to retrieve the UUIDs.

```
[root@serverb ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 489 MiB (512749568 bytes)
no label, UUID=87976166-4697-47b7-86d1-73a02f0fc803
[root@serverb ~]# mkswap /dev/vdb3
Setting up swapspace version 1, size = 488 MiB (511700992 bytes)
no label, UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942
```

- 4.2. 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 with the `pri` option.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=a3665c6b-4fbf-49b6-a528-74e268b058dd    /backup xfs    defaults  0 0
UUID=87976166-4697-47b7-86d1-73a02f0fc803    swap      swap  pri=10    0 0
UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942    swap      swap  pri=20    0 0
```

4.3. Force the systemd daemon to reread the /etc/fstab file.

```
[root@serverb ~]# systemctl daemon-reload
```

4.4. Activate the new swap spaces. Verify the correct activation of the swap spaces.

```
[root@serverb ~]# swapon -a
[root@serverb ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2  partition 489M   0B   10
/dev/vdb3  partition 488M   0B   20
```

- To verify your work, reboot the serverb machine. Confirm that the system automatically mounts the first partition onto the /backup directory. Also, confirm that the system activates the two swap spaces.

5.1. Reboot serverb.

```
[root@serverb ~]# systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

5.2. Wait for serverb to boot and then log in as the student user.

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

- Verify that the system automatically mounts the /dev/vdb1 partition onto the /backup directory.

```
[student@serverb ~]$ mount | grep /backup
/dev/vdb1 on /backup type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

5.4. Verify that the system activates both swap spaces.

```
[student@serverb ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2  partition 489M   0B   10
/dev/vdb3  partition 488M   0B   20
```

5.5. 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, 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 storage-review
```

This concludes the section.

# Summary

---

- The `root` user can use the `mount` command to manually mount a file system.
- The `parted` command adds, modifies, and removes partitions on disks with the MBR or the GPT partitioning scheme.
- The `mkfs.xfs` command creates XFS file systems on disk partitions.
- The `/etc/fstab` file contains devices that must be persistently mounted.
- The `mkswap` command initializes swap spaces.



## Chapter 10

# Manage Storage Stack

### Goal

Create and manage logical volumes that contain file systems or swap spaces from the command line.

### Objectives

- Describe logical volume manager components and concepts, and implement LVM storage and display LVM component information.
- Analyze the multiple storage components that make up the layers of the storage stack.

### Sections

- Create and Extend Logical Volumes (and Guided Exercise)
- Manage Layered Storage (and Guided Exercise)

### Lab

Manage Storage Stack

# Create and Extend Logical Volumes

## Objectives

Describe logical volume manager components and concepts, and implement LVM storage and display LVM component information.

## Logical Volume Manager Overview

Use the *Logical Volume Manager (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, RAID arrays, or SAN disks. You must initialize the device as 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)

Volume groups are storage pools made from one or more PVs. It is the functional equivalent of a whole disk compared to physical storage. A PV must only be allocated 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 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 gets 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 used for creating 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.

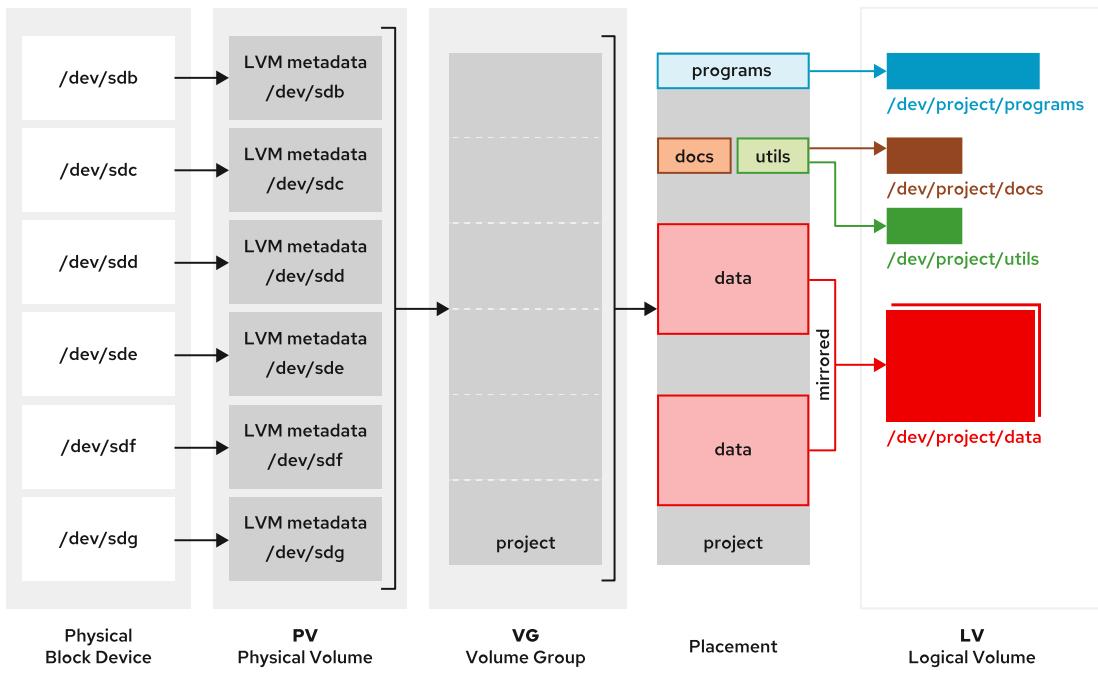


Figure 10.1: Logical Volume Manager workflow

**Note**

The examples here use a /dev/vdb device name and its storage partitions. The device names on your classroom system may 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

Partitioning is optional when already present. Use the `parted` command to create a new partition on the physical device. Set the physical device to the Linux LVM partition type. Use the `udevadm settle` command to register the new partition with the kernel.

```
[root@host ~]# parted /dev/vdb mklabel gpt mkpart primary 1MiB 769MiB
...output omitted...
[root@host ~]# parted /dev/vdb mkpart primary 770MiB 1026MiB
[root@host ~]# parted /dev/vdb set 1 lvm on
[root@host ~]# parted /dev/vdb set 2 lvm on
[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/vdb1 and /dev/vdb2 devices as PVs that are ready for creating volume groups.

```
[root@host ~]# pvcreate /dev/vdb1 /dev/vdb2
Physical volume "/dev/vdb1" successfully created.
Physical volume "/dev/vdb2" 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 using the /dev/vdb1 and /dev/vdb2 PVs.

```
[root@host ~]# vgcreate vg01 /dev/vdb1 /dev/vdb2
Volume group "vg01" successfully created
```

## Create a Logical Volume

The lvcreate command creates a new 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 that will contain this logical volume. This example creates lv01 LV with 700 MiB in size 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 sufficient free physical extents. The LV size rounds up to the next value of PE size when the size does not exactly match.

The lvcreate command -L option requires sizes in bytes, mebibytes (binary megabytes, 1048576 bytes), and gibibytes (binary gigabytes), or similar. The lower case -l requires sizes 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 is 128 MiB.

## Create a Logical Volume with Deduplication and Compression

RHEL 9 uses an LVM VDO implementation for managing VDO volumes. The previous python-based VDO management tools are still available but are no longer needed.

The *Virtual Data Optimizer* (VDO) provides in-line block-level deduplication, compression, and thin provisioning for storage. Configure a VDO volume to use up to 256 TB of physical storage. Manage VDO as a type of LVM logical volume (LVs), similar to LVM thinly provisioned volumes. An LVM VDO is composed of two logical volumes:

### VDO pool LV

This LV stores, deduplicates, and compresses data and sets the size of the VDO volume backed by the physical device. VDO is deduplicated and compresses each VDO LV separately because each VDO pool LV can hold only one VDO LV.

### VDO LV

A virtual device is provisioned on top of the VDO pool LV and sets the logical size of the VDO volume storing the data before deduplication and compression occur.

LVM VDO presents the deduplicated storage as a regular logical volume (LV). The VDO volume can be formatted with a standard file systems, shared as a block device, or used to build other storage layers, the same as any normal logical volume.

To be able to use VDO deduplication and compression, install the vdo and kmod-kvdo packages.

```
[root@host ~]# dnf install vdo kmod-kvdo
```

Verify that the selected LVM volume group has sufficient free storage capacity. Use the lvcreate command with the --type vdo parameter to create a VDO LV.

```
[root@host ~]# lvcreate --type vdo --name vdo-lv01 --size 5G vg01
Logical blocks defaulted to 523108 blocks.
The VDO volume can address 2 GB in 1 data slab.
It can grow to address at most 16 TB of physical storage in 8192 slabs.
If a larger maximum size might be needed, use bigger slabs.
Logical volume "vdo-lv01" created.
```

## 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 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
```

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 looking for the UUID. This behavior successful even when the VG was created using a name, because the PV will always contain a UUID.

## Display LVM Component Status

LVM provides various utilities to display the status information of PV, VG, and LV. Use the `pvdisplay`, `vgdisplay`, and `lvdisplay` commands to show the status information of 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 of all PVs present on the system. Providing the name of the PV as the argument to the command shows the information specific to the PV.

```
[root@host ~]# pvdisplay /dev/vdb1
--- Physical volume ---
PV Name      /dev/vdb1          ①
VG Name      vg01              ②
PV Size      731.98 MiB / not usable 3.98 MiB ③
Allocatable   yes
PE Size      4.00 MiB          ④
Total PE     182
Free PE      107              ⑤
Allocated PE 75
PV UUID      zP0gD9-NxTV-Qtoi-yfQD-TGpL-0Yj0-wExh2N
```

- 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 PE size available in the VG to create new LVs or 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 specific to the VG.

```
[root@host ~]# vgdisplay vg01
--- Volume group ---
VG Name      vg01          ①
System ID    lvm2
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	1012.00 MiB	❷
PE Size	4.00 MiB	
Total PE	253	❸
Alloc PE / Size	75 / 300.00 MiB	
Free PE / Size	178 / 712.00 MiB	❹
VG UUID	jK5M1M-Yv1k-kxU2-bxmS-dNjQ-Bs3L-DRlJNc	

- ❶ VG Name shows the name of the volume group.
- ❷ VG Size displays the total size of the storage pool available for LV allocation.
- ❸ Total PE shows the total size of PE units.
- ❹ Free PE / Size shows the available space in the VG to create new LVs or extend existing LVs.

## Display Logical Volume Information

The `lvdisplay` command displays information about logical volumes. Use the command without arguments to list information of all LVs. Providing the name of the LV as an argument displays the information specific to the LV.

[root@host ~]# <b>lvdisplay</b> /dev/vg01/lv01	
--- Logical volume ---	
LV Path	/dev/vg01/lv01
LV Name	lv01
VG Name	vg01
LV UUID	FVmNel-u25R-dt3p-C5L6-VP2w-QRNP-scqrbq
LV Write Access	read/write
LV Creation host, time	servera.lab.example.com, 2022-04-07 10:45:34 -0400
LV Status	available
# open	1
LV Size	300.00 MiB
Current LE	75
Segments	1
Allocation	inherit
Read ahead sectors	auto
- currently set to	8192
Block device	253:0

- ❶ LV Path shows the device name of the LV.
- ❷ VG Name shows the VG used for creating this LV.
- ❸ LV Size shows the total size of the LV. Use the file-system tools to determine the free and used space for the LV.
- ❹ Current LE shows the number of logical extents used by this LV.

## Extend and Reduce LVM Storage

After creating a logical volume, you can extend the volume to expand the file system. You may require extending PV or VG to increase the storage capacity of the LV.

### Extend a Volume Group Size

You might need to add more disk space to extend a VG. You can add additional physical volumes to a VG to extend its available size.

Prepare the physical device and create the physical volume when not present.

```
[root@host ~]# parted /dev/vdb mkpart primary 1072MiB 1648MiB
...output omitted...
[root@host ~]# parted /dev/vdb set 3 lvm on
...output omitted...
[root@host ~]# udevadm settle
[root@host ~]# pvcreate /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
```

The `vgextend` command adds the new PV to the VG. Provide the VG and PV names as arguments to the `vgextend` command.

```
[root@host ~]# vgextend vg01 /dev/vdb3
Volume group "vg01" successfully extended
```

This command extends the `vg01` VG by the `/dev/vdb3` PV size.

### Extend a Logical Volume Size

A benefit of using logical volumes is increasing their size without experiencing any downtime. Add free physical extents to the LV in the VG to extend its capacity to expand the LV's file system. Use the `vgdisplay` command to confirm that the volume group has sufficient 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 300.00 MiB (75 extents) to 800.00
MiB (200 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 you want to add this value to the existing size; otherwise, without the plus sign, the value defines the final size of the LV.

The `lvextend` command uses different methods to specify the size. The `lvextend` command `-l` option expects the number of PE as the argument. The `lvextend` command `-L` option expects sizes in bytes, mebibytes, gibibytes, and similar.

## Extend an XFS File System to the Logical Volume Size

The `xfs_growfs` command helps 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 while resizing.

```
[root@host ~]# xfs_growfs /mnt/data/
...output omitted...
data blocks changed from 76800 to 204800
```



### Important

Always run the `xfs_growfs` command after executing the `lvextend` command. Use the `lvextend` command `-r` option to run the two steps consecutively. After extending the LV, it resizes the file system by using the `fsadm` command. This option supports several other file systems.

## Extend an EXT4 File System to the Logical Volume Size

The steps for extending LV using the ext4 file system are the same for LV 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 while resizing.

```
[root@host ~]# resize2fs /dev/vg01/lv01
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/vg01/lv01 to 256000 (4k) blocks.
The filesystem on /dev/vg01/lv01 is now 256000 (4k) blocks long.
```

The primary difference between `xfs_growfs` and `resize2fs` is the argument passed to identify the file system. The `xfs_growfs` command takes the mount point as an argument, and the `resize2fs` command takes the LV name as an argument. The `xfs_growfs` command only supports an online resize whereas the `resize2fs` command supports both online and offline resizing. You can resize an ext4 file system up or down, but you can only resize an XFS file system up.

## Extend Swap Space Logical Volumes

You can extend the LVs used as swap space; however, the process differs from expanding the ext4 or XFS file system. Logical volumes 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 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=25b4d602-6180-4b1c-974e-7f40634ad660
```

Use the `swapon` command to activate the swap space on the LV.

```
[root@host ~]# swapon /dev/vg01/swap
```

## Reduce Volume Group Storage

Reducing a VG involves removing unused PV 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 may continue to use the LV from the same VG while reducing. Use the `pvmove` command `-A` option to automatically backup the metadata of the VG after a change. This option uses the `vgcfgbackup` command to backup metadata automatically.

```
[root@host ~]# pvmove /dev/vdb3
```



### Warning

Before using the `pvmove` command, back up the data 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/vdb3
  Removed "/dev/vdb3" from volume group "vg01"
```



### Important

The GFS2 and XFS file systems do not support shrinking, so you cannot reduce the size of an LV.

## Remove LVM Storage

Use the `lvremove`, `vgremove`, and `pvremove` commands to remove an LVM component that is no longer required.

## Prepare the File System

Move all data that must be kept to another file system. Use the `umount` command to unmount the file system and then remove any `/etc/fstab` entries associated with this file system.

```
[root@host ~]# umount /mnt/data
```



### Warning

Removing a logical volume destroys any data stored on the logical volume. Back up or move your data **before** you remove the logical volume.

## Remove the Logical Volume

Use the `lvremove DEVICE-NAME` 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 made available for assignment to existing or new LVs in the volume group.

## Remove the Volume Group

Use the `vgremove VG-NAME` 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 made available for assignment 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 to remove more than one at a time. This command deletes the PV metadata from the partition (or disk). The partition is now free for reallocation or reformatting.

```
[root@host ~]# pvremove /dev/vdb1 /dev/vdb2
Labels on physical volume "/dev/vdb1" successfully wiped.
Labels on physical volume "/dev/vdb2" successfully wiped.
```



## References

fdisk(8), gdisk(8), parted(8), partprobe(8), lvm(8), pvcreate(8), vgcreate(8), lvcreate(8), mkfs(8), pvdisk(8), vgdisk(8), lvdisk(8), vgextend(8), lvextend(8), xfs\_growfs(8), resize2fs(8), swapoff(8), mkswap(8), swapon(8), pvmount(8), vgcfgbackup(8), vgreduce(8), lvremove(8), vgremove(8), pvremove(8) man pages

For further information, refer to *Configuring and Managing Logical Volumes* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_and\\_managing\\_logical\\_volumes/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_logical_volumes/index)

## ► Guided Exercise

# Create and Extend Logical Volumes

In this lab, you create and extend a physical volume, volume group, logical volume, and an XFS file system. You also persistently mount the logical volume file system.

## Outcomes

- Create physical volumes, volume groups, and logical volumes with LVM tools.
- Create new file systems on logical volumes and persistently mount them.
- Extend the volume group to include an additional physical volume.
- Resize the logical volume while the file system is still mounted and in use.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start lvm-manage
```

## 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. Create the physical device partition on the `/dev/vdb` storage device.

- 2.1. Create two partitions of 256 MiB each in size and set to the Linux LVM type. Use `first` and `second` as the names for these partitions.

```
[root@servera ~]# parted /dev/vdb mklabel gpt
[root@servera ~]# parted /dev/vdb mkpart first 1MiB 258MiB
[root@servera ~]# parted /dev/vdb set 1 lvm on
[root@servera ~]# parted /dev/vdb mkpart second 258MiB 514MiB
[root@servera ~]# parted /dev/vdb set 2 lvm on
```

- 2.2. Register the new partitions with the kernel.

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

- 2.3. List the partitions on the /dev/vdb storage device. In the Number column, the values 1 and 2 correspond to the /dev/vdb1 and /dev/vdb2 device partitions. The Flags column indicates the partition type.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  271MB  269MB   first      lvm
 2      271MB   539MB  268MB   second     lvm
```

- 3. Label the two new partitions as physical volumes.

```
[root@servera ~]# pvcreate /dev/vdb1 /dev/vdb2
Physical volume "/dev/vdb1" successfully created.
Physical volume "/dev/vdb2" successfully created.
Creating devices file /etc/lvm/devices/system.devices
```

- 4. Create the servera\_group volume group by using the two new PVs.

```
[root@servera ~]# vgcreate servera_group /dev/vdb1 /dev/vdb2
Volume group "servera_group" successfully created
```

- 5. Create the servera\_volume logical volume with a size of 400 MiB. This command creates the /dev/servera\_group/servera\_volume LV without a file system.

```
[root@servera ~]# lvcreate -n servera_volume -L 400M servera_group
Logical volume "servera_volume" created.
```

- 6. Format the newly created LV and mount it persistently.

- 6.1. Format the servera\_volume LV with the XFS file system.

```
[root@servera ~]# mkfs -t xfs /dev/servera_group/servera_volume
...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/servera_group/servera_volume /data xfs defaults 0 0
```

- 6.4. Mount the servera\_volume LV.

```
[root@servera ~]# mount /data
```

- 7. Verify that the mounted file system is accessible and display the status information of the LVM.

- 7.1. Verify that you can copy files to the /data directory.

```
[root@servera ~]# cp -a /etc/*.* /data
[root@servera ~]# ls /data | wc -l
32
```

- 7.2. View the PV status information. The output shows that the PV uses the servera\_group VG. The PV has a size of 256 MiB and the physical extent size of 4 MiB.

There are 63 PEs, with 27 PEs available for allocation and 36 PEs currently allocated to LVs. When calculating the size in MiBs:

- Total 252 MiB (63 PEs x 4 MiB).
- Free 108 MiB (27 PEs x 4 MiB).
- Allocated 144 MiB (36 PEs x 4 MiB).

```
[root@servera ~]# pvdisplay /dev/vdb2
--- Physical volume ---
PV Name           /dev/vdb2
VG Name           servera_group
PV Size          256.00 MiB / not usable 4.00 MiB
Allocatable       yes
PE Size          4.00 MiB
Total PE         63
Free PE          27
Allocated PE     36
PV UUID          FKKFYJ-wJiR-1jt2-sfy3-yjPy-TyLN-LG92jj
```

- 7.3. View the VG status information of the servera\_group VG. The output shows the VG size of 508 MiB with PE size of 4 MiB. The available size from the VG is 108 MiB.

```
[root@servera ~]# vgdisplay servera_group
--- Volume group ---
VG Name           servera_group
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	508.00 MiB
PE Size	4.00 MiB
Total PE	127
Alloc PE / Size	100 / 400.00 MiB
Free PE / Size	27 / 108.00 MiB
VG UUID	g0ahyT-90J5-iGic-nnb5-G6T9-tLdK-dX8c9M

- 7.4. View the status information for the `servera_volume` LV. The output shows the VG name used for creating the LV. It also shows the LV size of 400 MiB and LE size of 100.

```
[root@servera ~]# lvdisplay /dev/servera_group/servera_volume
--- Logical volume ---
LV Path          /dev/servera_group/servera_volume
LV Name          servera_volume
VG Name          servera_group
LV UUID          93MfUt-esgT-B5HM-r1p5-DVZH-n5cn-J5e2tw
LV Write Access  read/write
LV Creation host, time servera.lab.example.com, 2022-04-11 03:25:12 -0400
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
```

- 7.5. View the free disk space in human-readable units. The output shows the total size of 395 MiB with the available size of 372 MiB.

```
[root@servera ~]# df -h /data
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/servera_group-servera_volume  395M   24M  372M   6% /data
```

## ► 8. Create the physical resource on the /dev/vdb storage device.

- 8.1. Create an additional partition of 512 MiB and set it to type Linux LVM. Use `third` as the name for this partition.

```
[root@servera ~]# parted /dev/vdb mkpart third 514MiB 1026MiB
[root@servera ~]# parted /dev/vdb set 3 lvm on
```

- 8.2. Register the new partition with the kernel.

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

- 8.3. Add the new partition as a PV.

```
[root@servera ~]# pvcreate /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
```

- ▶ 9. Using the newly created disk space, extend the file system on the `servera_volume` to be a total size of 700 MiB.

- 9.1. Extend the `servera_group` VG using the new `/dev/vdb3` PV.

```
[root@servera ~]# vgextend servera_group /dev/vdb3
Volume group "servera_group" successfully extended
```

- 9.2. Extend the existing `servera_volume` LV to 700 MiB.

```
[root@servera ~]# lvextend -L 700M /dev/servera_group/servera_volume
Size of logical volume servera_group/servera_volume changed from 400.00 MiB (100
extents) to 700.00 MiB (175 extents).
Logical volume servera_group/servera_volume successfully resized.
```

- 9.3. Extend the XFS file system using the free space on the LV.

```
[root@servera ~]# xfs_growfs /data
...output omitted...
data blocks changed from 102400 to 179200
```

- ▶ 10. Verify that the LV size has been extended and the contents are still present in the volume.

- 10.1. Verify the size of the extended LV by using the `lvdisplay` command.

```
[root@servera ~]# lvdisplay /dev/servera_group/servera_volume
--- Logical volume ---
LV Path          /dev/servera_group/servera_volume
LV Name          servera_volume
VG Name          servera_group
LV UUID          mLQhsD-hyL0-KC2B-2nug-o2Nc-0znS-Q428fK
LV Write Access  read/write
LV Creation host, time servera.lab.example.com, 2022-04-12 06:04:12 -0400
LV Status        available
# open           1
LV Size          700.00 MiB
Current LE       175
Segments         3
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:0
```

- 10.2. Verify the new file-system size. Verify that the previously copied files are still present.

```
[root@servera ~]# df -h /data
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/servera_group-servera_volume  695M   26M  670M   4% /data
[root@servera ~]# ls /data | wc -l
32
```

- 11. 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, 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 lvm-manage
```

This concludes the section.

# Manage Layered Storage

---

## Objectives

Analyze the multiple storage components that make up the layers of the storage stack.

## Storage Stack

Storage in RHEL is comprised of multiple layers of drivers, managers, and utilities that are mature, stable, and full of modern features. Managing storage requires familiarity with stack components, and recognizing that storage configuration affects the boot process, application performance, and the ability to provide needed storage features for specific application use cases.

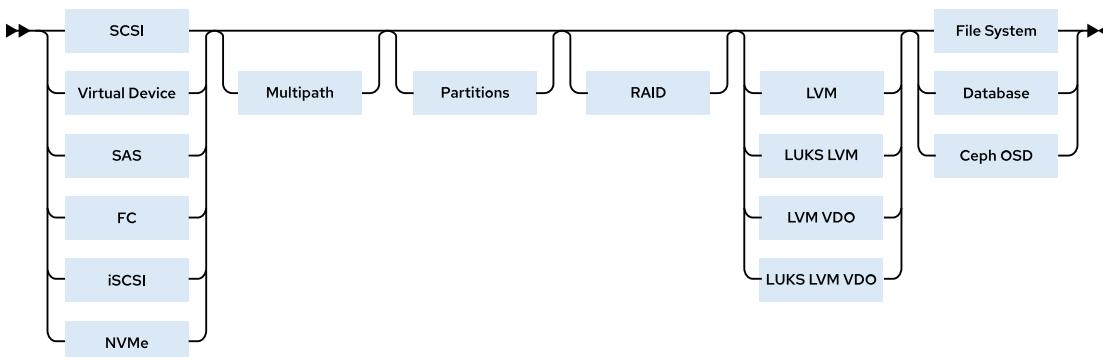


Figure 10.2: Storage Stack

Previous sections in Red Hat System Administration courses have presented XFS file systems, network storage sharing, partitioning, and the Logical Volume Manager. This section overviews the bottom-to-top RHEL storage stack and introduces each layer.

This section also covers Stratis, the daemon that unifies, configures and monitors the underlying RHEL storage stack components, and provides automated local storage management from either the CLI or from the RHEL web console.

## Block Device

Block devices are at the bottom of the storage stack and present a stable, consistent device protocol that allows virtually any block device to be transparently included in a RHEL storage configuration. Most block devices today are accessed through the RHEL SCSI device driver, and appear as a SCSI device, including legacy ATA hard drives, solid-state devices, and common enterprise host bus adapters (HBAs). RHEL also supports iSCSI, Fibre Channel over Ethernet (FCoE), virtual machine driver (`virtio`), serial-attached storage (SAS), Non-Volatile Memory Express (NVMe) and others.

An iSCSI target can be a dedicated physical device in a network or an iSCSI software-configured logical device on a networked storage server. The target is the portal endpoint in a SCSI protocol bus communication, to access the storage as Logical Unit Numbers (LUNs).

Fibre Channel over Ethernet (FCoE) protocol transmits Fibre Channel frames over Ethernet networks. Typically, data centers have dedicated LAN and Storage Area Network (SAN) cabling, each uniquely configured for their traffic. With FCoE, both traffic types can be combined into

a larger, converged, Ethernet network architecture. FCoE benefits include lower hardware and energy costs.

## Multipath

A path is a connection between a server and the underlying storage. Device Mapper multipath (`dm-multipath`) is a RHEL native multipath tool for configuring redundant I/O paths into a single, path-aggregated logical device. A logical device created by using the device mapper (`dm`) appears as a unique block device under `/dev/mapper/` for each LUN attached to the system.

Storage multipath redundancy can also be implemented by using network bonding when the storage, such as iSCSI and FCoE, uses network cabling.

## Partitions

A block device can be further divided into partitions. Partitions may consume the entire block device size or divide the block device for creating multiple partitions. These partitions can be used to create a file system, LVM devices, or can be used directly for database structures or other raw storage.

## RAID

A Redundant Array of Inexpensive Disks (RAID) is a storage virtualization technology that creates large logical volumes from multiple physical or virtual block device components. Different forms of RAID volumes offer data redundancy, performance improvement, or both, by implementing mirroring or striping layouts.

LVM supports RAID levels 0, 1, 4, 5, 6, and 10. RAID logical volumes created and managed by LVM leverage the Multiple Devices (`mdadm`) kernel drivers. When not using LVM, Device Mapper RAID (`dm-raid`) provides a device mapper interface to the `mdadm` kernel drivers.

## Logical Volume Manager

LVM physical volumes, volume groups and logical volumes were discussed in a previous section. LVM can take almost any form of physical or virtual block devices, and build storage as new logical storage volumes, effectively hiding the physical storage configuration from applications and other storage clients.

You can stack LVM volumes and implement advanced features such as encryption and compression for each part of the stack. There are mandated rules and recommended practices to follow for practical layering for specific scenarios, but this introduction focuses only on introducing the components. Use-case-specific recommendations are found in the *Configuring and Managing Logical Volumes* user guide.

LVM supports *LUKS encryption* where a lower block device or partition is encrypted and presented as a secure volume to create a file system on top. The practical advantage for LUKS over file system-based or file-based encryption is that a LUKS-encrypted device does not allow public visibility or access to the file-system structure, such that a physical device remains secure even when removed from a computer.

LVM now incorporates *VDO deduplication and compression* as a configurable feature of regular logical volumes. LUKS encryption and VDO can be used together with logical volumes, with the LVM LUKS encryption enabled underneath the LVM VDO volume.

## File System or Other Use

The top layer of the stack is typically a file system, but can be used as raw space for databases or custom application data requirements. RHEL supports multiple file-system types, but recommends XFS for most modern use cases. XFS is required when the utility implementing LVM is Red Hat Ceph Storage or the Stratis storage tool.

Database server applications consume storage in different ways, depending on their architecture and size. Some smaller database store their structures in regular files that are contained in a file system. Because of the additional overhead or restrictions of file system access, this architecture has scaling limits. Larger databases that want to bypass file system caching, and use their own caching mechanisms, prefer to create their database structures on raw storage. Logical volumes are suitable for use for database and other raw storage use cases.

Red Hat Ceph Storage also prefers to create its own storage management metadata structures on raw devices to be used to create Ceph Object Storage Devices (OSDs). In the latest Red Hat Ceph Storage versions, Ceph uses LVM to initialize disk devices for use as OSDs. More information is available in the *Cloud Storage with Red Hat Ceph Storage* (CL260) course.

## Stratis Storage Management

*Stratis* is a local storage management tool developed by Red Hat and the upstream Fedora community. Stratis makes it easier to perform an initial storage configuration, change a storage configuration, and use advanced storage features.



### Important

Stratis is currently available as a Technology Preview, but is expected to be supported in a later RHEL 9 version. For information on Red Hat scope of support for Technology Preview features, see the Technology Features Support Scope [<https://access.redhat.com/support/offering/techpreview>] document.

Red Hat encourages customers to provide feedback when deploying Stratis.

Stratis runs as a service that manages pools of physical storage devices and transparently creates and manages volumes for the newly created file systems.

Stratis builds file systems from shared pools of disk devices by using a concept known as *thin provisioning*. Instead of immediately allocating physical storage space to the file system when you create it, Stratis dynamically allocates that space from the pool as the file system stores more data. Therefore, the file system might appear to be 1 TiB in size, but might only have 100 GiB of real storage actually allocated to it from the pool.

You can create multiple pools from different storage devices. From each pool, you can create one or more file systems. Currently, you can create up to  $2^{24}$  file systems per pool.

Stratis builds the components that make up a Stratis-managed file system from standard Linux components. Internally, Stratis uses the Device Mapper infrastructure that LVM also uses. Stratis formats the managed file systems with XFS.

*Figure 10.3* illustrates how Stratis assembles the elements of its storage management solution. Stratis assigns block storage devices such as hard disks or SSDs to pools, each contributing some physical storage to the pool. Then, it creates file systems from the pools, and maps physical storage to each file system as needed.

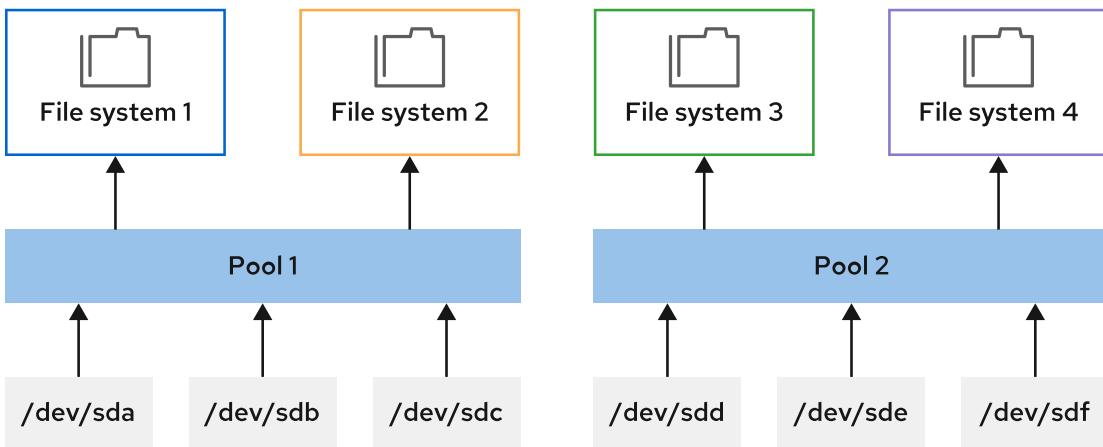


Figure 10.3: Stratis Architecture

## Stratis Administration Methods

To manage file systems with the Stratis storage management solution, install the `stratis-cli` and `stratisd` packages. The `stratis-cli` package provides the `stratis` command, which sends reconfiguration requests to the `stratisd` system daemon. The `stratisd` package provides the `stratisd` service, which handles reconfiguration requests and manages and monitors Stratis's block devices, pools, and file systems.

Stratis administration is included in the RHEL web console.



### Warning

Reconfigure file systems created by Stratis only with Stratis tools and commands.

Stratis uses stored metadata to recognize managed pools, volumes, and file systems. Manually configuring Stratis file systems with non-Stratis commands can result in overwriting that metadata and prevent Stratis from recognizing the file system volumes it had previously created.

## Install and Enable Stratis

To use Stratis, ensure that your system has the required software and that the `stratisd` service is running. Install the `stratis-cli` and `stratisd` packages, and start and enable the `stratisd` service.

```
[root@host ~]# dnf install stratis-cli stratisd
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
[root@host ~]# systemctl enable --now stratisd
```

## Create Stratis Pools

Create pools of one or more block devices by using the `stratis pool create` command. Then, use the `stratis pool list` command to view the list of available pools.

```
[root@host ~]# stratis pool create pool1 /dev/vdb
[root@host ~]# stratis pool list
Name          Total Physical  Properties          UUID
pool1    5 GiB / 37.63 MiB / 4.96 GiB   ~Ca,-Cr   11f6f3c5-5...
```

**Warning**

The `stratis pool list` command displays the actual storage space that is in use and the pool space that is still available. Currently, if a pool becomes full, then further data written to the pool's file systems is quietly discarded.

Use the `stratis pool add-data` command to add additional block devices to a pool. Then, use the `stratis blockdev list` command to verify the block devices of a pool.

```
[root@host ~]# stratis pool add-data pool1 /dev/vdc
[root@host ~]# stratis blockdev list pool1
Pool Name  Device Node  Physical Size  Tier
pool1      /dev/vdb      5 GiB       Data
pool1      /dev/vdc      5 GiB       Data
```

## Manage Stratis File Systems

Use the `stratis filesystem create` command to create a file system from a pool. The links to the Stratis file systems are in the `/dev/stratis/pool1` directory. Use the `stratis filesystem list` command to view the list of available file systems.

```
[root@host ~]# stratis filesystem create pool1 fs1
[root@host ~]# stratis filesystem list
Pool Name  Name  Used     Created           Device          UUID
pool1      fs1   546 MiB  Apr 08 2022 04:05  /dev/stratis/pool1/fs1
c7b5719...
```

Create a Stratis file system snapshot by using the `stratis filesystem snapshot` command. Snapshots are independent of the source file systems. Stratis dynamically allocates the snapshot storage space and uses an initial 560 MB to store the file system's journal.

```
[root@host ~]# stratis filesystem snapshot pool1 fs1 snapshot1
```

## Persistently Mount Stratis File Systems

You can persistently mount Stratis file systems by editing the `/etc/fstab` file and specifying the details of the file system. Use the `lsblk` command to display the UUID of the file system that you should use in the `/etc/fstab` file to identify the file system. You can also use the `stratis filesystem list` command to obtain the UUID of the file system.

```
[root@host ~]# lsblk --output=UUID /dev/stratis/pool1/fs1
UUID
c7b57190-8fba-463e-8ec8-29c80703d45e
```

The following is an example entry in the `/etc/fstab` file to mount a Stratis file system persistently. This example entry is a single long line in the file. The `x-systemd.requires=stratisd.service` mount option delays mounting the file system until the `systemd` daemon starts the `stratisd` service during the boot process.

```
UUID=c7b57190-8fba-463e-8ec8-29c80703d45e /dir1 xfs defaults,x-
systemd.requires=stratisd.service 0 0
```



### Important

If you do not include the `x-systemd.requires=stratisd.service` mount option in the `/etc/fstab` file for each Stratis file system, then the machine fails to start properly and aborts to `emergency.target` the next time you reboot it.



### Warning

Do not use the `df` command to query Stratis file system space.

The `df` command reports that any mounted Stratis-managed XFS file system is 1 TiB in size, regardless of the current allocation. Because the file system is thinly provisioned, a pool might not have enough physical storage to back the entire file system. Other file systems in the pool could use up all the available storage.

Therefore, it is possible to consume the whole storage pool, even as the `df` command reports that the file system has available space. Writes to a file system with no available pool storage can fail.

Instead, always use the `stratis pool list` command to accurately monitor a pool's available storage.



### References

For further information, refer to *Deduplicating And Compressing Logical Volumes On RHEL* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/deduplicating\\_and\\_compressing\\_logical\\_volumes\\_on\\_rhel/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/deduplicating_and_compressing_logical_volumes_on_rhel/index)

For further information, refer to *Red Hat Enterprise Linux 9 Managing File Systems Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/managing\\_file\\_systems](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_file_systems)

### Stratis Storage

<https://stratis-storage.github.io/>

### What Stratis learned from ZFS, Btrfs, and Linux Volume Manager

<https://opensource.com/article/18/4/stratis-lessons-learned>

## ► Guided Exercise

# Manage Layered Storage

In this exercise, you use Stratis to create file systems from pools of storage provided by physical storage devices.

### Outcomes

- Create a thin-provisioned file system by using Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thin-provisioned file system.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start lvm-stratis
```

### 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. Install the `stratisd` and `stratis-cli` packages.

```
[root@servera ~]# dnf install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 3. Activate the `stratisd` service.

```
[root@servera ~]# systemctl enable --now stratisd
```

- 4. Ensure that the `stratispool1` Stratis pool exists on the `/dev/vdb` block device.

- 4.1. Create the `stratispool1` Stratis pool.

```
[root@servera ~]# stratis pool create stratispool1 /dev/vdb
```

4.2. Verify the availability of the `stratispool1` pool. Note the size of the pool.

```
[root@servera ~]# stratis pool list
Name           Total Physical  Properties          UUID
stratispool1   5 GiB / 37.63 MiB / 4.96 GiB    ~Ca,~Cr   3557c389-7...
```

► 5. Expand the capacity of the `stratispool1` pool by adding the `/dev/vdc` block device.

5.1. Add the `/dev/vdc` block device to the `stratispool1` pool.

```
[root@servera ~]# stratis pool add-data stratispool1 /dev/vdc
```

5.2. Verify the size of the `stratispool1` pool. The `stratispool1` pool size increases when you add the block device.

```
[root@servera ~]# stratis pool list
Name           Total Physical  Properties          UUID
stratispool1   10 GiB / 41.63 MiB / 9.96 GiB   ~Ca,~Cr   3557c389-7...
```

5.3. Verify the block devices that are currently members of the `stratispool1` pool.

```
[root@servera ~]# stratis blockdev list stratispool1
Pool Name     Device Node  Physical Size  Tier
stratispool1  /dev/vdb      5 GiB       Data
stratispool1  /dev/vdc      5 GiB       Data
```

► 6. Add a thin-provisioned file system called `stratis-filesystem1` in the `stratispool1` pool. Mount the file system on the `/stratisvol` directory. Create a file on the `stratis-filesystem1` file system called `file1` containing the text `Hello World!`. Modify the `/etc/fstab` file to persistently mount the file system on the `/stratisvol` directory. Reboot your system and verify that the file system is persistently mounted across reboots.

6.1. Create the thin-provisioned file system `stratis-filesystem1` on the `stratispool1` pool. It might take up to a minute for the command to complete.

```
[root@servera ~]# stratis filesystem create stratispool1 stratis-filesystem1
```

6.2. Verify the availability of the `stratis-filesystem1` file system. Note the current usage of the `stratis-filesystem1` file system. The usage of the file system will increase on-demand in the later steps.

```
[root@servera ~]# stratis filesystem list
Pool Name     Name           Used        Created          Device
                                         UUID
stratispool1  stratis-filesystem1  546 MiB   Apr 08 2022 07:12  /dev/stratis/
stratispool1/stratis-filesystem1   48e8...
```

6.3. Create the `/stratisvol` directory.

```
[root@servera ~]# mkdir /stratisvol
```

- 6.4. Mount the `stratis-filesystem1` file system on the `/stratisvol` directory.

```
[root@servera ~]# mount /dev/stratis/stratispool1/stratis-filesystem1 \
/stratisvol
```

- 6.5. Verify that the `stratis-filesystem1` volume is mounted on the `/stratisvol` directory.

```
[root@servera ~]# mount
...output omitted...
/dev/mapper/stratis-1-3557...fb3-thin-fs-48e8...9ebe on /stratisvol type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,sunit=2048,swidth=2048,
noquota)
```

- 6.6. Create the `/stratisvol/file1` text file.

```
[root@servera ~]# echo "Hello World!" > /stratisvol/file1
```

- 6.7. Obtain the UUID of the file system. Note that the UUID would be different in your system.

```
[root@servera ~]# lsblk --output=UUID \
/dev/stratis/stratispool1/stratis-filesystem1
UUID
d18cb4fc-753c-473a-9ead-d6661533b475
```

- 6.8. Modify the `/etc/fstab` file to persistently mount the file system on the `/stratisvol` directory. To do so, use the `vim /etc/fstab` command and add the following line. Replace the UUID by the correct one for your system.

```
UUID=d18c... /stratisvol xfs defaults,x-systemd.requires=stratisd.service 0 0
```

- 6.9. Reboot your system and verify that the file system is persistently mounted across reboots.

```
[root@servera ~]# systemctl reboot
...output omitted...
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# mount
...output omitted...
/dev/mapper/stratis-1-3557...fb3-thin-fs-d18c...b475 on /stratisvol type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,sunit=2048,swidth=2048,
noquota,x-systemd.requires=stratisd.service)
```

- 7. Verify that the `stratis-filesystem1` thin-provisioned file system dynamically grows as the data on the file system grows.

7.1. View the current usage of the `stratis-filesystem1` file system.

```
[root@servera ~]# stratis filesystem list
Pool Name      Name          Used       Created        Device
                  UUID
stratispool1   stratis-filesystem1  546 MiB  Apr 08 2022 07:12  /dev/stratis/
stratispool1/stratis-filesystem1  48e8...
```

7.2. Create a 2 GiB file on the `stratis-filesystem1` file system. It might take up to a minute for the command to complete.

```
[root@servera ~]# dd if=/dev/urandom of=/stratisvol/file2 bs=1M count=2048
```

7.3. Verify the space used in the `stratis-filesystem1` file system.

The output shows that the used space in the `stratis-filesystem1` file system has increased. The used-space increase confirms that the thin-provisioned file system dynamically expands as needed.

```
[root@servera ~]# stratis filesystem list
Pool Name      Name          Used       Created        Device
                  UUID
stratispool1   stratis-filesystem1  2.60 GiB  Apr 08 2022 07:12  /dev/stratis/
stratispool1/stratis-filesystem1  48e8...
```

- 8. Create a snapshot of the `stratis-filesystem1` file system called `stratis-filesystem1-snap`. The snapshot provides you with access to any file that you delete from the `stratis-filesystem1` file system.

8.1. Create a snapshot of the `stratis-filesystem1` file system. It might take up to a minute for the command to complete.

```
[root@servera ~]# stratis filesystem snapshot stratispool1 \
stratis-filesystem1 stratis-filesystem1-snap
```

8.2. Verify the availability of the snapshot.

```
[root@servera ~]# stratis filesystem list
Pool Name      Name          Used       Created        Device
                  UUID
stratispool1   stratis-filesystem1-snap  2.73 GiB  Apr 08 2022 07:22  /dev/
stratis/stratispool1/stratis-filesystem1-snap  5774...
stratispool1   stratis-filesystem1       2.73 GiB  Apr 08 2022 07:12  /dev/
stratis/stratispool1/stratis-filesystem1       48e8...
```

8.3. Remove the `/stratisvol/file1` file.

```
[root@servera ~]# rm /stratisvol/file1
rm: remove regular file '/stratisvol/file1'? y
```

- 8.4. Create the /stratisvol-snap directory.

```
[root@servera ~]# mkdir /stratisvol-snap
```

- 8.5. Mount the stratis-filesystem1-snap snapshot on the /stratisvol-snap directory.

```
[root@servera ~]# mount /dev/stratis/stratispool1/stratis-filesystem1-snap \
/stratisvol-snap
```

- 8.6. Verify that you can still access the file you deleted from the stratis-filesystem1 file system in the snapshot.

```
[root@servera ~]# cat /stratisvol-snap/file1
Hello World!
```

▶ 9. Unmount the /stratisvol and /stratisvol-snap volumes.

```
[root@servera ~]# umount /stratisvol-snap
[root@servera ~]# umount /stratisvol
```

▶ 10. Remove the stratis-filesystem1 thin-provisioned file system and the stratis-filesystem1-snap snapshot from the system.

- 10.1. Destroy the stratis-filesystem1-snap snapshot.

```
[root@servera ~]# stratis filesystem destroy stratispool1 stratis-filesystem1-snap
```

- 10.2. Destroy the stratis-filesystem1 file system.

```
[root@servera ~]# stratis filesystem destroy stratispool1 stratis-filesystem1
```

- 10.3. 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, 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 lvm-stratis
```

This concludes the section.

## ► Lab

# Manage Storage Stack

In this lab, you resize an existing logical volume, add LVM resources as necessary, and then add a new logical volume with a persistently mounted XFS file system on it.

## Outcomes

- Resize the `serverb_01_lv` logical volume to 768 MiB.
- Create the new `serverb_02_lv` logical volume with 128 MiB with an XFS file system.
- Persistently mount the volume on the `/storage/data2` directory.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

On the `serverb` machine, the `serverb_01_lv` logical volume mounted on the `/storage/data1` directory is running out of disk space and needs to be extended to 768 MiB. You must ensure that the `serverb_01_lv` LV remains persistently mounted on the `/storage/data1` directory.

The `serverb_01_lv` LV is present on the `serverb_01_vg` volume group. Unfortunately, it has insufficient space to extend the existing logical volume. A 512 MiB partition exists on the `/dev/vdb` disk. Create a new partition using the successive 512 MiB size on the `/dev/vdb` disk.

Create the `serverb_02_lv` LV with 128 MiB. Create the XFS file system on the newly created volume. Mount the newly created logical volume on the `/storage/data2` directory

1. Create a 512 MiB partition on the `/dev/vdb` disk. Initialize this partition as a physical volume, and extend the `serverb_01_vg` volume group to use this partition.
2. Extend the `serverb_01_lv` logical volume to 768 MiB.
3. In the existing volume group, create the new `serverb_02_lv` logical volume with 128 MiB. Add an XFS file system and mount it persistently on the `/storage/data2` directory.
4. Verify that the newly created LV is mounted with the desired size.

## 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, 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 lvm-review
```

This concludes the section.

## ► Solution

# Manage Storage Stack

In this lab, you resize an existing logical volume, add LVM resources as necessary, and then add a new logical volume with a persistently mounted XFS file system on it.

## Outcomes

- Resize the `serverb_01_lv` logical volume to 768 MiB.
- Create the new `serverb_02_lv` logical volume with 128 MiB with an XFS file system.
- Persistently mount the volume on the `/storage/data2` directory.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

On the `serverb` machine, the `serverb_01_lv` logical volume mounted on the `/storage/data1` directory is running out of disk space and needs to be extended to 768 MiB. You must ensure that the `serverb_01_lv` LV remains persistently mounted on the `/storage/data1` directory.

The `serverb_01_lv` LV is present on the `serverb_01_vg` volume group. Unfortunately, it has insufficient space to extend the existing logical volume. A 512 MiB partition exists on the `/dev/vdb` disk. Create a new partition using the successive 512 MiB size on the `/dev/vdb` disk.

Create the `serverb_02_lv` LV with 128 MiB. Create the XFS file system on the newly created volume. Mount the newly created logical volume on the `/storage/data2` directory

1. Create a 512 MiB partition on the `/dev/vdb` disk. Initialize this partition as a physical volume, and extend the `serverb_01_vg` volume group to use this partition.
  - 1.1. Log in to the `serverb` machine as the `student` user and switch to the `root` user.

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

- 1.2. Create the 512 MiB partition and set the `lvm` partition type.

```
[root@serverb ~]# parted /dev/vdb mkpart primary 514MiB 1026MiB
[root@serverb ~]# parted /dev/vdb set 2 lvm on
```

- 1.3. Register the new partition with the kernel.

```
[root@serverb ~]# udevadm settle
```

- 1.4. Initialize the partition as a PV.

```
[root@serverb ~]# pvcreate /dev/vdb2
Physical volume "/dev/vdb2" successfully created.
```

- 1.5. Extend the serverb\_01\_vg VG using the new /dev/vdb2 PV.

```
[root@serverb ~]# vgextend serverb_01_vg /dev/vdb2
Volume group "serverb_01_vg" successfully extended
```

2. Extend the serverb\_01\_lv logical volume to 768 MiB.

- 2.1. Extend the serverb\_01\_lv LV to 768 MiB.

Alternatively, you can also use the lvcreate command -L +512M option to resize the LV.

```
[root@serverb ~]# lvextend -L 768M /dev/serverb_01_vg/serverb_01_lv
Size of logical volume serverb_01_vg/serverb_01_lv changed from 256.00 MiB (64
extents) to 768.00 MiB (192 extents).
Logical volume serverb_01_vg/serverb_01_lv successfully resized.
```

- 2.2. Extend the XFS file system consuming the remaining space on the LV.

```
[root@serverb ~]# xfs_growfs /storage/data1
meta-data=/dev/mapper/serverb_01_vg-serverb_01_lv isize=512    agcount=4,
agsize=16384 blks
...output omitted...
data blocks changed from 65536 to 196608
```



### Note

The xfs\_growfs command introduces an additional step to extend the file system. An alternative would be using the lvextend command -r option.

3. In the existing volume group, create the new serverb\_02\_lv logical volume with 128 MiB. Add an XFS file system and mount it persistently on the /storage/data2 directory.

- 3.1. Create the serverb\_02\_lv LV with 128 MiB from the serverb\_01\_vg VG.

```
[root@serverb ~]# lvcreate -n serverb_02_lv -L 128M serverb_01_vg
Logical volume "serverb_02_lv" created.
```

3.2. Create the xfs file system on the serverb\_02\_lv LV.

```
[root@serverb ~]# mkfs -t xfs /dev/serverb_01_vg/serverb_02_lv
...output omitted...
```

3.3. Create the /storage/data2 directory as the mount point.

```
[root@serverb ~]# mkdir /storage/data2
```

3.4. Add the following line to the end of the /etc/fstab file:

```
/dev/serverb_01_vg/serverb_02_lv /storage/data2 xfs defaults 0 0
```

3.5. Update the systemd daemon with the new /etc/fstab configuration file.

```
[root@serverb ~]# systemctl daemon-reload
```

3.6. Mount the serverb\_02\_lv LV.

```
[root@serverb ~]# mount /storage/data2
```

**4.** Verify that the newly created LV is mounted with the desired size.

4.1. Use the df command to verify the serverb\_01\_lv LV size.

```
[root@serverb ~]# df -h /storage/data1
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/serverb_01_vg-serverb_01_lv  763M   19M  744M   3% /storage/data1
```

4.2. Verify the serverb\_02\_lv LV size.

```
[root@serverb ~]# df -h /storage/data2
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/serverb_01_vg-serverb_02_lv  123M   7.6M  116M   7% /storage/data2
```

4.3. Verify the serverb\_01\_lv LV details.

```
[root@serverb ~]# lvdisplay /dev/serverb_01_vg/serverb_01_lv
--- Logical volume ---
LV Path          /dev/serverb_01_vg/serverb_01_lv
LV Name          serverb_01_lv
VG Name          serverb_01_vg
LV UUID          1pY3DZ-fs1F-mptC-fL32-e8tG-PFBT-bs7LSJ
LV Write Access  read/write
LV Creation host, time serverb.lab.example.com, 2022-05-05 14:40:51 -0400
LV Status        available
# open           1
LV Size          768.00 MiB
Current LE       192
Segments         2
```

```
Allocation           inherit
Read ahead sectors   auto
- currently set to    8192
Block device         253:0
```

4.4. Verify the `serverb_02_lv` LV details.

```
[root@serverb ~]# lvdisplay /dev/serverb_01_vg/serverb_02_lv
--- Logical volume ---
LV Path          /dev/serverb_01_vg/serverb_02_lv
LV Name          serverb_02_lv
VG Name          serverb_01_vg
LV UUID          0aJIB6-Ti2b-jLCK-imB6-rkLx-mUoX-acjkz9
LV Write Access  read/write
LV Creation host, time serverb.lab.example.com, 2022-05-05 14:45:46 -0400
LV Status        available
# open           1
LV Size          128.00 MiB
Current LE       32
Segments         1
Allocation       inherit
Read ahead sectors  auto
- currently set to 8192
Block device     253:1
```

## 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 lvm-review
```

## 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 lvm-review
```

This concludes the section.

# Summary

---

- You can use LVM to create flexible storage by allocating space on multiple storage devices.
- You can manage physical volumes, volume groups, and logical volumes with the `pvcreate`, `vgreduce`, and `lvextend` commands.
- You can format logical volumes with a file system or swap space, and can mount them persistently.
- You can add storage to volume groups, and can extend logical volumes dynamically.
- You can use the layers and components of the storage stack to manage storage efficiently.
- You can use the Virtual Data Optimizer (VDO) for compression and deduplication of data in LVM storage.
- You can use Stratis for initial storage configuration or to enable advanced storage features.

## Chapter 11

# Control Services and Boot Process

### Goal

Control and monitor network services, system daemons and the boot process using `systemd`

### Objectives

- List system daemons and network services that were started by the systemd service and socket units.
- Control system daemons and network services with systemctl.
- Describe the Red Hat Enterprise Linux boot process, set the default target when booting, and boot a system to a non-default target.
- Log in to a system and change the root password when the current root password is lost.
- Manually repair file-system configuration or corruption issues that stop the boot process.

### Sections

- Identify Automatically Started System Processes (and Guided Exercise)
- Control System Services (and Guided Exercise)
- Select the Boot Target (and Guided Exercise)
- Reset the Root Password (and Guided Exercise)
- Repair File System Issues at Boot (and Guided Exercise)

### Lab

- Control the Boot Process

# Identify Automatically Started System Processes

## Objectives

List system daemons and network services that were started by the `systemd` service and socket units.

## Introduction to the `systemd` Daemon

The `systemd` daemon manages the startup process for Linux, including service startup and service management in general. The `systemd` daemon activates system resources, server daemons, and other processes both at boot time and on a running system.

Daemons are processes that either wait or run in the background, to perform various tasks. Generally, daemons start automatically at boot time and continue to run until shutdown or until you manually stop them. It is a convention to end the daemon names with the letter d.

A service in the `systemd` sense often refers to one or more daemons. However, starting or stopping a service might instead change the state of the system once, without leaving a running daemon process afterward (called `oneshot`).

In Red Hat Enterprise Linux, the first process that starts (PID 1) is the `systemd` daemon, which provides these features:

- Parallelization capabilities (starting multiple services simultaneously), which increase the boot speed of a system.
- On-demand starting of daemons without requiring a separate service.
- Automatic service dependency management, which can prevent long timeouts. For example, a network-dependent service does not attempt to start until the network is available.
- A method of tracking related processes together by using Linux control groups.

## Service Units Description

The `systemd` daemon uses *units* to manage different types of objects:

- Service *units* have a `.service` extension and represent system services. You can use service units to start frequently accessed daemons, such as a web server.
- Socket *units* have a `.socket` extension and represent inter-process communication (IPC) sockets that `systemd` should monitor. If a client connects to the socket, then the `systemd` manager starts a daemon and passes the connection to it. You can use socket units to delay the start of a service at boot time and to start less frequently used services on demand.
- Path *units* have a `.path` extension and delay the activation of a service until a specific file-system change occurs. You can use path units for services that use spool directories, such as a printing system.

To manage units, use the `systemctl` command. For example, display available unit types with the `systemctl -t help` command. The `systemctl` command can take abbreviated unit names, process tree entries, and unit descriptions.

## List Service Units

Use the `systemctl` command to explore the system's current state. For example, the following command lists and paginates all currently loaded service units.

```
[root@host ~]# systemctl list-units --type=service
UNIT           LOAD   ACTIVE   SUB      DESCRIPTION
atd.service    loaded  active   running  Job spooling tools
auditd.service loaded  active   running  Security Auditing Service
chronyd.service loaded  active   running  NTP client/server
crond.service  loaded  active   running  Command Scheduler
dbus.service   loaded  active   running  D-Bus System Message Bus
...output omitted...
```

In this example, the `--type=service` option limits the type of `systemd` units to service units. The output has the following columns:

### UNIT

The service unit name.

### LOAD

Whether the `systemd` daemon properly parsed the unit's configuration and loaded the unit into memory.

### ACTIVE

The high-level activation state of the unit. This information indicates whether the unit started successfully.

### SUB

The low-level activation state of the unit. This information indicates more detailed information about the unit. The information varies based on unit type, state, and how the unit is executed.

### DESCRIPTION

The short description of the unit.

By default, the `systemctl list-units --type=service` command lists only the service units with `active` activation states. The `systemctl list-units --all` option lists all service units regardless of the activation states. Use the `--state=` option to filter by the values in the LOAD, ACTIVE, or SUB fields.

```
[root@host ~]# systemctl list-units --type=service --all
UNIT           LOAD   ACTIVE   SUB      DESCRIPTION
atd.service    loaded  active   running  Job spooling tools
auditd.service loaded  active   running  Security Auditing ...
auth-rpcgss-module.service loaded  inactive dead    Kernel Module ...
chronyd.service loaded  active   running  NTP client/server
cpupower.service loaded  inactive dead    Configure CPU power ...
crond.service  loaded  active   running  Command Scheduler
dbus.service   loaded  active   running  D-Bus System Message Bus
● display-manager.service        not-found inactive dead    display-manager.service
...output omitted...
```

The `systemctl` command without any arguments lists units that are both loaded and active.

```
[root@host ~]# systemctl
UNIT                      LOAD ACTIVE SUB   DESCRIPTION
proc-sys-fs-binfmt_misc.automount    loaded active waiting  Arbitrary...
sys-devices-....device           loaded active plugged  Virtio network...
sys-subsystem-net-devices-ens3.device loaded active plugged  Virtio network...
...output omitted...
-.mount                     loaded active mounted  Root Mount
boot.mount                  loaded active mounted  /boot
...output omitted...
systemd-ask-password-plymouth.path  loaded active waiting  Forward Password...
systemd-ask-password-wall.path     loaded active waiting  Forward Password...
init.scope                  loaded active running  System and Servi...
session-1.scope             loaded active running  Session 1 of...
atd.service                 loaded active running  Job spooling tools
audited.service             loaded active running  Security Auditing...
chronyd.service             loaded active running  NTP client/server
crond.service               loaded active running  Command Scheduler
...output omitted...
```

The `systemctl` command `list-units` option displays units that the `systemd` service attempts to parse and load into memory. This option does not display services that are installed but not enabled. You can use the `systemctl` command `list-unit-files` option to see the state of all the installed unit files:

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                      STATE      VENDOR PRESET
arp-ethers.service            disabled   disabled
atd.service                   enabled    enabled
audited.service              enabled    enabled
auth-rpcgss-module.service   static     -
autovt@.service              alias     -
blk-availability.service     disabled   disabled
...output omitted...
```

In the output of the `systemctl list-unit-files` command, some common entries for the `STATE` field are `enabled`, `disabled`, `static`, and `masked`. All `STATE` values are listed in the `systemctl` command manual pages.

## View Service States

View a unit's status with the `systemctl status name.type` command. If the unit type is omitted, then the command expects a service unit with that name.

```
[root@host ~]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2022-03-14 05:38:12 EDT; 25min ago
       Docs: man:sshd(8)
              man:sshd_config(5)
   Main PID: 1114 (sshd)
      Tasks: 1 (limit: 35578)
     Memory: 5.2M
```

```
CPU: 64ms
CGroup: /system.slice/sshd.service
└─1114 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 14 05:38:12 workstation systemd[1]: Starting OpenSSH server daemon...
Mar 14 05:38:12 workstation sshd[1114]: Server listening on 0.0.0.0 port 22.
Mar 14 05:38:12 workstation sshd[1114]: Server listening on :: port 22.
Mar 14 05:38:12 workstation systemd[1]: Started OpenSSH server daemon.
...output omitted...
```

Some fields from the `systemctl` command `status` option output:

### Service Unit Information

Field	Description
Loaded	Whether the service unit is loaded into memory.
Active	Whether the service unit is running and if so, for how long.
Docs	Where to find more information about the service.
Main PID	The main process ID of the service, including the command name.
Status	More information about the service.
Process	More information about related processes.
CGroup	More information about related control groups.

Not all these fields are always present in the command output.

Keywords in the status output indicate the state of the service:

### Service States in the Output of `systemctl`

Keyword	Description
loaded	The unit configuration file is processed.
active (running)	The service is running with continuing processes.
active (exited)	The service successfully completed a one-time configuration.
active (waiting)	The service is running but waiting for an event.
inactive	The service is not running.
enabled	The service starts at boot time.
disabled	The service is not set to start at boot time.
static	The service cannot be enabled, but an enabled unit might start it automatically.

**Note**

The `systemctl status NAME` command replaces the service `NAME status` command from Red Hat Enterprise Linux 6 and earlier versions.

## Verify the Status of a Service

The `systemctl` command verifies the specific states of a service. For example, use the `systemctl` command `is-active` option to verify whether a service unit is active (running):

```
[root@host ~]# systemctl is-active sshd.service  
active
```

The command returns the service unit state, which is usually `active` or `inactive`.

Run the `systemctl` command `is-enabled` option to verify whether a service unit is enabled to start automatically during system boot:

```
[root@host ~]# systemctl is-enabled sshd.service  
enabled
```

The command returns whether the service unit is enabled to start at boot time, and is usually `enabled` or `disabled`.

To verify whether the unit failed during startup, run the `systemctl` command `is-failed` option:

```
[root@host ~]# systemctl is-failed sshd.service  
active
```

The command returns `active` if the service is properly running, or `failed` if an error occurred during startup. If the unit was stopped, it returns `unknown` or `inactive`.

To list all the failed units, run the `systemctl --failed --type=service` command.



### References

`systemd(1)`, `systemd.unit(5)`, `systemd.service(5)`, `systemd.socket(5)`, and `systemctl(1)` man pages

For more information, refer to the *Managing Services with systemd* chapter in the *Red Hat Enterprise Linux 9 Configuring Basic System Settings Guide* at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/managing-services-with-systemd\\_configuring-basic-system-settings#managing-services-with-systemd\\_configuring-basic-system-settings](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#managing-services-with-systemd_configuring-basic-system-settings)

## ► Guided Exercise

# Identify Automatically Started System Processes

In this exercise, you list installed service units and identify which services are currently enabled and active on a server.

### Outcomes

- List installed service units.
- Identify active and enabled services on the system.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start services-identify
```

### Instructions

- 1. Use the `ssh` command to log in to the `servera` machine as the `student` user.

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

- 2. List all installed service units on the `servera` machine.

```
[student@servera ~]$ systemctl list-units --type=service
UNIT                  LOAD    ACTIVE   SUB      DESCRIPTION
atd.service           loaded  active  running  Deferred execution scheduler
auditd.service        loaded  active  running  Security Auditing Service
chronyd.service       loaded  active  running  NTP client/server
crond.service         loaded  active  running  Command Scheduler
dbus-broker.service   loaded  active  running  D-Bus System Message Bus
...output omitted...
```

Press `q` to exit the command.

- 3. List all socket units, active and inactive, on the `servera` machine.

```
[student@servera ~]$ systemctl list-units --type=socket --all
UNIT                  LOAD    ACTIVE   SUB      DESCRIPTION
dbus.socket           loaded  active  running  D-Bus System Message Bus Socket
dm-event.socket       loaded  active  listening  Device-mapper event daemon FIFOs
```

```
lvm2-lvmpolld.socket loaded active listening LVM2 poll daemon socket
...output omitted...
```

```
LOAD = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB = The low-level unit activation state, values depend on unit type.
13 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.
```

- ▶ 4. Explore the status of the `chronyd` service. You can use this service for network time protocol synchronization (NTP).

- 4.1. Display the status of the `chronyd` service. Note the process ID of any active daemon.

```
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Mon 2022-03-14 05:38:15 EDT; 1h 16min ago
      Docs: man:chronyd(8)
             man:chrony.conf(5)
   Process: 728 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/
 SUCCESS)
 Main PID: 747 (chronyd)
   Tasks: 1 (limit: 10800)
     Memory: 3.7M
        CPU: 37ms
      CGroup: /system.slice/chronyd.service
              └─747 /usr/sbin/chronyd -F 2
```

```
Mar 14 05:38:15 servera.lab.example.com systemd[1]: Starting NTP client/server...
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: chronyd version 4.1 starting
(+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SIGND +ASYNCDNS +NTS +SECHASH
+IPV6 +DEBUG)
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: commandkey directive is no
longer supported
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: generatecommandkey directive
is no longer supported
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: Frequency -11.870 +/- 1.025
ppm read from /var/lib/chrony/drift
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: Loaded seccomp filter (level
2)
Mar 14 05:38:15 servera.lab.example.com systemd[1]: Started NTP client/server.
Mar 14 05:38:23 servera.lab.example.com chronyd[747]: Selected source
172.25.254.254
```

Press q to exit the command.

- 4.2. Confirm that the `chronyd` daemon is running by using its process ID. In the preceding command, the output of the process ID that is associated with the `chronyd` service is 747. The process ID might differ on your system.

```
[student@servera ~]$ ps -p 747
 PID TTY      TIME CMD
 747 ?        00:00:00 chronyd
```

- 5. Explore the status of the sshd service. You can use this service for secure encrypted communication between systems.

- 5.1. Determine whether the sshd service is enabled to start at system boot.

```
[student@servera ~]$ systemctl is-enabled sshd
enabled
```

- 5.2. Determine whether the sshd service is active without displaying all of the status information.

```
[student@servera ~]$ systemctl is-active sshd
active
```

- 5.3. Display the status of the sshd service.

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-03-14 05:38:16 EDT; 1h 19min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 784 (sshd)
    Tasks: 1 (limit: 10800)
   Memory: 6.7M
      CPU: 82ms
     CGroup: /system.slice/sshd.service
             └─784 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 14 05:38:16 servera.lab.example.com systemd[1]: Starting OpenSSH server
daemons...
Mar 14 05:38:16 servera.lab.example.com sshd[784]: Server listening on 0.0.0.0
port 22.
Mar 14 05:38:16 servera.lab.example.com sshd[784]: Server listening on :: port 22.
Mar 14 05:38:16 servera.lab.example.com systemd[1]: Started OpenSSH server daemon.
Mar 14 06:51:36 servera.lab.example.com sshd[1090]: Accepted publickey for student
from 172.25.250.9 port 53816 ssh2: RSA SHA256:M8ikhEDm2tQ95Z0o7ZvufqEixCFCT
+wowZLNzNlBT0
Mar 14 06:51:36 servera.lab.example.com sshd[1090]: pam_unix(sshd:session):
 session opened for user student(uid=1000) by (uid=0)
```

Press q to exit the command.

- 6. List the enabled or disabled states of all service units.

```
[student@servera ~]$ systemctl list-unit-files --type=service
UNIT FILE                      STATE      VENDOR PRESET
arp-ethers.service              disabled   disabled
atd.service                     enabled    enabled
audited.service                 enabled    enabled
auth-rpcgss-module.service     static     -
autovt@.service                alias     -
blk-availability.service       disabled   disabled
bluetooth.service              enabled    enabled
chrony-wait.service            disabled   disabled
chronyd.service                enabled    enabled
...output omitted...
```

Press q to exit the command.

- 7. Return to the `workstation` system as the `student` user.

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

## 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 services-identify
```

This concludes the section.

# Control System Services

## Objectives

Control system daemons and network services with `systemctl`.

## Start and Stop Services

You can manually start, stop, or reload services to update the service, update the configuration file, uninstall the service, or manually manage an infrequently used service.

Use the `systemctl status` command to verify the status of a service, if the service is running or stopped.

```
[root@host ~]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2022-03-23 11:58:18 EDT; 2min 56s ago
    ...output omitted...
```

Use the `systemctl start` command as the `root` user (with the `sudo` command if necessary). If you run the `systemctl start` command with the service name only (without the service type), then the `systemd` service looks for `.service` files.

```
[root@host ~]# systemctl start sshd
```

To stop a running service, use the `systemctl` command `stop` option. The following example shows how to stop the `sshd.service` service:

```
[root@host ~]# systemctl stop sshd.service
```

## Restart and Reload Services

When you restart a running service, the service first stops and then starts again. On the service restart, the new process gets a new ID during the startup and thus the process ID changes. To restart a running service, use the `systemctl` command `restart` option. The following example shows how to restart the `sshd` service:

```
[root@host ~]# systemctl restart sshd.service
```

Some services can reload their configuration files without requiring a restart, which is called a *service reload*. Reloading a service does not change the process ID that is associated with various service processes. To reload a running service, use the `systemctl` command `reload` option. The following example shows how to reload the `sshd.service` service after configuration changes:

```
[root@host ~]# systemctl reload sshd.service
```

If you are unsure whether the service has the function to reload the configuration file changes, use the `systemctl` command `reload-or-restart` option. The command reloads the configuration changes if the reloading function is available. Otherwise, the command restarts the service to implement the new configuration changes:

```
[root@host ~]# systemctl reload-or-restart sshd.service
```

## List Unit Dependencies

Some services require other services to be running first, which creates dependencies on the other services. Other services start only on demand, rather than at boot time. In both cases, `systemd` and `systemctl` start services as needed, whether to resolve the dependency or to start an infrequently used service. For example, if the printing system (CUPS) service is not running and you place a file into the print spool directory, then the system starts the CUPS-related daemons or commands to satisfy the print request.

```
[root@host ~]# systemctl stop cups.service
Warning: Stopping cups, but it can still be activated by:
  cups.path
  cups.socket
```

However, to completely stop the printing services on a system, you must stop all three units. Disabling the service disables the dependencies.

The `systemctl list-dependencies` *UNIT* command displays a hierarchy mapping of dependencies to start the service unit. To list reverse dependencies (units that depend on the specified unit), use the `--reverse` option with the command.

```
[root@host ~]# systemctl list-dependencies sshd.service
sshd.service
• └─system.slice
• └─sshd-keygen.target
•   └─sshd-keygen@ecdsa.service
•   └─sshd-keygen@ed25519.service
•   └─sshd-keygen@rsa.service
• └─sysinit.target
...output omitted...
```

## Mask and Unmask Services

At times, different installed services on your system might conflict with each other. For example, multiple methods are available to manage mail servers (the `postfix` and `sendmail` services). Masking a service prevents an administrator from accidentally starting a service that conflicts with others. Masking creates a link in the configuration directories to the `/dev/null` file which prevents the service from starting. To mask a service, use the `systemctl` command `mask` option.

```
[root@host ~]# systemctl mask sendmail.service
Created symlink /etc/systemd/system/sendmail.service → /dev/null.
```

Then, check the state of the service by using the `systemctl list-unit-files` command:

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                                     STATE
...output omitted...
sendmail.service                               masked
...output omitted...
```

Attempting to start a masked service unit fails with the following output:

```
[root@host ~]# systemctl start sendmail.service
Failed to start sendmail.service: Unit sendmail.service is masked.
```

Use the `systemctl unmask` command to unmask the service unit.

```
[root@host ~]# systemctl unmask sendmail
Removed /etc/systemd/system/sendmail.service.
```



### Important

You or another unit file can manually start a disabled service, but it does not start automatically at boot. A masked service will not start manually or automatically.

## Enable Services to Start or Stop at Boot

Starting a service on a running system does not guarantee that the service automatically starts when the system reboots. Similarly, stopping a service on a running system does not keep it from starting again when the system reboots. Creating links in the `systemd` configuration directories enables the service to start at boot. You can create or remove these links by using the `systemctl` command with the `enable` or `disable` option.

```
[root@root ~]# systemctl enable sshd.service
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service → /usr/
lib/systemd/system/sshd.service.
```

This command creates a symbolic link from the service unit file, usually in the `/usr/lib/systemd/system` directory, to the disk location where the `systemd` command looks for files, in the `/etc/systemd/system/TARGETNAME.target.wants` directory. Enabling a service does not start the service in the current session. To start the service and enable it to start automatically during boot, you can execute both the `systemctl start` and `systemctl enable` commands, or use the equivalent `systemctl enable --now` command.

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

To disable the service from starting automatically, use the `systemctl disable` command, which removes the symbolic link that was created while enabling a service. Disabling a service does not stop the service if it is currently running. To disable and stop a service, you can execute both the `systemctl stop` and `systemctl disable` commands, or use the equivalent `systemctl disable --now` command.

```
[root@host ~]# systemctl disable --now sshd.service
Removed /etc/systemd/system/multi-user.target.wants/sshd.service.
```

To verify whether the service is enabled or disabled, use the `systemctl is-enabled` command.

```
[root@host ~]# systemctl is-enabled sshd.service
enabled
```

## Summary of systemctl Commands

You can start and stop services on a running system and enable or disable them for an automatic start at boot time.

### Useful Service Management Commands

Command	Task
<code>systemctl status <i>UNIT</i></code>	View detailed information about a unit's state.
<code>systemctl stop <i>UNIT</i></code>	Stop a service on a running system.
<code>systemctl start <i>UNIT</i></code>	Start a service on a running system.
<code>systemctl restart <i>UNIT</i></code>	Restart a service on a running system.
<code>systemctl reload <i>UNIT</i></code>	Reload the configuration file of a running service.
<code>systemctl mask <i>UNIT</i></code>	Disable a service from being started, both manually and at boot.
<code>systemctl unmask <i>UNIT</i></code>	Make a masked service available.
<code>systemctl enable <i>UNIT</i></code>	Configure a service to start at boot time. Use the <code>--now</code> option to also start the service.
<code>systemctl disable <i>UNIT</i></code>	Disable a service from starting at boot time. Use the <code>--now</code> option to also stop the service.



### References

`systemd(1)`, `systemd.unit(5)`, `systemd.service(5)`, `systemd.socket(5)`, and `systemctl(1)` man pages

For more information, refer to the *Managing System Services with systemctl* chapter in the *Red Hat Enterprise Linux 9 Configuring Basic System Settings Guide* at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/index#managing-system-services-with-systemctl\\_configuring-basic-system-settings](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#managing-system-services-with-systemctl_configuring-basic-system-settings)

## ► Guided Exercise

# Control System Services

In this exercise, you use `systemctl` to stop, start, restart, reload, enable, and disable a `systemd`-managed service.

## Outcomes

- Use the `systemctl` command to control `systemd`-managed services.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start services-control
```

## 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. Restart and reload the `sshd` service and observe the results.

- 2.1. Display the status of the `sshd` service. Note the process ID of the `sshd` daemon. Press `q` to exit the command.

```
[root@servera ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:04:45 EDT; 16min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
 Main PID: 784 (sshd)
   Tasks: 1 (limit: 10799)
  Memory: 6.6M
  ...output omitted...
```

- 2.2. Restart the `sshd` service and view the status. In this example, the process ID of the daemon changes from 784 to 1193. Press `q` to exit the command.

```
[root@servera ~]# systemctl restart sshd
[root@servera ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:21:40 EDT; 5s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 1193 (sshd)
    Tasks: 1 (limit: 10799)
   Memory: 1.7M
...output omitted...
```

- 2.3. Reload the sshd service and view the status. The process ID of the daemon does not change. Press q to exit the command.

```
[root@servera ~]# systemctl reload sshd
[root@servera ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:21:40 EDT; 52s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Process: 1201 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/
SUCCESS)
  Main PID: 1193 (sshd)
    Tasks: 1 (limit: 10799)
   Memory: 1.7M
...output omitted...
```

- 3. Verify that the chronyd service is running. Press q to exit the command.

```
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:04:44 EDT; 19min ago
...output omitted...
```

- 4. Stop the chronyd service and view the status. Press q to exit the command.

```
[root@servera ~]# systemctl stop chronyd
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
preset: enabled)
  Active: inactive (dead) since Thu 2022-05-19 04:24:59 EDT; 4s ago
...output omitted...
```

```
May 19 04:24:59 servera.lab.example.com systemd[1]: Stopping NTP client/server...
May 19 04:24:59 servera.lab.example.com systemd[1]: chronyd.service: Deactivated
      successfully.
May 19 04:24:59 servera.lab.example.com systemd[1]: Stopped NTP client/server.
```

- 5. Determine whether the chronyd service is enabled to start at system boot.

```
[root@servera ~]# systemctl is-enabled chronyd
enabled
```

- 6. Reboot the servera machine, and then view the status of the chronyd service.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

Log in as the student user on the servera machine and switch to root user. View the status of the chronyd service. Press q to exit the command.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
   preset: enabled)
     Active: active (running) since Thu 2022-05-19 04:27:12 EDT; 40s ago
       ...output omitted...
```

- 7. Disable the chronyd service so that it does not start at boot, and then view the status of the service. Press q to exit the command.

```
[root@servera ~]# systemctl disable chronyd
Removed /etc/systemd/system/multi-user.target.wants/chronyd.service.
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor
   preset: enabled)
     Active: active (running) since Thu 2022-05-19 04:27:12 EDT; 2min 48s ago
       ...output omitted...
```

- 8. Reboot servera, and then view the status of the chronyd service.

```
[root@servera ~]# systemctl reboot  
Connection to servera closed by remote host.  
Connection to servera closed.  
[student@workstation ~]$
```

Log in as the student user on servera and view the status of the chronyd service. Press q to exit the command.

```
[student@workstation ~]$ ssh student@servera  
...output omitted...  
[student@servera ~]$ systemctl status chronyd  
● chronyd.service - NTP client/server  
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor  
   preset: enabled)  
     Active: inactive (dead)  
       Docs: man:chronyd(8)  
              man:chrony.conf(5)
```

- 9. Return to the workstation system as the student user.

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

## 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 services-control
```

This concludes the section.

# Select the Boot Target

## Objectives

Describe the Red Hat Enterprise Linux boot process, set the default target when booting, and boot a system to a non-default target.

## Describe the Red Hat Enterprise Linux 9 Boot Process

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. The following list gives a high-level overview of the tasks involved for a physical x86\_64 system booting Red Hat Enterprise Linux 9. The list for x86\_64 virtual machines is roughly the same, but the hypervisor handles some hardware-specific steps in software.

- The machine is powered on. The system firmware, either modern UEFI or older BIOS, runs a *Power On Self Test (POST)* and starts to initialize the hardware.

Configured using the system BIOS or UEFI configuration screens that you typically reach by pressing a specific key combination, such as F2, early during the boot process.

- The system firmware searches for a bootable device, either configured in the UEFI boot firmware or by searching for a *Master Boot Record (MBR)* on all disks, in the order configured in the BIOS or UEFI.

Configured using the system BIOS or UEFI configuration screens, which you typically reach by pressing a specific key combination, such as F2, early during the boot process.

- The system firmware reads a boot loader from disk and then passes control of the system to the boot loader. On a Red Hat Enterprise Linux 9 system, the boot loader is the *GRand Unified Bootloader version 2 (GRUB2)*.

Configured using the `grub2-install` command, which installs GRUB2 as the boot loader on the disk for BIOS systems. Do not use the `grub2-install` command directly to install the UEFI boot loader. RHEL 9 provides a prebuilt `/boot/efi/EFI/redhat/grubx64.efi` file, which contains the required authentication signatures for a Secure Boot system. Executing `grub2-install` directly on a UEFI system generates a new `grubx64.efi` file without those required signatures. You can restore the correct `grubx64.efi` file from the `grub2-efi` package.

- GRUB2 loads its configuration from the `/boot/grub2/grub.cfg` file for BIOS and from the `/boot/efi/EFI/redhat/grub.cfg` file for UEFI, and displays a menu where you can select which kernel to boot.

Configured using the `/etc/grub.d/` directory, and the `/etc/default/grub` file, the `grub2-mkconfig` command generates the `/boot/grub2/grub.cfg` or `/boot/efi/EFI/redhat/grub.cfg` files for BIOS or UEFI, respectively.

- After you select a kernel, or the timeout expires, the boot loader loads the kernel and `initramfs` from disk and places them in memory. An `initramfs` is an archive containing the kernel modules for all the hardware required at boot, initialization scripts, and more. On Red Hat Enterprise Linux 9, the `initramfs` contains an entire usable system by itself.

Configured using the `/etc/dracut.conf.d/` directory, the `dracut` command, and the `lsinitrd` command to inspect the `initramfs` file.

- The boot loader hands control over to the kernel, passing in any options specified on the kernel command line in the boot loader, and the location of the `initramfs` in memory.

Configured using the `/etc/grub.d/` directory, the `/etc/default/grub` file, and the `grub2-mkconfig` command to generate the `/boot/grub2/grub.cfg` file.

- The kernel initializes all hardware for which it can find a driver in the `initramfs`, then executes `/sbin/init` from the `initramfs` as PID 1. On Red Hat Enterprise Linux 9, `/sbin/init` is a link to `systemd`.

Configured using the kernel `init=` command-line parameter.

- The `systemd` instance from the `initramfs` executes all units for the `initrd.target` target. This includes mounting the root file system on disk on to the `/sysroot` directory.

Configured using the `/etc/fstab` file.

- The kernel switches (pivots) the root file system from `initramfs` to the root file system in the `/sysroot` directory. `systemd` then re-executes itself by using the copy of `systemd` installed on the disk.
- `systemd` looks for a default target, either passed in from the kernel command line or configured on the system, then starts (and stops) units to comply with the configuration for that target, solving dependencies between units automatically. In essence, a `systemd` target is a set of units that the system should activate to reach the desired state. These targets typically start a text-based login or a graphical login screen.

Configured using the `/etc/systemd/system/default.target` file and the `/etc/systemd/system/` directory.

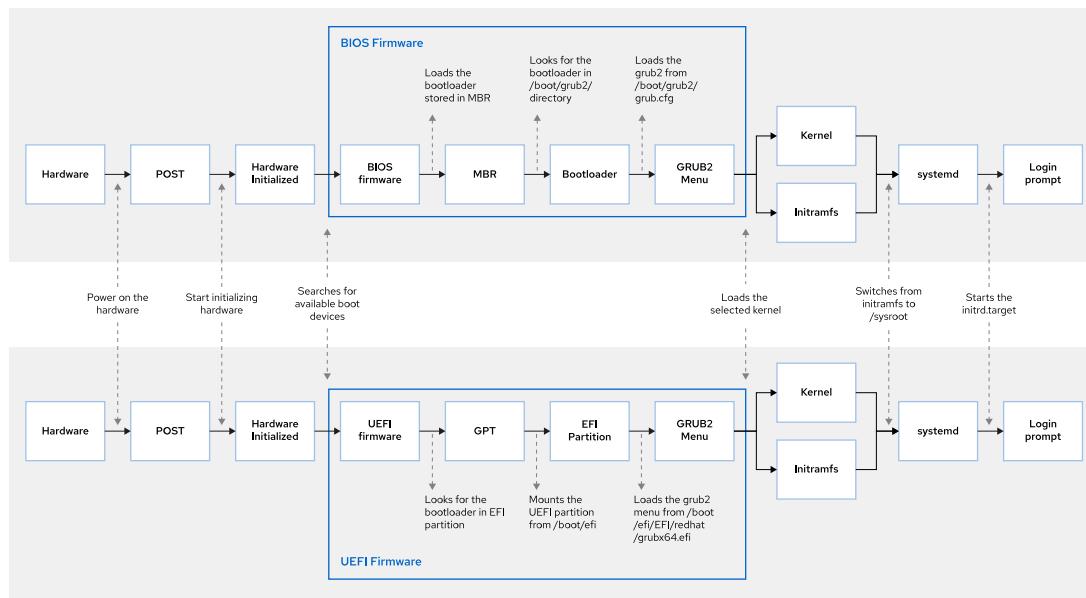


Figure 11.1: Boot process for BIOS-based and UEFI-based systems

## Reboot and Shut Down

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 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 their `systemctl` equivalents.



### Note

The `systemctl halt` and the `halt` commands are also available to stop the system, but unlike the `poweroff` command, these commands do not power off the system; they bring a system down to a point where it is safe to power it off manually.

## Select a Systemd Target

A `systemd` target is a set of `systemd` units that the system should start to reach a desired state. The following table lists the most important targets.

### Commonly Used Targets

Target	Purpose
<code>graphical.target</code>	System supports multiple users, graphical- and text-based logins.
<code>multi-user.target</code>	System supports multiple users, text-based logins only.
<code>rescue.target</code>	<code>sulogin</code> prompt, basic system initialization completed.
<code>emergency.target</code>	<code>sulogin</code> prompt, <code>initramfs</code> pivot complete, and system root mounted on <code>/</code> read only.

A target can be a part of another target. For example, the `graphical.target` includes `multi-user.target`, which in turn depends on `basic.target` 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 (Pre)   loaded  inactive dead   Local Encrypted Volumes
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 not required by that target (and its dependencies), and starts any required services not yet started.

Not all targets can be isolated. You can only isolate targets that have `AllowIsolate=yes` set in their unit files. For example, you can isolate the graphical target, but not the `cryptsetup` 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
[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, `systemd` activates the `default.target` target. Normally the default target in `/etc/systemd/system/` is a symbolic link to either the `graphical.target` or the

`multi-user.target` targets. Instead of editing this symbolic link by hand, the `systemctl` command provides two subcommands to manage this link: `get-default` and `set-default`.

```
[root@host ~]# systemctl get-default
multi-user.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.
[root@host ~]# systemctl get-default
graphical.target
```

## 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 only affects a single boot, making it a useful tool to troubleshoot the boot process.

To use this method to select a different target, use the following procedure:

1. Boot or reboot the system.
2. Interrupt the boot loader menu countdown by pressing any key (except Enter which would initiate a normal boot).
3. Move the cursor to the kernel entry that you want to start.
4. Press `e` to edit the current entry.
5. Move the cursor to the line that starts with `linux`. This 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.



## References

`info grub2 (GNU GRUB manual)`

`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 services with systemd* chapter in the *Configuring basic system settings* guide at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/index#managing-services-with-systemd](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#managing-services-with-systemd)

## ► Guided Exercise

# Select the Boot Target

In this exercise, you determine the default target into which a system boots, and boot that system into other targets.

### Outcomes

- Update the system default target and use a temporary target from the boot loader.

### Before You Begin

As the student user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start boot-selecting
```

### Instructions

- 1. On the **workstation** machine, open a terminal and confirm that the default target is **graphical.target**.

```
[student@workstation ~]$ systemctl get-default  
graphical.target
```

- 2. On the **workstation** machine, switch to the **multi-user** target manually without rebooting. Use the **sudo** command and if prompted, use **student** as the password.

```
[student@workstation ~]$ sudo systemctl isolate multi-user.target  
[sudo] password for student: student
```

- 3. Access a text-based console. Use the **Ctrl+Alt+F1** key sequence by using the relevant button or menu entry. Log in as the **root** user by using **redhat** as the password.



#### Note

Reminder: If you are using the terminal through a web page, then you can click the Show Keyboard icon in the menu on the right side of the screen under your web browser's URL bar.

```
workstation login: root  
Password: redhat  
[root@workstation ~]#
```

- 4. Configure the **workstation** machine to automatically boot into the **multi-user** target, and then reboot the **workstation** machine to verify. When done, change the default **systemd** target back to the **graphical** target.

- 4.1. Set the default 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. After reboot, the system presents a text-based console and not a graphical login screen.

```
[root@workstation ~]# systemctl reboot
```

- 4.3. Log in as the **root** user.

```
workstation login: root
Password: redhat
Last login: Thu Mar 28 14:50:53 on tty1
[root@workstation ~]#
```

- 4.4. Set the default **systemd** 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. In this second part of the exercise, you will practice by using rescue mode to recover the system.

Access the boot loader by rebooting **workstation** again. From within the boot loader menu, boot into the **rescue** target.

- 5.1. Initiate the reboot.

```
[root@workstation ~]# systemctl reboot
```

- 5.2. When the boot loader menu appears, press any key to interrupt the countdown (except **Enter**, which would initiate a normal boot).

- 5.3. Use the cursor keys to highlight the default boot loader entry.

- 5.4. Press **e** to edit the current entry.

- 5.5. Using the cursor keys, navigate to the line that starts with **linux**.

- 5.6. Press **End** to move the cursor to the end of the line.

- 5.7. Append **systemd.unit=rescue.target** to the end of the line.

- 5.8. Press **Ctrl+x** to boot by using the modified configuration.

5.9. Log in to rescue mode. You might need to hit enter to get a clean prompt.

```
Give root password for maintenance  
(or press Control-D to continue): redhat  
[root@workstation ~]#
```

- 6. Confirm that in rescue mode, the root file system is in read/write mode.

```
[root@workstation ~]# mount  
...output omitted...  
/dev/vda4 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 presents a graphical login. Log in as the **student** user.

## 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 boot-selecting
```

This concludes the section.

# Reset the Root Password

## Objectives

Log in to a system and change the root password when the current root password is lost.

## Reset the Root Password from the Boot Loader

One task that every system administrator should be able to accomplish is resetting a lost `root` password. This task is trivial if the administrator is still logged in, either as an unprivileged user but with full `sudo` access, or as `root`. This task becomes slightly more involved when the administrator is not logged in.

Several methods exist to set a new `root` password. A system administrator could, for example, boot the system by using a Live CD, mount the root file system from there, and edit `/etc/shadow`. In this section, we explore a method that does not require the use of external media.



### Note

On Red Hat Enterprise Linux 6 and earlier, administrators can boot the system into runlevel 1 to get a `root` prompt. The closest analogs to runlevel 1 on a Red Hat Enterprise Linux 8 or later machine, are the rescue and emergency targets, both of which require the `root` password to log in.

If your system was deployed from a Red Hat cloud image, then you will not have a rescue kernel in your boot menu, however your default kernel will have similar behavior that allows you to enter maintenance mode without the `root` password.

On Red Hat Enterprise Linux 9, it is possible to have the scripts that run from the `initramfs` pause at certain points, provide a `root` shell, and then continue when that shell exits. This is mostly meant for debugging, but you can also use this method to reset a lost `root` password.

Starting from Red Hat Enterprise Linux 9, if you install your system from a DVD, then the default kernel asks for the `root` password when you try to enter maintenance mode. Thus, for the purpose of resetting a lost `root` password, you need to use the rescue kernel. However, if you deploy your system from a Red Hat cloud image, then you do not have a rescue kernel in your boot menu, but your default kernel has the similar behavior that allows you to enter maintenance mode without the `root` password.

To access that `root` shell, follow these steps:

1. Reboot the system.
2. Interrupt the boot-loader countdown by pressing any key, except `Enter`.
3. Move the cursor to the rescue kernel entry to boot (the one with the word `rescue` in its name).
4. Press `e` to edit the selected entry.
5. Move the cursor to the kernel command line (the line that starts with `linux`).

6. Append `rd.break`. With that option, the system breaks just before the system hands control from the `initramfs` to the actual system.
7. Press `Ctrl+x` to boot with the changes.
8. Press `Enter` to perform maintenance when prompted.

At this point, the system presents a `root` shell, with the actual root file system on the disk mounted read-only on `/sysroot`. Because troubleshooting often requires modification to the root file system, you must remount the root file system as read/write. The following step shows how the `remount`, `rw` option to the `mount` command remounts the file system with the new option (`rw`) set.



### Note

Prebuilt images may place multiple `console=` arguments to the kernel to support a wide array of implementation scenarios. Those `console=` arguments indicate the devices to use for console output. The caveat with `rd.break` is that even though the system sends the kernel messages to all the consoles, the prompt ultimately uses whichever console is given last. If you do not get your prompt, then you might want to temporarily reorder the `console=` arguments when you edit the kernel command line from the boot loader.



### Important

The system has not yet enabled SELinux, therefore any file you create does not have SELinux context. Some tools, such as the `passwd` command, first create a temporary file, then replace it with the file that is intended for edit, effectively creating a new file without SELinux context. For this reason, when you use the `passwd` command with `rd.break`, the `/etc/shadow` file does not receive SELinux context.

To reset the `root` password from this point, use the following procedure:

1. Remount `/sysroot` as read/write.

```
sh-5.1# mount -o remount,rw /sysroot
```

2. Switch into a `chroot` jail, where `/sysroot` is treated as the root of the file-system tree.

```
sh-5.1# chroot /sysroot
```

3. Set a new `root` password.

```
sh-5.1# passwd root
```

4. Make sure that all unlabeled files, including `/etc/shadow` at this point, get relabeled during boot.

```
sh-5.1# touch /.autorelabel
```

5. Type `exit` twice. The first command exits the `chroot` jail, and the second command exits the `initramfs` debug shell.

At this point, the system continues booting, performs a full SELinux relabel, and then reboots again.

## Inspect Logs

Looking at the logs of previously failed boots can be useful. If the system journals are persistent across reboots, you can use the `journalctl` tool to inspect those logs.

Remember that by default, the system journals are kept in the `/run/log/journal` directory, which means the journals are cleared when the system reboots. To store journals in the `/var/log/journal` directory, which persists across reboots, set the `Storage` parameter to `persistent` in `/etc/systemd/journald.conf`.

```
[root@host ~]# vim /etc/systemd/journald.conf
...output omitted...
[Journal]
Storage=persistent
...output omitted...
[root@host ~]# systemctl restart systemd-journald.service
```

To inspect the logs of a previous boot, use the `journalctl` command `-b` option. Without any arguments, the `journalctl` command `-b` option only displays messages since the last boot. With a negative number as an argument, it displays the logs of previous boots.

```
[root@host ~]# journalctl -b -1 -p err
```

This command shows all messages rated as an error or worse from the previous boot.

## Repair Systemd Boot Issues

To troubleshoot service startup issues at boot time, Red Hat Enterprise Linux 8 and later have the following tools available.

### Enable the Early Debug Shell

By enabling the `debug-shell` service with `systemctl enable debug-shell.service`, the system spawns a `root` shell on TTY9 (`Ctrl+Alt+F9`) early during the boot sequence. This shell is automatically logged in as `root`, so that administrators can debug the system while the operating system is still booting.



#### Warning

Do not forget to disable the `debug-shell.service` service when you are done debugging, because it leaves an unauthenticated `root` shell open to anyone with local console access.

Alternatively, to activate the debug shell during the boot using the GRUB2 menu, follow these steps:

1. Reboot the system.

2. Interrupt the boot-loader countdown by pressing any key, except Enter.
3. Move the cursor to the kernel entry to boot.
4. Press **e** to edit the selected entry.
5. Move the cursor to the kernel command line (the line that starts with `linux`).
6. Append `systemd.debug-shell`. With this parameter, the system boots into the debug shell.
7. Press **Ctrl+x** to boot with the changes.

## Use the Emergency and Rescue Targets

By appending either `systemd.unit=rescue.target` or `systemd.unit=emergency.target` to the kernel command line from the boot loader, the system spawns into a rescue or emergency shell instead of starting normally. Both of these shells require the `root` password.

The emergency target keeps the root file system mounted read-only, while the rescue target waits for `sysinit.target` to complete, so that more of the system is initialized, such as the logging service or the file systems. The root user at this point can not make changes to `/etc/fstab` until the drive is remounted in a read write state with the `mount -o remount,rw /` command.

Administrators can use these shells to fix any issues that prevent the system from booting normally; for example, a dependency loop between services, or an incorrect entry in `/etc/fstab`. Exiting from these shells continues with the regular boot process.

## Identify Stuck Jobs

During startup, `systemd` spawns a number of jobs. If some of these jobs cannot complete, they block other jobs from running. To inspect the current job list, administrators can use the `systemctl list-jobs` command. Any jobs listed as running must complete before the jobs listed as waiting can continue.



### References

`dracut.cmdline(7)`, `systemd-journald(8)`, `journald.conf(5)`,  
`journalctl(1)`, and `systemctl(1)` man pages

## ► Guided Exercise

# Reset the Root Password

In this exercise, you reset the `root` password on a system.

### Outcomes

- Reset the lost `root` user password.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command runs a start script that determines if the `servera` machine is reachable on the network. It also resets the `root` password to a random string and sets a higher time-out for the GRUB2 menu.

```
[student@workstation ~]$ lab start boot-resetting
```

### Instructions

- ▶ 1. Reboot `servera`, and interrupt the countdown in the boot-loader menu.
  - 1.1. Locate the icon for the `servera` console, as appropriate for your classroom environment, then open the console.  
Send a `Ctrl+Alt+Delete` to your system by using the relevant button or menu entry.
  - 1.2. When the boot-loader menu appears, press any key to interrupt the countdown, except `Enter`.
- ▶ 2. Edit the rescue kernel boot-loader entry, in memory, to abort the boot process just after the kernel mounts all the file systems, but before it hands over control to `systemd`.
  - 2.1. Use the cursor keys to highlight the rescue kernel entry (the one with the word `rescue` in its name).
  - 2.2. Press `e` to edit the current entry.
  - 2.3. Use the cursor keys to navigate to the line that starts with `linux`.
  - 2.4. Press `End` to move the cursor to the end of the line.
  - 2.5. Append `rd.break` to the end of the line.
  - 2.6. Press `Ctrl+x` to boot using the modified configuration.
- ▶ 3. Press `Enter` to perform maintenance. At the `sh-5.1#` prompt, remount the `/sysroot` file system read/write, then use the `chroot` command to enter a `chroot` jail at `/sysroot`.

```
sh-5.1# mount -o remount,rw /sysroot  
...output omitted...  
sh-5.1# chroot /sysroot
```

- ▶ 4. Change the root password back to redhat.

```
sh-5.1# passwd root  
Changing password for user root.  
New password: redhat  
BAD PASSWORD: The password is shorter than 8 characters  
Retype new password: redhat  
passwd: all authentication tokens updated successfully.
```

- ▶ 5. Configure the system to automatically perform a full SELinux relabel after boot. This is necessary because the passwd command recreates the /etc/shadow file without an SELinux context.

```
sh-5.1# touch /.autorelabel
```

- ▶ 6. Type exit twice to continue booting your system as usual. The system runs an SELinux relabel operation, then reboots automatically. When the system is up, verify your work by logging in as root at the console.

## 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 boot-resetting
```

This concludes the section.

# Repair File System Issues at Boot

## Objectives

Manually repair file-system configuration or corruption issues that stop the boot process.

## File-system Issues

During the boot process, the `systemd` service mounts the persistent file-systems defined in the `/etc/fstab` file.

Errors in the `/etc/fstab` file or corrupt 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 password.

The following list describes some common file system mounting issues when parsing `/etc/fstab` during the boot process:

### Corrupt file system

The `systemd` service attempts to repair the file system. If the problem can not be automatically repaired, then the system opens an emergency shell.

### Nonexistent device or UUID

The `systemd` service times out waiting for the device to become available. If the device does not respond, then the system opens an emergency shell.

### Nonexistent or incorrect mount point

The system opens an emergency shell.

## Repair File-system Issues

To gain access to a system that cannot complete booting because of file-system issues, the `systemd` architecture provides an `emergency` boot target, which opens an emergency shell that requires the `root` password for access.

The next example demonstrates 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/vda2 (27s / 1min 30s)
[ TIME ] Timed out waiting for device /dev/vda2.
[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):
```

## Chapter 11 | Control Services and Boot Process

The `systemd` daemon failed to mount the `/dev/vda2` 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, determine and repair the fault, then reload the '`systemd`' configuration to retry the automatic mounting.

Use the `mount` command to determine which file systems are currently mounted by the `systemd` daemon.

```
[root@host ~]# mount  
...output omitted...  
/dev/vda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)  
...output omitted...
```

If the root file system is mounted with the `ro` (read-only) option, then you are unable to edit the `/etc/fstab` file. Temporarily remount the root file system with the `rw` (read-write) option, if necessary, before opening the `/etc/fstab` file. The remount option allows an in-use file system to modify its mount parameters without actually unmounting the file system.

```
[root@host ~]# mount -o remount,rw /
```

Attempt to mount all the file systems listed in the `/etc/fstab` file by using the `mount --all` option. This option mounts processes on every entry file-system entry, but skips those 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, the `/mnt/mountFolder` mount directory does not exist, create the `/mnt/mountFolder` directory before reattempting the mount. Other error messages can occur, including typos in the entries, or incorrect device names or UUIDs.

When you have corrected all of the issues in the `/etc/fstab` file, inform the `systemd` daemon to register the new `/etc/fstab` file by using the `systemctl daemon-reload` command, then reattempt mounting all of the entries.

```
[root@host ~]# systemctl daemon-reload  
[root@host ~]# mount --all
```



### Note

The `systemd` service processes the `/etc/fstab` file by transforming each entry into a `.mount` type `systemd` unit configuration and then starting the unit as a service. The `daemon-reload` requests `systemd` to rebuild and reload all unit configurations.

If the `mount --all` attempt succeeds without further errors, then the final test is to verify that file-system mounting is successfully during a system boot. Reboot the system and wait for the boot to complete normally.

```
[root@host ~]# systemctl reboot
```

To perform quick tests in the `/etc/fstab` file, use the `nofail` mount entry option. Using the `nofail` option in an `/etc/fstab` entry allows the system to boot even if that file-system mount is unsuccessful. This option should not be used with production file systems that must always mount. With the `nofail` option, an application could start with its file-system data missing, with possibly severe consequences.



### References

`systemd-fsck(8)`, `systemd-fstab-generator(8)`, and `systemd.mount(5)`  
man pages

## ► Guided Exercise

# Repair File System Issues at Boot

In this exercise, you recover a system from a misconfiguration in the `/etc/fstab` file where the boot process fails.

## Outcomes

- Diagnose `/etc/fstab` file issues and use emergency mode to recover the system.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start boot-repairing
```

## Instructions

- 1. Access the `servera` machine console and notice that the boot process is stuck early on.
  - 1.1. Locate the icon for the `servera` console, as appropriate for your classroom environment. Open the console.

Notice that a start job does not seem to complete. Take a minute to speculate about a possible cause for this behavior.
  - 1.2. Reboot the `servera` machine, sending a `Ctrl+Alt+Del` to your system by using the relevant button or menu entry. With this particular boot problem, this key sequence might not immediately abort the running job, and you might have to wait for it to time out before the system reboots.

If you wait for the task to time out without sending a `Ctrl+Alt+Del`, then the system eventually spawns an emergency shell by itself.
  - 1.3. When the boot-loader menu appears, press any key to interrupt the countdown, except the `Enter` key.
- 2. Looking at the error from the previous boot, it appears that at least parts of the system are still functioning. Use `redhat` as the `root` user password to attempt an emergency boot.
  - 2.1. Use the cursor keys to highlight the default boot loader entry.
  - 2.2. Press the `e` key to edit the current entry.
  - 2.3. Use the cursor keys to navigate to the line that starts with the `linux` word.
  - 2.4. Press `End` to move the cursor to the end of the line.
  - 2.5. Append the `systemd.unit=emergency.target` string to the end of the line.

2.6. Press **Ctrl+x** to boot by using the modified configuration.

- 3. Log in to emergency mode.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@servera ~]#
```

- 4. Determine which file systems the **systemd** daemon currently mounts. Notice the **systemd** daemon mounts the root file system in read-only mode.

```
[root@servera ~]# mount
...output omitted...
/dev/vda4 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. Attempt to mount all the other file systems. The **--all (-a)** option mounts all the file systems listed in the **/etc/fstab** file that are not yet mounted.

```
[root@servera ~]# mount -a
mount: /RemoveMe: mount point does not exist.
```

- 7. Edit the **/etc/fstab** file to fix the issue.

7.1. Remove or comment out the incorrect line by using the **vim /etc/fstab** command.

```
[root@servera ~]# cat /etc/fstab
...output omitted...
# /dev/sdz1  /RemoveMe  xfs  defaults  0 0
```

7.2. Reload the **systemd** daemon 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 normally.

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

## 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 boot-repairing
```

This concludes the section.

## ▶ Lab

# Control the Boot Process

In this lab, you reset the `root` password on a system, recover from a misconfiguration, and set the default boot target.

## Outcomes

- Reset a lost password for the `root` user.
- Diagnose and fix boot issues.
- Set the default `systemd` target.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. On the `serverb` machine, reset the password to `redhat` for the `root` user.  
Locate the icon for the `serverb` machine console as appropriate for your classroom environment, then open the console.
2. In the boot-loader menu, select the default kernel boot-loader entry. The system fails to boot because a start job does not complete successfully. Fix the issue from the console of the `serverb` machine.
3. Change the default `systemd` 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. Only set the default target for this exercise and do not install the packages.

## 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, 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 boot-review
```

This concludes the section.

## ► Solution

# Control the Boot Process

In this lab, you reset the `root` password on a system, recover from a misconfiguration, and set the default boot target.

## Outcomes

- Reset a lost password for the `root` user.
- Diagnose and fix boot issues.
- Set the default `systemd` target.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. On the `serverb` machine, reset the password to `redhat` for the `root` user.  
Locate the icon for the `serverb` machine console as appropriate for your classroom environment, then open the console.
  - 1.1. Send a `Ctrl+Alt+Del` to your system by using the relevant button or menu entry.
  - 1.2. When the boot-loader menu appears, press any key to interrupt the countdown, except the `Enter` key.
  - 1.3. Use the cursor keys to highlight the rescue kernel boot-loader entry (the one with the word `rescue` in its name).
  - 1.4. Press `e` to edit the current entry.
  - 1.5. Use the cursor keys to navigate the line that starts with the `linux` text.
  - 1.6. Press `Ctrl+e` to move the cursor to the end of the line.
  - 1.7. Append the `rd.break` text to the end of the line.
  - 1.8. Press `Ctrl+x` to boot using the modified configuration.
  - 1.9. Press `Enter` to enter the maintenance mode.
  - 1.10. At the `sh-5.1` prompt, remount the `/sysroot` file system as writable, and then use the `chroot` command for the `/sysroot` directory.

```
sh-5.1# mount -o remount,rw /sysroot
...output omitted...
sh-5.1# chroot /sysroot
```

- 1.11. Set **redhat** as the password for the **root** user.

```
sh-5.1# passwd root
Changing password for user root.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 1.12. Configure the system to perform a full SELinux relabel after boot automatically.

```
sh-5.1# touch /.autorelabel
```

- 1.13. Exit the **chroot** environment and the **sh-5.1** prompt. After the file system is relabeled, the system prompts to enter maintenance mode, but if you wait, it completes the reboot and shows the boot-loader menu.
2. In the boot-loader menu, select the default kernel boot-loader entry. The system fails to boot because a start job does not complete successfully. Fix the issue from the console of the **serverb** machine.
  - 2.1. Boot the system into emergency mode. Reboot the **serverb** machine by sending a **Ctrl+Alt+Del** to your system by using the relevant button or menu entry.
  - 2.2. When the boot-loader menu appears, press any key to interrupt the countdown, except **Enter**.
  - 2.3. Use the cursor keys to highlight the default boot-loader entry.
  - 2.4. Press **e** to edit the current entry.
  - 2.5. Use the cursor keys to navigate the line that starts with **linux** text.
  - 2.6. Press **Ctrl+e** to move the cursor to the end of the line.
  - 2.7. Append the **systemd.unit=emergency.target** text to the end of the line.
  - 2.8. Press **Ctrl+x** to boot using the modified configuration.
  - 2.9. Log in to emergency mode.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

- 2.10. Remount the **/** file system as writable.

```
[root@serverb ~]# mount -o remount,rw /
...output omitted...
```

- 2.11. Mount all file systems.

```
[root@serverb ~]# mount -a  
mount: /olddata: can't find UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc.
```

- 2.12. Edit the `/etc/fstab` file to remove or comment out the incorrect line mounting the `/olddata` mount point.

```
[root@serverb ~]# vim /etc/fstab  
...output omitted...  
#UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc  /olddata  xfs  defaults  0 0
```

- 2.13. Update the `systemd` daemon for the system to register the changes in the `/etc/fstab` file configuration.

```
[root@serverb ~]# systemctl daemon-reload
```

- 2.14. Verify that the `/etc/fstab` file configuration is correct by attempting to mount all entries.

```
[root@serverb ~]# mount -a
```

- 2.15. Reboot the system and wait for the boot to complete. The system should now boot normally.

```
[root@serverb ~]# systemctl reboot
```

3. Change the default `systemd` 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. Only set the default target for this exercise and do not install the packages.

- 3.1. Log in to the `servera` machine as the `student` user and switch to the `root` user.

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

- 3.2. 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.
```

- 3.3. Verify that the correct default is set.

```
[root@serverb ~]# systemctl get-default  
graphical.target
```

3.4. Return to the **workstation** machine as the **student** user.

```
[root@serverb ~]# exit  
logout  
[student@serverb ~]$ exit  
logout  
Connection to serverb 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 boot-review
```

## 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 boot-review
```

This concludes the section.

# Summary

---

- Use the `systemctl` utility to start, stop, reload, enable, and disable services.
- Use the `systemd` utility to manage service units, socket units, and path units.
- Use the `systemctl status` command to determine the status of system daemons and network services that `systemd` started.
- The `systemctl list-dependencies` command lists all service units that a specific service unit depends on.
- The `systemd` utility can mask a service unit so that it does not run even to satisfy dependencies.
- The `systemctl reboot` and `systemctl poweroff` commands reboot and power down a system, respectively.
- The `systemctl isolate target-name.target` command switches to a new target at runtime.
- The `systemctl get-default` and `systemctl set-default` commands query and set the default target.
- The `rd.break` option on the kernel command line interrupts the boot process before control is handed over from the initramfs file system. The root file system is mounted read-only under the `/sysroot` directory.
- The emergency target can help to diagnose and fix file-system issues.

## Chapter 12

# Analyze and Store Logs

### Goal

Locate and accurately interpret system event logs for troubleshooting purposes.

### Objectives

- Describe the basic Red Hat Enterprise Linux logging architecture to record events.
- Interpret events in relevant syslog files to troubleshoot problems or review system status.
- Find and interpret entries in the system journal to troubleshoot problems or review system status.
- Configure the system journal to preserve the record of events when a server is rebooted.
- Maintain accurate time synchronization with Network Time Protocol (NTP) and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

### Sections

- Describe System Log Architecture (and Quiz)
- Review Syslog Files (and Guided Exercise)
- Review System Journal Entries (and Guided Exercise)
- Preserve the System Journal (and Guided Exercise)
- Maintain Accurate Time (and Guided Exercise)

### Lab

Analyze and Store Logs

# Describe System Log Architecture

## Objectives

Describe the basic Red Hat Enterprise Linux logging architecture to record events.

## 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 9.

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
- 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 syslog messages that the `systemd-journald` service receives from the journal as 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.

The `rsyslog` service sorts and writes syslog messages to the log files that do persist across reboots in the `/var/log` directory. The service also sorts the log messages to specific log files according to the type of program that sent each message and the priority of each syslog message.

In addition to 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.

### Selected System Log Files

Log file	Type of stored messages
<code>/var/log/messages</code>	Most syslog 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>	Syslog messages about security and authentication events.
<code>/var/log/maillog</code>	Syslog messages about the mail server.

Log file	Type of stored messages
/var/log/cron	Syslog messages about scheduled job execution.
/var/log/boot.log	Non-syslog console messages about system startup.

Some applications do not use the sys log service to manage their log messages. For example, the Apache Web 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 Using Log Files* section in the *Red Hat Enterprise Linux 9 Configuring Basic System Settings Guide* at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index)

## ► Quiz

# Describe System Log Architecture

Choose the correct answer 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 /var/log?
  - a. rsyslog
  - b. systemd-journald
  - c. auditd
  - 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

# Describe System Log Architecture

Choose the correct answer 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 /var/log?
  - a. rsyslog
  - b. systemd-journald
  - c. auditd
  - 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

# Review Syslog Files

---

## Objectives

Interpret events in relevant syslog files to troubleshoot problems or review system status.

## Log Events to the System

Many programs use the syslog protocol to log events to the system. Each log message is categorized by facility (which subsystem produces the message) and priority (the message's severity).

The following table lists the standard syslog facilities.

### 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 daemons messages
4	auth	Authentication and security messages
5	syslog	Internal syslog messages
6	lpr	Printer messages
7	news	Network news messages
8	uucp	UUCP protocol messages
9	cron	Clock daemon messages
10	authpriv	Non-system 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.

### Overview of Syslog Priorities

Code	Priority	Priority description
0	emerg	System is unusable

Code	Priority	Priority description
1	alert	Action must be taken immediately
2	crit	Critical condition
3	err	Non-critical error condition
4	warning	Warning condition
5	notice	Normal but significant event
6	info	Informational event
7	debug	Debugging-level message

The `rsyslog` service uses the facility and priority of log messages to determine how to handle them. Rules configure this 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 easily add rules by installing an appropriate file in the `/etc/rsyslog.d` directory.

Each rule that controls how to sort syslog messages has a line in one of the configuration files. The left side of each line indicates the facility and priority of the syslog messages that the rule matches. The right side of each line indicates which file to save the log message in (or where else to deliver the message). An asterisk (\*) is a wildcard that matches all values.

For example, the following line in the `/etc/rsyslog.d` file would record messages that are sent to the `authpriv` facility at any priority to the `/var/log/secure` file:

```
authpriv.*           /var/log/secure
```

Sometimes, log messages match more than one rule in the `rsyslog.conf` file. One message is stored in more than one log file in such cases. 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, 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                                         :omusrmsg:

# 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 rotates log files to prevent them from taking 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 old `/var/log/messages` file is renamed to the `/var/log/messages-20220320` file when it is rotated on 2022-03-20. After the old log file rotates, it creates a log file and notifies the service that wrote the log file.

After rotations during typically four weeks, the oldest log file is discarded to free disk space. A scheduled job runs the `logrotate` command daily to see the rotation requirement of any log files. Most log files rotate weekly; the `logrotate` command rotates some log files faster, or more slowly, or when they reach a specific size.

## Analyze a Syslog Entry

Log messages start with the oldest message at the start and the newest message at the end of the log file. The `rsyslog` service uses a standard format for recording entries in log files. The following example explains 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**: Records the time stamp of the log entry.
- **localhost**: The host that sends 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, to monitor for failed login attempts, run the `tail` command in one terminal, and then run in another terminal the `ssh` command as the `root` user while a user tries to log in to the system.

In the first terminal, run the `tail` command:

```
[root@host ~]# tail -f /var/log/secure
```

In the second terminal, run the `ssh` command:

```
[root@host ~]# ssh root@hostA
root@hostA's password: redhat
...output omitted...
[root@hostA ~]#
```

The log messages are visible in the first terminal.

```
...output omitted...
Mar 20 09:01:13 host sshd[2712]: Accepted password for root from 172.25.254.254
port 56801 ssh2
Mar 20 09:01:13 host sshd[2712]: pam_unix(sshd:session): session opened for user
root by (uid=0)
```

## Send Syslog Messages Manually

The `logger` command sends messages to the `rsyslog` service. By default, it sends the message to the user type with the `notice` priority (`user.notice`) unless specified otherwise with the `-p` option. It is helpful to test any change to the `rsyslog` service configuration.

To send a message to the `rsyslog` service to be recorded in the `/var/log/boot.log` log file, execute the following `logger` command:

```
[root@host ~]# logger -p local7.notice "Log entry created on host"
```



## References

`logger(1)`, `tail(1)`, `rsyslog.conf(5)`, and `logrotate(8)` man pages

`rsyslog` Manual

- `/usr/share/doc/rsyslog/html/index.html` provided by the `rsyslog-doc` package

For further information, refer to *Troubleshooting Problems Using Log Files* at  
[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/assembly\\_troubleshooting-problems-using-log-files\\_configuring-basic-system-settings](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/assembly_troubleshooting-problems-using-log-files_configuring-basic-system-settings)

## ► Guided Exercise

# Review Syslog Files

In this exercise, you reconfigure the `rsyslog` service to write specific log messages to a new file.

### Outcomes

- Configure the `rsyslog` service to write all log messages with the debug priority to the `/var/log/messages-debug` log file.

### Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start logs-syslog
```

### 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. Configure the `rsyslog` service on the `servera` machine to log all messages with the debug or higher priority, for any service into the new `/var/log/messages-debug` log file by changing the `/etc/rsyslog.d/debug.conf` configuration file.

- 2.1. Create the `/etc/rsyslog.d/debug.conf` file with the necessary entries to redirect all log messages with the debug priority to the `/var/log/messages-debug` log file.

```
* .debug /var/log/messages-debug
```

This configuration line logs syslog messages with any type and with the debug or higher priority level. The `rsyslog` service writes the matching messages to the `/var/log/messages-debug` log file. The wildcard (\*) in the type or priority fields of the configuration line indicates any type or priority of log messages.

- 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 ten 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"
  swVersion="8.37.0-9.el8" x-pid="25176" x-info="http://www.rsyslog.com"] exiting
  on signal 15.
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 not set, auto
  correcting this to TZ=/etc/localtime [v8.37.0-9.el8 try http://www.rsyslog.com/
  e/2442 ]
Feb 13 18:22:38 servera systemd[1]: Started System Logging Service.
Feb 13 18:22:38 servera rsyslogd[25410]: [origin software="rsyslogd"
  swVersion="8.37.0-9.el8" x-pid="25410" x-info="http://www.rsyslog.com"] start
Feb 13 18:27:58 servera student[25416]: Debug Message Test
```

3.3. 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, 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 logs-syslog
```

This concludes the section.

# Review System Journal Entries

## Objectives

Find and interpret entries in the system journal to troubleshoot problems or review system status.

## Find Events on the System Journal

The `systemd-journald` service stores logging data in a structured, indexed binary file called *journal*. This data includes extra information about the log event. For example, for 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 chapter.

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 a wide range of options and criteria. If you run the command as `root`, then you have full access to the journal. Regular users can also use the `journalctl` command, but the system restricts them from seeing certain messages.

```
[root@host ~]# journalctl
...output omitted...
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Listening on PipeWire
Multimedia System Socket.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Starting Create User's
Volatile Files and Directories...
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Listening on D-Bus User
Message Bus Socket.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Sockets.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Finished Create User's
Volatile Files and Directories.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Basic System.
Mar 15 04:42:16 host.lab.example.com systemd[1]: Started User Manager for UID 0.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Main User
Target.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Startup finished in 90ms.
Mar 15 04:42:16 host.lab.example.com systemd[1]: Started Session 6 of User root.
Mar 15 04:42:16 host.lab.example.com sshd[2110]: pam_unix(sshd:session): session
opened for user root(uid=0) by (uid=0)
Mar 15 04:42:17 host.lab.example.com systemd[1]: Starting Hostname Service...
Mar 15 04:42:17 host.lab.example.com systemd[1]: Started Hostname Service.
lines 1951-2000/2000 (END) q
```

The `journalctl` command highlights important log messages; messages at `notice` or `warning` priority are in bold text, while messages at 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 you want to display. For example, if you want to review the last five log entries, then you can run the following `journalctl` command:

```
[root@host ~]# journalctl -n 5
Mar 15 04:42:17 host.lab.example.com systemd[1]: Started Hostname Service.
Mar 15 04:42:47 host.lab.example.com systemd[1]: systemd-hostnamed.service:
  Deactivated successfully.
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Created slice User Background
  Tasks Slice.
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Starting Cleanup of User's
  Temporary Files and Directories...
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Finished Cleanup of User's
  Temporary Files and Directories.
```

Similar to the `tail` command, the `journalctl` command `-f` option outputs the last 10 lines of the system journal and continues to output new journal entries as the journal appends them. To exit the `journalctl` command `-f` option, use the `Ctrl+C` key combination.

```
[root@host ~]# journalctl -f
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Finished Cleanup of User's
  Temporary Files and Directories.
Mar 15 05:01:01 host.lab.example.com CROND[2197]: (root) CMD (run-parts /etc/
  cron.hourly)
Mar 15 05:01:01 host.lab.example.com run-parts[2200]: (/etc/cron.hourly) starting
  @anacron
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Anacron started on 2022-03-15
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.daily' in
  29 min.
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.weekly' in
  49 min.
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.monthly' in
  69 min.
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Jobs will be executed
  sequentially
Mar 15 05:01:01 host.lab.example.com run-parts[2210]: (/etc/cron.hourly) finished
  @anacron
Mar 15 05:01:01 host.lab.example.com CROND[2196]: (root) CMDEND (run-parts /etc/
  cron.hourly)
^C
[root@host ~]#
```

To help to troubleshoot problems, you might want to filter the output of the journal by the priority of the journal entries. The `journalctl` command `-p` option shows the journal entries at 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 at the `err` priority or higher:

```
[root@host ~]# journalctl -p err
Mar 15 04:22:00 host.lab.example.com pipewire-pulse[1640]: pw.conf: execvp error
  'pactl': No such file or direct
Mar 15 04:22:17 host.lab.example.com kernel: Detected CPU family 6 model 13
  stepping 3
Mar 15 04:22:17 host.lab.example.com kernel: Warning: Intel Processor - this
  hardware has not undergone testing by Red Hat and might not be certif>
Mar 15 04:22:20 host.lab.example.com smartd[669]: DEVICESCAN failed: glob(3)
  aborted matching pattern /dev/disks/disc*
Mar 15 04:22:20 host.lab.example.com smartd[669]: In the system's table of devices
  NO devices found to scan
```

You might want to show messages for a specified systemd unit. 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
May 15 04:30:18 host.lab.example.com systemd[1]: Starting OpenSSH server daemon...
May 15 04:30:18 host.lab.example.com sshd[1142]: Server listening on 0.0.0.0 port
  22.
May 15 04:30:18 host.lab.example.com sshd[1142]: Server listening on :: port 22.
May 15 04:30:18 host.lab.example.com systemd[1]: Started OpenSSH server daemon.
May 15 04:32:03 host.lab.example.com sshd[1796]: Accepted publickey for user1 from
  172.25.250.254 port 43876 ssh2: RSA SHA256:1UGy...>
May 15 04:32:03 host.lab.example.com sshd[1796]: pam_unix(sshd:session): session
  opened for user user1(uid=1000) by (uid=0)
May 15 04:32:26 host.lab.example.com sshd[1866]: Accepted publickey for user2
  from ::1 port 36088 ssh2: RSA SHA256:M8ik...
May 15 04:32:26 host.lab.example.com sshd[1866]: pam_unix(sshd:session): session
  opened for user user2(uid=1001) by (uid=0)
lines 1-8/8 (END) q
```

When looking for specific events, you can limit the output to a specific time frame. The `journalctl` command has two options to limit the output to a specific time range, the `--since` and `--until` options. 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 in the option).

The `journalctl` command assumes that the day starts at 00:00:00 when you omit the time argument. The command also assumes the current day when you omit the day argument. Both options take `yesterday`, `today`, and `tomorrow` as valid arguments in addition to the date and time field.

As an example, run the following `journalctl` command to list all journal entries from today's records.

```
[root@host ~]# journalctl --since today
...output omitted...
Mar 15 05:04:20 host.lab.example.com systemd[1]: Started Session 8 of User
  student.
Mar 15 05:04:20 host.lab.example.com sshd[2255]: pam_unix(sshd:session): session
  opened for user student(uid=1000) by (uid=0)
Mar 15 05:04:20 host.lab.example.com systemd[1]: Starting Hostname Service...
```

**Chapter 12 |** Analyze and Store Logs

```
Mar 15 05:04:20 host.lab.example.com systemd[1]: Started Hostname Service.
Mar 15 05:04:50 host.lab.example.com systemd[1]: systemd-hostnamed.service:
  Deactivated successfully.
Mar 15 05:06:33 host.lab.example.com systemd[2261]: Starting Mark boot as
  successful...
Mar 15 05:06:33 host.lab.example.com systemd[2261]: Finished Mark boot as
  successful.
lines 1996-2043/2043 (END) q
```

Run the following `journalctl` command to list all journal entries from 2022-03-11 20:30:00 to 2022-03-14 10:00:00.

```
[root@host ~]# journalctl --since "2022-03-11 20:30" --until "2022-03-14 10:00"
...output omitted...
```

You can also specify all entries since a relative time to the present. For example, to specify all entries in 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 visible content of the journal, you can view additional log entries if you turn on the verbose output. You can use any displayed extra field to filter the output of a journal query. The verbose output is useful to reduce the output of complex searches for certain events in the journal.

```
[root@host ~]# journalctl -o verbose
Tue 2022-03-15 05:10:32.625470 EDT [s=e7623387430b4c14b2c71917db58e0ee;i...]
 _BOOT_ID=beaadd6e5c5448e393ce716cd76229d4
 _MACHINE_ID=4ec03abd2f7b40118b1b357f479b3112
 PRIORITY=6
 SYSLOG_FACILITY=3
 SYSLOG_IDENTIFIER=systemd
 _UID=0
 _GID=0
 _TRANSPORT=journal
 _CAP_EFFECTIVE=1fffffff
 TID=1
 CODE_FILE=src/core/job.c
 CODE_LINE=744
 CODE_FUNC=job_emit_done_message
 JOB_RESULT=done
 _PID=1
 _COMM=systemd
 _EXE=/usr/lib/systemd/systemd
 _SYSTEMD_CGROUP=/init.scope
 _SYSTEMD_UNIT=init.scope
```

```
_SYSTEMD_SLICE=--.slice
JOB_TYPE=stop
MESSAGE_ID=9d1aaa27d60140bd96365438aad20286
_HOSTNAME=host.lab.example.com
_CMDLINE=/usr/lib/systemd/systemd --switched-root --system --deserialize 31
_SELINUX_CONTEXT=system_u:system_r:init_t:s0
UNIT=user-1000.slice
MESSAGE=Removed slice User Slice of UID 1000.
INVOCATION_ID=0e5efc1b4a6d41198f0cf02116ca8aa8
JOB_ID=3220
_SOURCE_REALTIME_TIMESTAMP=1647335432625470
lines 46560-46607/46607 (END) q
```

The following list shows the common fields of the system journal that you can use to search for relevant lines to a particular process or event:

- `_COMM` is the command name.
- `_EXE` is the path to the executable file for the process.
- `_PID` is the PID of the process.
- `_UID` is the UID of the user that runs the process.
- `_SYSTEMD_UNIT` is the `systemd` unit that started the process.

You can combine multiple system journal fields to form a granular search query with the `journalctl` command. For example, the following `journalctl` command shows all related journal entries to the `sshd.service` `systemd` unit from a process with PID 2110.

```
[root@host ~]# journalctl _SYSTEMD_UNIT=sshd.service _PID=2110
Mar 15 04:42:16 host.lab.example.com sshd[2110]: Accepted
publickey for root from 172.25.250.254 port 46224 ssh2: RSA
SHA256:1UGybTe52L2jzEJa1HLVKn9QUCKrTv3ZxxnMJol1Fro
Mar 15 04:42:16 host.lab.example.com sshd[2110]: pam_unix(sshd:session): session
opened for user root(uid=0) by (uid=0)
```



### Note

For a list of commonly used journal fields, consult 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 Using Log Files* section in the *Red Hat Enterprise Linux 9 Configuring Basic System Settings Guide* at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/index#troubleshooting-problems-using-log-files\\_getting-started-with-system-administration](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration)

## ► Guided Exercise

# Review System Journal Entries

In this exercise, you search the system journal for entries to record events that match specific criteria.

### Outcomes

- Search the system journal for entries to record events based on different criteria.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start logs-systemd
```

### Instructions

- 1. From the `workstation` machine, open an SSH session to the `servera` machine as the student user.

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

- 2. Use the `journalctl` command `_PID=1` option to display only log events that originate from the `systemd` PID 1 process on the `servera` machine. To quit from the `journalctl` command, press `q`. The following output is an example and might differ on your system:

```
[student@servera ~]$ journalctl _PID=1  
Mar 15 04:21:14 localhost systemd[1]: Finished Load Kernel Modules.  
Mar 15 04:21:14 localhost systemd[1]: Finished Setup Virtual Console.  
Mar 15 04:21:14 localhost systemd[1]: dracut ask for additional cmdline parameters  
was skipped because all trigger condition checks failed.  
Mar 15 04:21:14 localhost systemd[1]: Starting dracut cmdline hook...  
Mar 15 04:21:14 localhost systemd[1]: Starting Apply Kernel Variables...  
lines 1-5 q  
[student@servera ~]$
```

- 3. Use the `journalctl` command `_UID=81` option to display all log events that originated from a system service with a PID of 81 on the `servera` machine.

```
[student@servera ~]$ journalctl _UID=81
Mar 15 04:21:17 servera.lab.example.com dbus-broker-lau[727]: Ready
```

- ▶ 4. Use the `journalctl` command `-p warning` option to display log events with warning and higher priority on the `servera` machine.

```
[student@servera ~]$ journalctl -p warning
Mar 15 04:21:14 localhost kernel: wait_for_initramfs() called before
rootfs_initcalls
Mar 15 04:21:14 localhost kernel: ACPI: PRMT not present
Mar 15 04:21:14 localhost kernel: acpi PNP0A03:00: fail to add MMCONFIG
information, can't access extended PCI configuration space under this bridge.
Mar 15 04:21:14 localhost kernel: device-mapper: core: CONFIG_IMA_DISABLE_HTABLE
is disabled. Duplicate IMA measurements will not be recorded in the IMA log.
...output omitted...
Mar 15 04:21:18 servera.lab.example.com NetworkManager[769]: <warn>
[1647332478.5504] device (eth0): mtu: failure to set IPv6 MTU
Mar 15 04:21:27 servera.lab.example.com chrony[751]: System clock wrong by
-0.919695 seconds
Mar 15 04:22:34 servera.lab.example.com chrony[751]: System clock wrong by
0.772805 seconds
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: error:
kex_exchange_identification: Connection closed by remote host
lines 1-19/19 (END) q
[student@servera ~]$
```

- ▶ 5. Display all recorded log events in the past 10 minutes from the current time on the `servera` machine.

```
[student@servera ~]$ journalctl --since "-10min"
Mar 15 05:40:01 servera.lab.example.com anacron[1092]: Job `cron.weekly' started
Mar 15 05:40:01 servera.lab.example.com anacron[1092]: Job `cron.weekly'
terminated
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: error:
kex_exchange_identification: Connection closed by remote host
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: Connection closed by
172.25.250.9 port 45370
Mar 15 05:41:14 servera.lab.example.com sshd[1105]: Accepted publickey for student
from 172.25.250.9 port 45372 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z007ZvufqEixCFCT
+wowZLNzNlBT0
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Created slice User Slice of
UID 1000.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Starting User Runtime
Directory /run/user/1000...
Mar 15 05:41:14 servera.lab.example.com systemd-logind[739]: New session 1 of user
student.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Finished User Runtime
Directory /run/user/1000.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Starting User Manager for UID
1000...
...output omitted...
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped target Sockets.
```

**Chapter 12 |** Analyze and Store Logs

```
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped target Timers.  
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped Mark boot as  
successful after the user session has run 2 minutes.  
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped Daily Cleanup of  
User's Temporary Directories.  
lines 1-48 q  
[student@servera ~]$
```

- 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 on the `servera` machine.

**Note**

Online classrooms typically run on the UTC timezone. To obtain results that start at 9:00AM in your local timezone, adjust your `--since` value by the amount of your offset from UTC. Alternately, ignore the local time and use a value of 9:00 to locate journal entries that occurred since 9:00 for `servera`'s timezone.

```
[student@servera ~]$ journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"  
Mar 15 09:41:14 servera.lab.example.com sshd[1105]: Accepted publickey for student  
from 172.25.250.9 port 45372 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFCT  
+wowZLNzNlBT0  
Mar 15 09:41:15 servera.lab.example.com sshd[1105]: pam_unix(sshd:session):  
session opened for user student(uid=1000) by (uid=0)  
Mar 15 09:44:56 servera.lab.example.com sshd[1156]: Accepted publickey for student  
from 172.25.250.9 port 45374 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFCT  
+wowZLNzNlBT0  
Mar 15 09:44:56 servera.lab.example.com sshd[1156]: pam_unix(sshd:session):  
session opened for user student(uid=1000) by (uid=0)
```

- 7. Return to the `workstation` system as the `student` user.

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

## 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 logs-systemd
```

This concludes the section.

# Preserve the System Journal

## Objectives

Configure the system journal to preserve the record of events when a server is rebooted.

## System Journal Storage

By default, Red Hat Enterprise Linux 9 stores the system journal in the `/run/log` directory, and the system clears the system journal after a reboot. You can change the configuration settings of the `systemd-journald` service in the `/etc/systemd/journald.conf` file so that the journals persist across a reboot.

The `Storage` parameter in the `/etc/systemd/journald.conf` file defines whether to store system journals in a volatile manner or persistently across a reboot. Set this parameter to `persistent`, `volatile`, `auto`, or `none` as follows:

- `persistent`: Stores journals in the `/var/log/journal` directory, which persists across reboots. If the `/var/log/journal` directory does not exist, then the `systemd-journald` service creates it.
- `volatile`: Stores journals in the volatile `/run/log/journal` directory. As the `/run` file system is temporary and exists only in the runtime memory, the data in it, including system journals, does not persist across a reboot.
- `auto`: If the `/var/log/journal` directory exists, then the `systemd-journald` service uses persistent storage; otherwise it uses volatile storage. This action is the default if you do not set the `Storage` parameter.
- `none`: Do not use any storage. The system drops all logs, but you can still forward the logs.

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'
Mar 15 04:21:14 localhost systemd-journald[226]: Runtime Journal (/run/log/
journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 113.3M, 105.3M free.
Mar 15 04:21:19 host.lab.example.com systemd-journald[719]: Runtime Journal (/run/
log/journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 113.3M, 105.3M free.
Mar 15 04:21:19 host.lab.example.com systemd-journald[719]: System Journal (/run/
log/journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 4.0G, 4.0G free.
```

**Note**

In the previous grep command, the vertical bar (|) symbol 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.

## Configure Persistent System Journals

To configure the `systemd-journald` service to preserve system journals persistently across a reboot, set the `Storage` parameter to the `persistent` value in the `/etc/systemd/journald.conf` file. Run your chosen text editor as the superuser to edit the `/etc/systemd/journald.conf` file.

```
[Journal]
Storage=persistent
...output omitted...
```

Restart the `systemd-journald` service to apply the configuration changes.

```
[root@host ~]# systemctl restart systemd-journald
```

If the `systemd-journald` service successfully restarts, then the service creates the `/var/log/journal` directory and it contains one or more subdirectories. These subdirectories have hexadecimal characters in their long names and contain files with the `.journal` extension. The `.journal` binary files store the structured and indexed journal entries.

```
[root@host ~]# ls /var/log/journal
4ec03abd2f7b40118b1b357f479b3112
[root@host ~]# ls /var/log/journal/4ec03abd2f7b40118b1b357f479b3112
system.journal user-1000.journal
```

While the system journals persist after a reboot, the `journalctl` command output includes entries from the current system boot as well as 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 2022-04-13 20:04:32 EDT-Wed 2022-04-13 21:09:36 EDT
-5 6a18... Tue 2022-04-26 08:32:22 EDT-Thu 2022-04-28 16:02:33 EDT
-4 e2d7... Thu 2022-04-28 16:02:46 EDT-Fri 2022-05-06 20:59:29 EDT
-3 45c3... Sat 2022-05-07 11:19:47 EDT-Sat 2022-05-07 11:53:32 EDT
-2 dfae... Sat 2022-05-07 13:11:13 EDT-Sat 2022-05-07 13:27:26 EDT
-1 e754... Sat 2022-05-07 13:58:08 EDT-Sat 2022-05-07 14:10:53 EDT
 0 ee2c... Mon 2022-05-09 09:56:45 EDT-Mon 2022-05-09 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, usually 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 -b -1` command limits the output to only the previous boot.



### References

`systemd-journald.conf(5)`, `systemd-journald(8)` man pages

For more information, refer to the *Troubleshooting Problems Using Log Files* section in the *Red Hat Enterprise Linux 9 Configuring Basic System Settings Guide* at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_basic\\_system\\_settings/index#troubleshooting-problems-using-log-files\\_getting-started-with-system-administration](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration)

## ► Guided Exercise

# Preserve the System Journal

In this exercise, you configure the system journal to preserve its data after a reboot.

### Outcomes

- Configure the system journal to preserve its data after a reboot.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start logs-preserve
```

### Instructions

- 1. From the workstation machine, log in to the servera machine as the student user.

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

- 2. As the superuser, confirm that the `/var/log/journal` directory does not exist. Use the `ls` command to list the `/var/log/journal` directory contents. Use the `sudo` command to elevate the student user privileges. Use `student` as the password if asked.

```
[student@servera ~]$ sudo ls /var/log/journal  
[sudo] password for student: student  
ls: cannot access '/var/log/journal': No such file or directory
```

Because the `/var/log/journal` directory does not exist, the `systemd-journald` service does not preserve the log data after a reboot.

- 3. Configure the `systemd-journald` service on the servera machine to preserve journals after a reboot.

- 3.1. Uncomment the `Storage=auto` line in the `/etc/systemd/journald.conf` file and set the `Storage` parameter to the `persistent` value. You might use the `sudo vim /etc/systemd/journald.conf` command to edit the configuration file. You can type `/Storage=auto` in the `vim` editor command mode to search for the `Storage=auto` line.

```
...output omitted...
[Journal]
Storage=persistent
...output omitted...
```

- 3.2. Restart the `systemd-journald` service to apply the configuration changes.

```
[student@servera ~]$ sudo systemctl restart systemd-journald.service
```

- ▶ 4. Verify that the `systemd-journald` service on the `servera` machine preserves its journals so that they persist after a reboot.

- 4.1. Restart the `servera` machine.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

The SSH connection terminates as soon as you restart the `servera` machine.

- 4.2. Log in to the `servera` machine.

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

- 4.3. Verify that the `/var/log/journal` directory exists. The `/var/log/journal` directory contains a subdirectory with a long hexadecimal name. You can find the journal files in that directory. The subdirectory name on your system might be different.

```
[student@servera ~]$ sudo ls /var/log/journal
[sudo] password for student: student
63b272eae8d5443ca7aaa5593479b25f
[student@servera ~]$ sudo ls /var/log/journal/63b272eae8d5443ca7aaa5593479b25f
system.journal user-1000.journal
```

- 4.4. Return to the `workstation` system as the `student` user.

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

## 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 logs-preserve
```

This concludes the section.

# Maintain Accurate Time

## Objectives

Maintain accurate time synchronization with Network Time Protocol (NTP) and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

## Administer Local Clocks and Time Zones

System time synchronization is critical for log file analysis across multiple systems. Also, some services might require time synchronization to work properly. The Network Time Protocol is a standard way for machines 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 sync with a high-quality hardware clock to serve accurate time 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
```

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 (not always) 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 (US Mountain time) does not have a daylight saving time adjustment at all and is in the time zone `America/Phoenix`.

## Chapter 12 | Analyze and Store Logs

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 time zone setting of the system.

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 2022-03-16 03:05:55 MST
    Universal time: Wed 2022-03-16 10:05:55 UTC
        RTC time: Wed 2022-03-16 10:05:55
        Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
    NTP service: active
    RTC in local TZ: no
```



### Note

If you need to use the Coordinated Universal Time (UTC) on a particular server, then set its time zone to UTC. The `tzselect` command does not include the name of the UTC time zone. Use the `timedatectl set-timezone UTC` command to set the system's current time zone to UTC.

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 time. For example, the following `timedatectl` command changes the time to 09:00:00.

```
[root@host ~]# timedatectl set-time 9:00:00
[root@host ~]# timedatectl
    Local time: Fri 2019-04-05 09:00:27 MST
    Universal time: Fri 2019-04-05 16:00:27 UTC
        RTC time: Fri 2019-04-05 16:00:27
        Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
    NTP service: active
    RTC in local TZ: no
```



### Note

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, as explained after this note.

The `timedatectl` command `set-ntp` option enables or disables NTP synchronization for automatic time adjustment. The option requires either a `true` or `false` argument to turn it 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 9, the `timedatectl set-ntp` command adjusts whether the `chrony` NTP service is enabled. Other Linux distributions might use this setting to adjust a different NTP or *Simple Network Time Protocol (SNTP)* service.

Enabling or disabling NTP with other utilities in Red Hat Enterprise Linux, such as in the graphical GNOME Settings application, also updates this setting.

## Configure and Monitor the `chrony` Service

The `chrony` service keeps on track the usually inaccurate local *Real-Time Clock (RTC)* by synchronizing it to the configured NTP servers. If no network connectivity is available, then the `chrony` service calculates the RTC clock drift, and records it in the file that the `driftfile` value specifies in the `/etc/chrony.conf` configuration file.

By default, the `chrony` service uses servers from the NTP Pool Project to synchronize time and requires no additional configuration. You might need to change the NTP servers for a machine that runs on an isolated network.

The *stratum* of the NTP time source determines its quality. The stratum determines the number of hops 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, while a machine that synchronizes time from the NTP server is a *stratum 2* time source.

The server and peer are the two categories of time sources that you can declare in the `/etc/chrony.conf` configuration file. The server is one stratum above the local NTP server, and the peer is at the same stratum level. You can define multiple servers and peers in the `chrony` configuration file, one per line.

The first argument of the `server` line is the IP address or DNS name of the NTP server. Following the server IP address or name, you can list a series of options for the server. Red Hat recommends to use the `iburst` option, because then the `chrony` service takes four measurements in a short time period for a more accurate initial clock synchronization after the service starts. Use the `man 5 chrony.conf` command for more information about the `chrony` configuration file options.

As an example, with the following `server classroom.example.com iburst` line in the `/etc/chrony.conf` configuration file, the `chrony` service uses the `classroom.example.com` server as the NTP time source.

```
# Use public servers from the pool.ntp.org project.
...output omitted...
server classroom.example.com iburst
...output omitted...
```

Restart the service after pointing the `chrony` service to the `classroom.example.com` local time source.

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

The `chronyc` command acts as a client to the `chrony` service. After setting up NTP synchronization, verify that the local system is seamlessly using the NTP server to synchronize the

system clock by using the `chronyc sources` command. For more verbose output with additional explanations about the output, use the `chronyc sources -v` command.

```
[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
||           Reachability register (octal) -.      |     xxxx = adjusted offset,
||           Log2(Polling interval) --.   |     |     yyyy = measured offset,
||           \    |     |     zzzz = estimated error.
||           |    |
MS Name/IP address       Stratum Poll Reach LastRx Last sample
=====
^* 172.25.254.254        3      6     17    26  +2957ns[+2244ns] +/-  25ms
```

The asterisk character (\*) in the S (Source state) field indicates that the `chronyd` service uses the `classroom.example.com` server as a time source and is the NTP server that the machine is currently synchronized to.



### References

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

### NTP Pool Project

<http://www.ntppool.org/>

### Time Zone Database

<http://www.iana.org/time-zones>

## ► Guided Exercise

# Maintain Accurate Time

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

## Outcomes

- Change the time zone on a server.
- Configure the server to synchronize its time with an NTP time source.

## Before You Begin

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start logs-maintain
```

## Instructions

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

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

- 2. For this practice, pretend that the **servera** machine is relocated to Haiti and you need to update the time zone. Elevate the privileges of the **student** user to run the **timedatectl** command to update the time zone.

- 2.1. Select the appropriate time zone for Haiti.

```
[student@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: Wed Mar 16 07:10:35 EDT 2022.  
 Universal Time is now: Wed Mar 16 11:10:35 UTC 2022.  
 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 time zone on the servera machine to America/Port-au-Prince.

```
[student@servera ~]$ sudo timedatectl set-timezone \
America/Port-au-Prince
[sudo] password for student: student
```

## 2.3. Verify that you correctly set the time zone to America/Port-au-Prince.

```
[student@servera ~]$ timedatectl
    Local time: Wed 2022-03-16 07:13:25 EDT
    Universal time: Wed 2022-03-16 11:13:25 UTC
        RTC time: Wed 2022-03-16 11:13:24
        Time zone: America/Port-au-Prince (EDT, -0400)
System clock synchronized: no
          NTP service: inactive
      RTC in local TZ: no
```

- 3. Configure the `chronyd` service on the `servera` machine 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. The following output shows the configuration line to add to the configuration file, which includes the `iburst` option to speed up initial time synchronization:

```
...output omitted...
server classroom.example.com iburst
...output omitted...
```

- 3.2. Enable time synchronization on the `servera` machine. The command activates the NTP server with the settings from the `/etc/chrony.conf` configuration file. That command might activate either the `chronyd` or the `ntpd` service, depending on which service is currently installed on the system.

```
[student@servera ~]$ sudo timedatectl set-ntp true
```

- 4. Verify that the `servera` machine configuration synchronizes with the `classroom.example.com` time source in the classroom environment.
- 4.1. Verify that time synchronization is enabled on the `servera` machine.



### 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.

```
[student@servera ~]$ timedatectl
    Local time: Wed 2022-03-16 07:24:13 EDT
    Universal time: Wed 2022-03-16 11:24:13 UTC
        RTC time: Wed 2022-03-16 11:24:13
        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 (S) field for the classroom.example.com NTP time source. The asterisk indicates that the local system time successfully synchronizes with the NTP time source.

```
[student@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
||   Reachability register (octal) -.      |   xxxx = adjusted offset,
||   Log2(Polling interval) --.    |   |   yyyy = measured offset,
||           \   |   |   zzzz = estimated error.
||           |   |   \
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
^* 172.25.254.254            2     6   377    33    +84us[ +248us] +/-   21ms
```

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

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

## 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 logs-maintain
```

This concludes the section.

## ▶ Lab

# Analyze and Store Logs

In this lab, you change the time zone on an existing server and configure a new log file for all events for authentication failures.

## Outcomes

- Update the time zone on an existing server.
- Configure a new log file to store all messages for authentication failures.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. Log in to the `serverb` machine as the `student` user.
2. Pretend that the `serverb` machine is relocated to Jamaica and that you must update the time zone. Verify that you correctly set the appropriate time zone.
3. View the recorded log events in the previous 30 minutes on the `serverb` machine.
4. Create the `/etc/rsyslog.d/auth-errors.conf` file. Configure the `rsyslog` service to write authentication and security messages to the `/var/log/auth-errors` file. Use the `authpriv` facility and the `alert` priority.

## 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, 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 logs-review
```

This concludes the section.

## ► Solution

# Analyze and Store Logs

In this lab, you change the time zone on an existing server and configure a new log file for all events for authentication failures.

## Outcomes

- Update the time zone on an existing server.
- Configure a new log file to store all messages for authentication failures.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. Log in to the `serverb` machine as the `student` user.

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

2. Pretend that the `serverb` machine is relocated to Jamaica and that you must update the time zone. Verify that you correctly set the appropriate time zone.

- 2.1. 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: Wed Mar 16 07:17:15 EST 2022.

Universal Time is now: Wed Mar 16 12:17:15 UTC 2022.

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

2.2. Elevate the student user privileges to update the time zone of the serverb server to America/Jamaica.

```
[student@serverb ~]$ sudo timedatectl set-timezone America/Jamaica
[sudo] password for student: student
```

2.3. Verify that you successfully set the time zone to America/Jamaica.

```
[student@serverb ~]$ timedatectl
    Local time: Wed 2022-03-16 07:18:40 EST
    Universal time: Wed 2022-03-16 12:18:40 UTC
        RTC time: Wed 2022-03-16 12:18:40
      Time zone: America/Jamaica (EST, -0500)
System clock synchronized: yes
          NTP service: active
    RTC in local TZ: no
```

3. View the recorded log events in the previous 30 minutes on the serverb machine.

- 3.1. Determine the time frame to view the journal entries.

```
[student@serverb ~]$ date
Wed Mar 16 07:19:29 AM EST 2022
[student@serverb ~]$ date -d "-30 minutes"
Wed Mar 16 06:49:38 AM EST 2022
```

- 3.2. View the recorded log events in the previous 30 minutes on the serverb machine.

```
[student@serverb ~]$ journalctl --since 06:49:00 --until 07:19:00
...output omitted...
Mar 16 07:10:58 localhost kernel: x86/PAT: Configuration [0-7]: WB WC UC- UC WB
    WP UC- WT
Mar 16 07:10:58 localhost kernel: found SMP MP-table at [mem
    0x000f5bd0-0x000f5bdf]
Mar 16 07:10:58 localhost kernel: Using GB pages for direct mapping
Mar 16 07:10:58 localhost kernel: RAMDISK: [mem 0x2e0d9000-0x33064fff]
Mar 16 07:10:58 localhost kernel: ACPI: Early table checksum verification disabled
Mar 16 07:10:58 localhost kernel: ACPI: RSDP 0x00000000000F5B90 000014 (v00
    BOCHS )
Mar 16 07:10:58 localhost kernel: ACPI: RSDT 0x000000007FFE12C4 00002C (v01 BOCHS
    BXPCRSDT 00000001 BXPC 00000001)
Mar 16 07:10:58 localhost kernel: ACPI: FACP 0x000000007FFE11D0 000074 (v01 BOCHS
    BXPCFACP 00000001 BXPC 00000001)
Mar 16 07:10:58 localhost kernel: ACPI: DSDT 0x000000007FFDFDC0 001410 (v01 BOCHS
    BXPCDSDT 00000001 BXPC 00000001)
lines 1-50/50 q
[student@serverb ~]$
```

4. Create the /etc/rsyslog.d/auth-errors.conf file. Configure the rsyslog service to write authentication and security messages to the /var/log/auth-errors file. Use the authpriv facility and the alert priority.

- 4.1. Create the /etc/rsyslog.d/auth-errors.conf file and specify the new /var/log/auth-errors file as the destination for authentication and security messages.

```
authpriv.alert /var/log/auth-errors
```

- 4.2. Restart the rsyslog service to apply the configuration file changes.

```
[student@serverb ~]$ sudo systemctl restart rsyslog
```

- 4.3. Write a example log message to the /var/log/auth-errors file.

```
[student@serverb ~]$ logger -p authpriv.alert "Logging test authpriv.alert"
```

- 4.4. Verify that the /var/log/auth-errors file contains the log entry with the Logging test authpriv.alert message.

```
[student@serverb ~]$ sudo tail /var/log/auth-errors  
Mar 16 07:25:12 serverb student[1339]: Logging test authpriv.alert
```

- 4.5. Return to the workstation system 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 logs-review
```

## 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 logs-review
```

This concludes the section.

# Summary

---

- The `systemd-journald` and `rsyslog` services capture and write log messages to the appropriate files.
- The `/var/log` directory contains log files.
- Periodic rotation of log files prevents them from filling up the file-system space.
- The `systemd` journals are temporary and do not persist across a reboot.
- The `chrony` service helps to synchronize time settings with a time source.
- You can update the time zone of the server based on its location.

## Chapter 13

# Manage Networking

### Goal

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

### Objectives

- Test and inspect the current network configuration with command-line utilities.
- Manage network settings and devices with the nmcli command.
- Modify network configuration by editing configuration files.
- Configure a server's static hostname and its name resolution and test the results.

### Sections

- Validate Network Configuration (and Guided Exercise)
- Configure Networking from the Command Line (and Guided Exercise)
- Edit Network Configuration Files (and Guided Exercise)
- Configure Hostnames and Name Resolution (and Guided Exercise)

### Lab

- Manage Networking

# Validate Network Configuration

---

## Objectives

Test and inspect the current network configuration with command-line utilities.

## Gather Network Interface Information

The `ip link` command lists all available network interfaces on your system. In the following example, the server has three network interfaces: `lo`, which is the loopback device that is connected to the server itself, and two Ethernet interfaces, `ens3` and `ens4`.

```
[user@host ~]$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
    group default qlen 1000
        link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
    group default qlen 1000
        link/ether 52:54:00:00:00:1e brd ff:ff:ff:ff:ff:ff
```

To configure a network interface correctly, you must know which interface is connected to which network. In many cases, you can obtain the MAC address of the interface that is connected to each network, either because it is physically printed on the card or server, or because it is a virtual machine and you know how it is configured. The MAC address of the device is listed after `link/ether` for each interface. So you know that the network card with the MAC address `52:54:00:00:00:0a` is the network interface `ens3`.

## Display IP Addresses

Use the `ip` command to view device and address information. A single network interface can have multiple IPv4 or IPv6 addresses.

```
[user@host ~]$ ip addr show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000①
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff②
        inet 192.0.2.2/24 brd 192.0.2.255 scope global ens3③
            valid_lft forever preferred_lft forever
        inet6 2001:db8:0:1:5054:ff:fe00:b/64 scope global④
            valid_lft forever preferred_lft forever
        inet6 fe80::5054:ff:fe00:b/64 scope link⑤
            valid_lft forever preferred_lft forever
```

① An active interface is UP.

② The `link/ether` string specifies the hardware (MAC) address of the device.

- ③ The `inet` string shows an IPv4 address, its network prefix length, and scope.
- ④ The `inet6` string shows an IPv6 address, its network prefix length, and scope. This address is of *global* scope and is normally used.
- ⑤ This `inet6` string shows that the interface has an IPv6 address of *link* scope that can be used only for communication on the local Ethernet link.

## Display Performance Statistics

The `ip` command can also show statistics about network performance. Counters for each network interface can identify the presence of network issues. The counters record statistics, such as for the number of received (RX) and transmitted (TX) packets, packet errors, and dropped packets.

```
[user@host ~]$ ip -s link show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
        RX: bytes   packets   errors   dropped overrun mcast
        269850      2931       0        0        0        0
        TX: bytes   packets   errors   dropped carrier collsns
        300556      3250       0        0        0        0
```

## Verify Connectivity Between Hosts

The `ping` command tests connectivity. The command continues to run until `Ctrl+c` is pressed unless options are given to limit the number of sent packets.

```
[user@host ~]$ ping -c3 192.0.2.254
PING 192.0.2.1 (192.0.2.254) 56(84) bytes of data.
64 bytes from 192.0.2.254: icmp_seq=1 ttl=64 time=4.33 ms
64 bytes from 192.0.2.254: icmp_seq=2 ttl=64 time=3.48 ms
64 bytes from 192.0.2.254: icmp_seq=3 ttl=64 time=6.83 ms

--- 192.0.2.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.485/4.885/6.837/1.424 ms
```

The `ping6` command is the IPv6 version of the `ping` command in Red Hat Enterprise Linux. The difference between these commands is that the `ping6` command communicates over IPv6 and takes IPv6 addresses.

```
[user@host ~]$ ping6 2001:db8:0:1::1
PING 2001:db8:0:1::1(2001:db8:0:1::1) 56 data bytes
64 bytes from 2001:db8:0:1::1: icmp_seq=1 ttl=64 time=18.4 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=3 ttl=64 time=0.180 ms
^C
--- 2001:db8:0:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.178/6.272/18.458/8.616 ms
[user@host ~]$
```

## Chapter 13 | Manage Networking

When you ping the link-local addresses and the link-local all-nodes multicast group (`ff02::1`), the network interface to use must be specified explicitly with a scope zone identifier (such as `ff02::1%ens3`). If this network interface is omitted, then the `connect: Invalid argument` error is displayed.

You can use the `ping6 ff02::1` command to find other IPv6 nodes on the local network.

```
[user@host ~]$ ping6 ff02::1%ens4
PING ff02::1%ens4(fe02::1) 56 data bytes
64 bytes from fe80::78cf:ffff:fed2:f97b: icmp_seq=1 ttl=64 time=22.7 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=30.1 ms (DUP!)
64 bytes from fe80::78cf:ffff:fed2:f97b: icmp_seq=2 ttl=64 time=0.183 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=2 ttl=64 time=0.231 ms (DUP!)
^C
--- ff02::1%ens4 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.183/13.320/30.158/13.374 ms
[user@host ~]$
[user@host ~]$ ping6 -c 1 fe80::f482:dbff:fe25:6a9f%ens4
PING fe80::f482:dbff:fe25:6a9f%ens4(fe80::f482:dbff:fe25:6a9f) 56 data bytes
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=22.9 ms

--- fe80::f482:dbff:fe25:6a9f%ens4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 22.903/22.903/22.903/0.000 ms
```

Other hosts on the same link can use IPv6 link-local addresses, like normal addresses.

```
[user@host ~]$ ssh fe80::f482:dbff:fe25:6a9f%ens4
user@fe80::f482:dbff:fe25:6a9f%ens4's password:
Last login: Thu Jun  5 15:20:10 2014 from host.example.com
[user@server ~]$
```

## Troubleshoot Router Issues

Network routing is complex, and sometimes traffic does not behave as you might expect. You can use different tools to diagnose router issues.

### Describe the Routing Table

Use the `ip` command `route` option to show routing information.

```
[user@host ~]$ ip route
default via 192.0.2.254 dev ens3 proto static metric 1024
192.0.2.0/24 dev ens3 proto kernel scope link src 192.0.2.2
10.0.0.0/8 dev ens4 proto kernel scope link src 10.0.0.11
```

All packets that are destined for the `10.0.0.0/8` network are sent directly to the destination through the `ens4` device. All packets that are destined for the `192.0.2.0/24` network are sent directly to the destination through the `ens3` device. All other packets are sent to the default router at `192.0.2.254`, and also through device `ens3`.

Use the `ip` command `-6` option to show the IPv6 routing table.

```
[user@host ~]$ ip -6 route
unreachable ::/96 dev lo metric 1024 error -101
unreachable ::ffff:0.0.0.0/96 dev lo metric 1024 error -101
2001:db8:0:1::/64 dev ens3 proto kernel metric 256
unreachable 2002:a00::/24 dev lo metric 1024 error -101
unreachable 2002:7f00::/24 dev lo metric 1024 error -101
unreachable 2002:a9fe::/32 dev lo metric 1024 error -101
unreachable 2002:ac10::/28 dev lo metric 1024 error -101
unreachable 2002:c0a8::/32 dev lo metric 1024 error -101
unreachable 2002:e000::/19 dev lo metric 1024 error -101
unreachable 3ffe:ffff::/32 dev lo metric 1024 error -101
fe80::/64 dev ens3 proto kernel metric 256
default via 2001:db8:0:1::ffff dev ens3 proto static metric 1024
```

1. The 2001:db8:0:1::/64 network uses the ens3 interface (which presumably has an address on that network).
2. The fe80::/64 network uses the ens3 interface, for the link-local address. On a system with multiple interfaces, a route to the fe80::/64 network exists in each interface for each link-local address.
3. The default route to all networks on the IPv6 Internet (the ::/0 network) uses the router at the 2001:db8:0:1::ffff network and it is reachable with the ens3 device.

## Trace Traffic Routes

To trace the network traffic path to reach a remote host through multiple routers, use either the **traceroute** or **tracepath** command. These commands can identify issues with one of your routers or an intermediate router. Both commands use UDP packets to trace a path by default; however, many networks block UDP and ICMP traffic. The **traceroute** command has options to trace the path with UDP (default), ICMP (-I), or TCP (-T) packets. Typically, the **traceroute** command is not installed by default.

```
[user@host ~]$ tracepath access.redhat.com
...output omitted...
4: 71-32-28-145.rcmt.qwest.net          48.853ms asymm 5
5: dcp-brdr-04.inet.qwest.net           100.732ms asymm 7
6: 206.111.0.153.ptr.us.xo.net         96.245ms asymm 7
7: 207.88.14.162.ptr.us.xo.net         85.270ms asymm 8
8: ae1d0.cir1.atlanta6-ga.us.xo.net    64.160ms asymm 7
9: 216.156.108.98.ptr.us.xo.net        108.652ms
10: bu-ether13.atlngamq46w-bcr00.tbone.rr.com 107.286ms asymm 12
...output omitted...
```

Each line in the output of the **tracepath** command represents a router or hop that the packet passes through between the source and the final destination. The command outputs information for each hop as it becomes available, including the *round trip timing (RTT)* and any changes in the *maximum transmission unit (MTU)* size. The **asymm** indication means that the traffic that reached the router returned from that router by different (*asymmetric*) routes. These routers here are for outbound traffic, not for return traffic.

The **tracepath6** and **traceroute -6** commands are the equivalent IPv6 commands to the **tracepath** and **traceroute** commands.

```
[user@host ~]$ tracepath 2001:db8:0:2::451
 1?: [LOCALHOST]                                0.091ms pmtu 1500
 1: 2001:db8:0:1::ba                          0.214ms
 2: 2001:db8:0:1::1                           0.512ms
 3: 2001:db8:0:2::451                         0.559ms reached
Resume: pmtu 1500 hops 3 back 3
```

## Troubleshoot Port and Service Issues

TCP services use sockets as endpoints for communication and are composed of an IP address, protocol, and port number. Services typically listen on standard ports, while clients use a random available port. Well-known names for standard ports are listed in the `/etc/services` file.

The `ss` command is used to display socket statistics. The `ss` command replaces the older `netstat` tool, from the `net-tools` package, which might be more familiar to some system administrators but is not always installed.

```
[user@host ~]$ ss -ta
State      Recv-Q Send-Q      Local Address:Port          Peer Address:Port
LISTEN     0       128           *:sunrpc                  *:*
LISTEN     0       128           *:ssh                     *:* ①
LISTEN     0       100          127.0.0.1:smtp            :②
LISTEN     0       128           *:36889                  *:*
ESTAB      0       0             172.25.250.10:ssh        172.25.254.254:59392 ③
LISTEN     0       128           :::sunrpc                :::*
LISTEN     0       128           :::ssh                   ::*:④
LISTEN     0       100          ::1:smtp                 ::*:⑤
LISTEN     0       128           :::34946                  :::*
```

- ① **\*:ssh**: The port that is used for SSH is listening on all IPv4 addresses. The asterisk (\*) character represents *all* when referencing IPv4 addresses or ports.
- ② **127.0.0.1:smtp**: The port that is used for SMTP is listening on the 127.0.0.1 IPv4 loopback interface.
- ③ **172.25.250.10:ssh**: The established SSH connection is on the 172.25.250.10 interface and originates from a system with an address of 172.25.254.254.
- ④ **:::ssh**: The port that is used for SSH is listening on all IPv6 addresses. The double colon (::) syntax represents all IPv6 interfaces.
- ⑤ **::1:smtp**: The port that is used for SMTP is listening on the ::1 IPv6 loopback interface.

### Options for ss and netstat

Option	Description
-n	Show numbers instead of names for interfaces and ports.
-t	Show TCP sockets.
-u	Show UDP sockets.
-l	Show only listening sockets.

Option	Description
-a	Show all (listening and established) sockets.
-p	Show the process that uses the sockets.
-A inet	Display active connections (but not listening sockets) for the <code>inet</code> address family. That is, ignore local UNIX domain sockets. For the <code>ss</code> command, both IPv4 and IPv6 connections are displayed. For the <code>netstat</code> command, only IPv4 connections are displayed. (The <code>netstat -A inet6</code> command displays IPv6 connections, and the <code>netstat -46</code> command displays IPv4 and IPv6 at the same time.)



## References

`ip-link(8)`, `ip-address(8)`, `ip-route(8)`, `ip(8)`, `ping(8)`, `tracepath(8)`, `traceroute(8)`, `ss(8)`, and `netstat(8)` man pages

For more information, refer to the *Configuring and Managing Networking Guide* at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_and\\_managing\\_networking/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/index)

## ► Guided Exercise

# Validate Network Configuration

In this exercise, you inspect the network configuration of one of your servers.

## Outcomes

- Identify the current network interfaces and basic network addresses.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start net-validate
```

## Instructions

- 1. Use the `ssh` command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication and passwordless access to `servera`.

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

- 2. Locate the network interface name that is associated with the Ethernet address `52:54:00:00:fa:0a`. Record or remember this name and use it to replace the `enX` placeholder in subsequent commands.



### Important

Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names will vary according to the course platform and hardware in use.

On your system, locate the interface name (such as `ens06` or `en1p2`) that is associated with the Ethernet address `52:54:00:00:fa:0a`. Use this interface name to replace the `enX` placeholder that is used throughout this exercise.

```
[student@servera ~]$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
```

- 3. Display the current IP address and netmask for all interfaces.

```
[student@servera ~]$ ip -br addr
lo          UP      127.0.0.1/8 ::1/128
enX:        UP      172.25.250.10/24 fe80::3059:5462:198:58b2/64
```

- 4. Display the statistics for the enX interface.

```
[student@servera ~]$ ip -s link show enX
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        RX: bytes   packets   errors   dropped overrun mcast
            89014225    168251      0       154418      0       0
        TX: bytes   packets   errors   dropped carrier collsns
            608808     6090      0       0       0       0
```

- 5. Display the route information.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev enX proto static metric 100
172.25.250.0/24 dev enX proto kernel scope link src 172.25.250.10 metric 100
```

- 6. Verify that the router is accessible.

```
[student@servera ~]$ ping -c3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.196 ms
64 bytes from 172.25.250.254: icmp_seq=2 ttl=64 time=0.436 ms
64 bytes from 172.25.250.254: icmp_seq=3 ttl=64 time=0.361 ms

--- 172.25.250.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 49ms
rtt min/avg/max/mdev = 0.196/0.331/0.436/0.100 ms
```

- 7. Show all the hops between the local system and classroom.example.com.

```
[student@servera ~]$ tracepath classroom.example.com
1?: [LOCALHOST]                                pmtu 1500
1:  bastion.lab.example.com                   0.337ms
1:  bastion.lab.example.com                   0.122ms
2:  172.25.254.254                           0.602ms reached
Resume: pmtu 1500 hops 2 back 2
```

- 8. Display the listening TCP sockets on the local system.

```
[student@servera ~]$ ss -lt
State      Recv-Q Send-Q      Local Address:Port      Peer Address:Port
LISTEN      0      128          0.0.0.0:sunrpc      0.0.0.0:*
LISTEN      0      128          0.0.0.0:ssh       0.0.0.0:*
LISTEN      0      128          [:]:sunrpc        [:]:*
LISTEN      0      128          [:]:ssh           [:]:*
```

- 9. Return to the workstation system as the student user.

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

## 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 net-validate
```

This concludes the section.

# Configure Networking from the Command Line

## Objectives

Manage network settings and devices with the `nmcli` command.

## Describe the NetworkManager Service

The NetworkManager service monitors and manages a system's network settings. In the GNOME graphical environment, a Notification Area applet displays network configuration and status information that is received from the NetworkManager daemon. You can interact with the NetworkManager service via the command line or with graphical tools. Service configuration files are stored in the `/etc/NetworkManager/system-connections/` directory.

The NetworkManager service manages network *devices* and *connections*. A *device* is a physical or virtual network interface that provides for network traffic. A *connection* is a collection of related configuration settings for a single network device. A connection can also be known as a *network profile*. Each connection must have a unique name or ID, which can match the device name that it configures.

A single device can have multiple connection configurations and switch between them, but only one connection can be active per device. For example, a laptop wireless device might configure a fixed IP address for use at a secure work site in a connection, but might configure a second connection with an automated address and a virtual private network (VPN) to access the same company network from home.



### Important

Starting in Red Hat Enterprise Linux 8, `ifcfg` format configuration files and the `/etc/sysconfig/network-scripts/` directory are deprecated.

NetworkManager now uses an INI-style key file format, which is a key-value pair structure to organize properties. NetworkManager stores network profiles in the `/etc/NetworkManager/system-connections/` directory. For compatibility with earlier versions, `ifcfg` format connections in the `/etc/sysconfig/network-scripts/` directory are still recognized and loaded.

## View Network Information

Use the `nmcli` utility to create and edit connection files from the command line. The `nmcli device status` command displays the status of all network devices:

```
[user@host ~]$ nmcli dev status
DEVICE  TYPE      STATE       CONNECTION
eno1    ethernet  connected   eno1
ens3    ethernet  connected   static-ens3
en02    ethernet  disconnected --
lo      loopback  unmanaged  --
```

**Note**

You can abbreviate nmcli objects and actions. For example, you can abbreviate nmcli device disconnect as nmcli dev dis and nmcli connection modify as nmcli con mod. The abbreviation can be as short as a single letter, but must use enough characters to uniquely identify the object to manage.

The nmcli connection show command displays a list of all connections. Use the --active option to list only active connections.

```
[user@host ~]$ nmcli con show
NAME      UUID                                  TYPE      DEVICE
eno2      ff9f7d69-db83-4fed-9f32-939f8b5f81cd 802-3-ethernet --
static-ens3 72ca57a2-f780-40da-b146-99f71c431e2b 802-3-ethernet ens3
eno1      87b53c56-1f5d-4a29-a869-8a7bdaf56dfa 802-3-ethernet eno1
[user@host ~]$ nmcli con show --active
NAME      UUID                                  TYPE      DEVICE
static-ens3 72ca57a2-f780-40da-b146-99f71c431e2b 802-3-ethernet ens3
eno1      87b53c56-1f5d-4a29-a869-8a7bdaf56dfa 802-3-ethernet eno1
```

## Add a Network Connection

Use the nmcli connection add command to add network connections.

The following example adds a connection for the eno2 interface called eno2. The network information for the connection uses a DHCP service and has the device autoconnect on startup. The system also obtains IPv6 networking settings by listening for router advertisements on the local link. The configuration file name contains the value of the nmcli command con-name parameter, which is eno2. The value of the con-name parameter is saved to the /etc/NetworkManager/system-connections/eno2.nmconnection file.

```
[root@host ~]# nmcli con add con-name eno2 \
type ethernet iface eno2
Connection 'eno2' (8159b66b-3c36-402f-aa4c-2ea933c7a5ce) successfully added
```

The next example creates the eno3 connection for the eno3 device with a static IPv4 network setting. This command configures the 192.168.0.5 IP address with a network prefix of /24 and a network gateway of 192.168.0.254. The nmcli connection add command fails if the connection name that you try to add exists.

```
[root@host ~]# nmcli con add con-name eno3 type ethernet iface eno3 \
ipv4.addresses 192.168.0.5/24 ipv4.gateway 192.168.0.254
```

The next example creates an eno4 connection for the eno4 device with static IPv6 and IPv4 addresses. This command configures the 2001:db8:0:1::c000:207 IPv6 address with the /64 network prefix and the 2001:db8:0:1::1 address as the default gateway. The command also configures the 192.0.2.7 IPv4 address with the /24 network prefix and the 192.0.2.1 address as the default gateway.

```
[root@host ~]# nmcli con add con-name eno4 type ethernet ifname eno4 \
    ipv6.addresses 2001:db8:0:1::c000:207/64 ipv6.gateway 2001:db8:0:1::1 \
    ipv4.addresses 192.0.2.7/24 ipv4.gateway 192.0.2.1
```

## Manage Network Connections

The `nmcli connection up` command activates a network connection on the device that it is bound to. Activating a network connection requires the connection name, not the device name.

```
[user@host ~]$ nmcli con show
NAME           UUID                                  TYPE      DEVICE
static-ens3    72ca57a2-f780-40da-b146-99f71c431e2b 802-3-ethernet --
static-ens5    87b53c56-1f5d-4a29-a869-8a7bdaf56dfa 802-3-ethernet --
[root@host ~]# nmcli con up static-ens3
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/2)
```

The `nmcli device disconnect` command disconnects the network device and brings down the connection.

```
[root@host ~]# nmcli dev disconnect ens3
```



### Important

Use `nmcli device disconnect` to stop traffic on a network interface and deactivate the connection.

Because most connections enable the `autoconnect` parameter, the `nmcli connection down` command is ineffective for stopping traffic. Although the connection deactivates, `autoconnect` immediately reactivates the connection if the device is up and available. `Autoconnect` is a desired behavior because it maintains connections through temporary network outages.

By disconnecting the device under the connection, the connection is forced to be down until the device is connected again.

## Update Network Connection Settings

`NetworkManager` service connections have two kinds of settings. Static connection properties are configured by the administrator and stored in the `/etc/NetworkManager/system-connections/*.nmconnection` configuration files. Dynamic connection properties are requested from a DHCP server and are not stored persistently.

To list the current settings for a connection, use the `nmcli connection show` command. Settings in lowercase are static properties that the administrator can change. Settings in uppercase are active settings in temporary use for this connection instance.

```
[root@host ~]# nmcli con show static-ens3
connection.id:                      static-ens3
connection.uuid:                     87b53c56-1f5d-4a29-a869-8a7bdaf56dfa
connection.interface-name:          --
connection.type:                     802-3-ethernet
```

```

connection.autoconnect: yes
connection.timestamp: 1401803453
connection.read-only: no
connection.permissions:
connection.zone: --
connection.master: --
connection.slave-type: --
connection.secondaries:
connection.gateway-ping-timeout: 0
802-3-ethernet.port: --
802-3-ethernet.speed: 0
802-3-ethernet.duplex: --
802-3-ethernet.auto-negotiate: yes
802-3-ethernet.mac-address: CA:9D:E9:2A:CE:F0
802-3-ethernet.cloned-mac-address: --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu: auto
802-3-ethernet.s390-subchannels:
802-3-ethernet.s390-nettype: --
802-3-ethernet.s390-options:
ipv4.method: manual
ipv4.dns: 192.168.0.254
ipv4.dns-search: example.com
ipv4.addresses: { ip = 192.168.0.2/24,
                  gw = 192.168.0.254 }

ipv4.routes:
ipv4.ignore-auto-routes: no
ipv4.ignore-auto-dns: no
ipv4.dhcp-client-id: --
ipv4.dhcp-send-hostname: yes
ipv4.dhcp-hostname: --
ipv4.never-default: no
ipv4.may-fail: yes
ipv6.method: manual
ipv6.dns: 2001:4860:4860::8888
ipv6.dns-search: example.com
ipv6.addresses: { ip = 2001:db8:0:1::7/64,
                  gw = 2001:db8:0:1::1 }

ipv6.routes:
ipv6.ignore-auto-routes: no
ipv6.ignore-auto-dns: no
ipv6.never-default: no
ipv6.may-fail: yes
ipv6.ip6-privacy: -1 (unknown)
ipv6.dhcp-hostname: --
...output omitted...

```

Use the `nmcli connection modify` command to update connection settings. These changes are saved in the `/etc/NetworkManager/system-connections/name.nmconnection` file. Consult the `nm-settings(5)` man page for the available settings.

Use the following command to update the `static-ens3` connection to set the `192.0.2.2/24` IPv4 address and the `192.0.2.254` default gateway. Use the `nmcli` command `connection.autoconnect` parameter to automatically enable or disable the connection at system boot.

```
[root@host ~]# nmcli con mod static-ens3 ipv4.addresses 192.0.2.2/24 \
    ipv4.gateway 192.0.2.254 connection.autoconnect yes
```

Use the following command to update the `static-ens3` connection to set the `2001:db8:0:1::a00:1/64` IPv6 address and the `2001:db8:0:1::1` default gateway.

```
[root@host ~]# nmcli con mod static-ens3 ipv6.addresses 2001:db8:0:1::a00:1/64 \
    ipv6.gateway 2001:db8:0:1::1
```



### Important

To change a DHCP connection configuration to be static, update the `ipv4.method` setting from `auto` or `dhcp` to `manual`. For an IPv6 connection, update the `ipv6.method` setting. If the method is not set correctly, then the connection might hang or be incomplete when activated, or it might obtain an address from DHCP or SLAAC in addition to the configured static address.

Some settings can have multiple values. A specific value can be added to the list or deleted from the connection settings by adding a plus (+) or minus (-) symbol to the start of the setting name. If a plus or minus is not included, then the specified value replaces the setting's current list. The following example adds the `2.2.2.2` DNS server to the `static-ens3` connection.

```
[root@host ~]# nmcli con mod static-ens3 +ipv4.dns 2.2.2.2
```

You can also modify network profiles by editing the connection's configuration file in `/etc/NetworkManager/system-connections/`. While `nmcli` commands communicate directly with `NetworkManager` to implement modifications immediately, connection file edits are not implemented until `NetworkManager` is asked to reload the configuration file. With manual editing, you can create complex configurations in steps, and then load the final configuration when ready. The following example loads all connection profiles.

```
[root@host ~]# nmcli con reload
```

The next example loads only the `eno2` connection profile at `/etc/NetworkManager/system-connections/eno2.nmconnection`.

```
[root@host ~]# nmcli con reload eno2
```

## Delete a Network Connection

The `nmcli connection delete` command deletes a connection from the system. This command disconnects the device and removes the connection configuration file.

```
[root@host ~]# nmcli con del static-ens3
```

## Permissions to Modify NetworkManager Settings

The `root` user can use the `nmcli` command to change the network configuration.

Non-privileged users that are logged in on the physical or virtual console can also make most network configuration changes. If a person is on the system's console, then the system is likely being used as a workstation or laptop where the user needs to configure, activate, and deactivate connections. Non-privileged users that log in with ssh must switch to the root user to change network settings.

Use the `nmcli general permissions` command to view your current permissions. The following example lists the root user's NetworkManager permissions.

```
[root@host ~]# nmcli gen permissions
PERMISSION                                     VALUE
org.freedesktop.NetworkManager.checkpoint-rollback      yes
org.freedesktop.NetworkManager.enable-disable-connectivity-check  yes
org.freedesktop.NetworkManager.enable-disable-network    yes
org.freedesktop.NetworkManager.enable-disable-statistics yes
org.freedesktop.NetworkManager.enable-disable-wifi       yes
org.freedesktop.NetworkManager.enable-disable-wimax     yes
org.freedesktop.NetworkManager.enable-disable-wwan     yes
org.freedesktop.NetworkManager.network-control          yes
org.freedesktop.NetworkManager.reload                  yes
org.freedesktop.NetworkManager.settings.modify.global-dns yes
org.freedesktop.NetworkManager.settings.modify.hostname   yes
org.freedesktop.NetworkManager.settings.modify.own        yes
org.freedesktop.NetworkManager.settings.modify.system   yes
org.freedesktop.NetworkManager.sleep-wake              yes
org.freedesktop.NetworkManager.wifi.scan               yes
org.freedesktop.NetworkManager.wifi.share.open         yes
org.freedesktop.NetworkManager.wifi.share.protected    yes
```

The following example list the user's NetworkManager permissions.

```
[user@host ~]$ nmcli gen permissions
PERMISSION                                     VALUE
org.freedesktop.NetworkManager.checkpoint-rollback      auth
org.freedesktop.NetworkManager.enable-disable-connectivity-check  no
org.freedesktop.NetworkManager.enable-disable-network    no
org.freedesktop.NetworkManager.enable-disable-statistics no
org.freedesktop.NetworkManager.enable-disable-wifi       no
org.freedesktop.NetworkManager.enable-disable-wimax     no
org.freedesktop.NetworkManager.enable-disable-wwan     no
org.freedesktop.NetworkManager.network-control          auth
org.freedesktop.NetworkManager.reload                  auth
org.freedesktop.NetworkManager.settings.modify.global-dns auth
org.freedesktop.NetworkManager.settings.modify.hostname   auth
org.freedesktop.NetworkManager.settings.modify.own        auth
org.freedesktop.NetworkManager.settings.modify.system   auth
org.freedesktop.NetworkManager.sleep-wake              no
org.freedesktop.NetworkManager.wifi.scan               auth
org.freedesktop.NetworkManager.wifi.share.open         no
org.freedesktop.NetworkManager.wifi.share.protected    no
```

## Useful NetworkManager Commands

The following table lists the key `nmcli` commands that are discussed in this section:

Command	Purpose
<code>nmcli dev status</code>	Show the NetworkManager status of all network interfaces.
<code>nmcli con show</code>	List all connections.
<code>nmcli con show <i>name</i></code>	List the current settings for the connection name.
<code>nmcli con add con-name <i>name</i></code>	Add and name a new connection profile.
<code>nmcli con mod <i>name</i></code>	Modify the connection name.
<code>nmcli con reload</code>	Reload the configuration files, after manual file editing.
<code>nmcli con up <i>name</i></code>	Activate the connection name.
<code>nmcli dev dis <i>dev</i></code>	Disconnect the interface, which also deactivates the current connection.
<code>nmcli con del <i>name</i></code>	Delete the specified connection and its configuration file.



## References

For more information, refer to the *Getting Started with nmcli* chapter at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_and\\_managing\\_networking/index#getting-started-with-nmcli\\_configuring-and-managing-networking](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/index#getting-started-with-nmcli_configuring-and-managing-networking)

`NetworkManager(8)`, `nmcli(1)`, `nmcli-examples(5)`, `nm-settings(5)`, `hostnamectl(1)`, `resolv.conf(5)`, `hostname(5)`, `ip(8)`, and `ip-address(8)` man pages

## ► Guided Exercise

# Configure Networking from the Command Line

In this exercise, you use the `nmcli` command to configure network settings.

### Outcomes

- Update a network connection setting from DHCP to static.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start net-configure
```

### Instructions

- 1. Use the `ssh` command to log in to the `servera` machine as the `student` user.

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

- 2. Display the network interface information.



#### Important

Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names might vary according to the course platform and hardware in use.

On your system, locate the interface name (such as `eth1`, `ens06`, or `enp0p2`) that is associated with the Ethernet address `52:54:00:00:fa:0a`. Use this interface name to replace the `eth0` placeholder throughout this exercise if different.

Locate the network interface name that is associated with the Ethernet address `52:54:00:00:fa:0a`. Record or remember this name and use it to replace the `eth0` placeholder in subsequent commands.

```
[root@servera ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        altname enp0s3
        altname ens3
```

► 3. Use the `nmcli` command to view network settings.

3.1. Use the `nmcli con show` to display all connections.

```
[root@servera ~]# nmcli con show
NAME           UUID                                  TYPE      DEVICE
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
System eth1    9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  --
```

3.2. Use the `nmcli con show --active` command to display only the active connections.

Your network interface name should appear under the `DEVICE` column of the output, and the name of the active connection for that device is listed under the `NAME` column. This exercise assumes that the active connection is called `System eth0`. If the name of the active connection is different, then use that name instead of `System eth0` for the rest of this exercise.

```
[root@servera ~]# nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
System eth0    03da038a-3257-4722-a478-53055cc90128  ethernet  eth0
```

3.3. Display all configuration settings for the active connection.

```
[root@servera ~]# nmcli con show "System eth0"
connection.id:          System eth0
connection.uuid:         5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
connection.stable-id:    --
connection.type:         802-3-ethernet
connection.interface-name: eth0
connection.autoconnect:  yes
...output omitted...
ipv4.method:            manual
ipv4.dns:                172.25.250.254,2.2.2.2
ipv4.dns-search:         lab.example.com,example.com
ipv4.dns-options:        --
ipv4.dns-priority:       0
ipv4.addresses:          172.25.250.10/24
ipv4.gateway:            172.25.250.254
...output omitted...
ipv6.method:            ignore
ipv6.dns:                --
```

```

ipv6.dns-search:          --
ipv6.dns-options:         --
ipv6.dns-priority:        0
ipv6.addresses:           --
ipv6.gateway:             --
ipv6.routes:              --
...output omitted...
GENERAL.NAME:             System eth0
GENERAL.UUID:              5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
GENERAL.DEVICES:           eth0
GENERAL.IP-IFACE:          eth0
GENERAL.STATE:             activated
GENERAL.DEFAULT:            yes

```

### 3.4. Show the device status.

```
[root@servera ~]# nmcli dev status
DEVICE  TYPE      STATE      CONNECTION
eth0    ethernet  connected  System eth0
lo     loopback  unmanaged  --
```

### 3.5. Display the settings for the eth0 device.

```
[root@servera ~]# nmcli dev show eth0
GENERAL.DEVICE:             eth0
GENERAL.TYPE:                ethernet
GENERAL.HWADDR:              52:54:00:00:FA:0A
GENERAL.MTU:                 1500
GENERAL.STATE:               100 (connected)
GENERAL.CONNECTION:          System eth0
GENERAL.CON-PATH:            /org/freedesktop/NetworkManager/ActiveConnection/3
WIRED-PROPERTIES.CARRIER:   on
IP4.ADDRESS[1]:              172.25.250.10/24
IP4.GATEWAY:                 172.25.250.254
IP4.ROUTE[1]:                dst = 172.25.250.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:                dst = 0.0.0.0/0, nh = 172.25.250.254, mt = 100
IP4.DNS[1]:                  172.25.250.254
IP4.SEARCHES[1]:              lab.example.com
IP4.SEARCHES[2]:              example.com
IP6.ADDRESS[1]:              fe80::5054:ff:fe00:fa0a/64
IP6.GATEWAY:                 --
IP6.ROUTE[1]:                dst = fe80::/64, nh = ::, mt = 256
```

- ▶ 4. Create a static connection with the same IPv4 address, network prefix, and default gateway as the active connection. Name the new connection `static-addr`.



#### Warning

Because access to your machine is provided over the primary network connection, setting incorrect values during network configuration might make your machine unreachable. If your machine is unreachable, then use the **Reset** button above what used to be your machine's graphical display and try again.

```
[root@servera ~]# nmcli con add con-name static-addr \
  ifname eth0 type ethernet ipv4.method manual \
  ipv4.addresses 172.25.250.10/24 ipv4.gateway 172.25.250.254
Connection 'static-addr' (c242697d-498e-481c-b974-5ae11d2a0291) successfully
added.
```

- 5. Modify the new connection to add the DNS setting.

```
[root@servera ~]# nmcli con mod static-addr ipv4.dns 172.25.250.254
```

- 6. Display and activate the new connection.

- 6.1. View all connections.

```
[root@servera ~]# nmcli con show
NAME           UUID                                  TYPE      DEVICE
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
static-addr    e4cf52d3-40fc-41b3-b5e8-cf280157f3bb  ethernet  --
System eth1    9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  --
```

- 6.2. View the active connections.

```
[root@servera ~]# nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
System eth0    5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
```

- 6.3. Activate the new static-addr connection.

```
[root@servera ~]# nmcli con up static-addr
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/4)
```

- 6.4. Verify the new active connection.

```
[root@servera ~]# nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
static-addr    e4cf52d3-40fc-41b3-b5e8-cf280157f3bb  ethernet  eth0
```

- 7. Update the previous connection so that it does not start at boot. Verify that the static-addr connection is used when the system reboots.

- 7.1. Disable the original connection so that it does not start automatically at boot.

```
[root@servera ~]# nmcli con mod "System eth0" \
connection.autoconnect no
```

- 7.2. Reboot the system.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- 7.3. Log in to the servera machine and verify that the static-addr connection is the active connection.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$ nmcli con show --active
NAME           UUID                                  TYPE      DEVICE
static-addr    e4cf52d3-40fc-41b3-b5e8-cf280157f3bb  ethernet  eth0
```

► 8. Test connectivity by using the new network addresses.

- 8.1. Verify the IP address.

```
[student@servera ~]$ ip -br addr show eth0
eth0      UP            172.25.250.10/24 fe80::47cd:2076:4a6b:e730/64
```

- 8.2. Verify the default gateway.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev eth0 proto static metric 100
172.25.250.0/24 dev eth0 proto kernel scope link src 172.25.250.10 metric 100
```

- 8.3. Ping the DNS address.

```
[student@servera ~]$ ping -c3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.669 ms
64 bytes from 172.25.250.254: icmp_seq=2 ttl=64 time=0.294 ms
64 bytes from 172.25.250.254: icmp_seq=3 ttl=64 time=0.283 ms

--- 172.25.250.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2035ms
rtt min/avg/max/mdev = 0.283/0.415/0.669/0.179 ms
```

- 8.4. Return to the workstation system as the student user.

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

## 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 net-configure
```

This concludes the section.

# Edit Network Configuration Files

---

## Objectives

Modify network configuration by editing configuration files.

## Connection Configuration Files

Starting with Red Hat Enterprise Linux 8, network configurations are stored in the `/etc/NetworkManager/system-connections/` directory. This new configuration location uses the key file format instead of the `ifcfg` format. However, the previously stored configurations at `/etc/sysconfig/network-scripts/` continue to work. The `/etc/NetworkManager/system-connections/` directory stores any changes with the `nmcli con mod name` command.

## Key File Format

The NetworkManager uses the INI-style key format for storing network connection profiles. The key-value pairs store configurations as sections (groups). Each configuration key/value pair in the section is one of the listed properties in the settings specification. This configuration file stores most of the settings in the same format as the INI-style format. For example, writing IP addresses as `192.168.0.1/24` is easier to read than as integer arrays.

Although the recommended way to manage profiles is with the `nmcli` command, users might still manually create or modify the configuration files. After editing the configuration file, run the `nmcli con reload` command to inform NetworkManager about these changes.

### Comparison of NetworkManager Settings and Key File Format File

<code>nmcli con mod</code>	<code>*.nmconnection</code> file	Effect
<code>ipv4.method manual</code>	<code>[ipv4]</code> <code>method=manual</code>	Configure IPv4 addresses statically.
<code>ipv4.method auto</code>	<code>[ipv4]</code> <code>method=auto</code>	Look for configuration settings from a DHCPv4 server. It does not bring up any static addresses until it has information from DHCPv4.
<code>ipv4.addresses 192.0.2.1/24</code>	<code>[ipv4]</code> <code>address1=192.0.2.1/24</code>	Set a static IPv4 address and network prefix. For more than one connection address, the <code>address2</code> key defines the second address, and the <code>address3</code> key defines the third address.

<b>nmcli con mod</b>	<b>* .nmconnection file</b>	<b>Effect</b>
ipv4.gateway 192.0.2.254	[ipv4] gateway=192.0.2.254	Set the default gateway.
ipv4.dns 8.8.8.8	[ipv4] dns=8.8.8.8	Modify /etc/resolv.conf to use this name server.
ipv4.dns-search example.com	[ipv4] dns-search=example.com	Modify /etc/resolv.conf to use this domain in the search directive.
ipv4.ignore-auto-dns true	[ipv4] ignore-auto-dns=true	Ignore DNS server information from the DHCP server.
ipv6.method manual	[ipv6] method=manual	Configure IPv6 addresses statically.
ipv6.method auto	[ipv6] method=auto	Configure network settings with SLAAC from router advertisements.
ipv6.method dhcp	[ipv6] method=dhcp	Configure network settings by using DHCPv6, but not SLAAC.
ipv6.addresses 2001:db8::a/64	[ipv6] address1=2001:db8::a/64	Set static IPv6 address and network prefix. When using more than one address for a connection, the address2 key defines the second address, and the address3 key defines the third address.
ipv6.gateway 2001:db8::1	[ipv6] gateway=2001:db8::1	Set the default gateway.
ipv6.dns fde2:6494:1e09:2::d	[ipv6] dns=fde2:6494:1e09:2::d	Modify /etc/resolv.conf to use this name server. The same as IPv4.
ipv6.dns-search example.com	[ipv6] dns-search=example.com	Modify /etc/resolv.conf to use this domain in the search directive.
ipv6.ignore-auto-dns true	[ipv6] ignore-auto-dns=true	Ignore DNS server information from the DHCP server.

<b>nmcli con mod</b>	<b>* .nmconnection file</b>	<b>Effect</b>
connection.autoconnect yes	[connection] autoconnect=true	Automatically activate this connection at boot.
connection.id ens3	[connection] id=Main eth0	The name of this connection.
connection.interface-name ens3	[connection] interface-name=ens3	The connection is bound to the network interface with this name.
802-3-ethernet.mac-address ...	[802-3-ethernet] mac-address=	The connection is bound to the network interface with this MAC address.

## Modify Network Configuration

You can also configure the network by directly editing the connection configuration files. Connection configuration files control the software interfaces for individual network devices. These files are usually called `/etc/sysconfig/network-scripts/name.nmconnection`, where `name` refers to the device's name or connection that the configuration file controls.

Depending on the purpose of the connection profile, NetworkManager uses the following directories to store the configuration files:

- The `/etc/NetworkManager/system-connections/` directory stores persistent profiles that the user created and edited. NetworkManager copies them automatically to the `/etc/NetworkManager/system-connections/` directory.
- The `/run/NetworkManager/system-connections/` directory stores temporary profiles, which are automatically removed when you reboot the system.
- The `/usr/lib/NetworkManager/system-connections/` directory stores predeployed immutable profiles. When you edit such a profile with the NetworkManager API, NetworkManager copies this profile to either the persistent or the temporary storage.

Sample configuration file content for static IPv4 configuration:

```
[connection]
id=Main eth0
uuid=27afa607-ee36-43f0-b8c3-9d245cdc4bb3
type=802-3-ethernet
autoconnect=true

[ipv4]
method=auto

[802-3-ethernet]
mac-address=00:23:5a:47:1f:71
```

**IPv4 Configuration Options for Key File Format**

<b>Static</b>	<b>Dynamic</b>	<b>Either</b>
[ipv4] address=172.25.0.10/24 gateway=172.25.0.254 dns=172.25.254.254	method=auto	[connection] interface-name=ens3 id>Main eth0 autoconnect=true uuid=f3e8(...)ad3e type=etherent

After modifying the configuration files, set permissions on the configuration file for the `root` user to read and modify the configuration file.

```
[root@host ~]# chown root:root /etc/NetworkManager/system-connections/"Main
eth0.nmconnection"
[root@host ~]# chmod 600 /etc/NetworkManager/system-connections/"Main
eth0.nmconnection"
```

Run the `nmcli con reload` command for NetworkManager to read the configuration changes. When the `autoconnect` variable in the profile uses the `false` value, then activate the connection.

```
[root@host ~]# nmcli con reload
[root@host ~]# nmcli con up "static-ens3"
```

**References**

`nmcli(1)`, `nm-settings(5)`, and `nm-settings-keyfile(5)` man page

For more information, refer to the *Manually Creating NetworkManager Profiles in Key File Format* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_and\\_managing\\_networking/assembly\\_manually-creating-networkmanager-profiles-in-key-file-format\\_configuring-and-managing-networking](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/assembly_manually-creating-networkmanager-profiles-in-key-file-format_configuring-and-managing-networking)

## ► Guided Exercise

# Edit Network Configuration Files

In this exercise, you manually modify network configuration files and ensure that the new settings take effect.

### Outcomes

- Configure additional network addresses on each system.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start net-edit
```

### Instructions

- 1. On the `workstation` machine, use the `ssh` command to log in to the `servera` machine as the `student` user.

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

- 2. Locate network interface names with the `ip link` command.



#### Important

Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names might vary according to the course platform and hardware in use.

Locate the network interface name that is associated with the Ethernet address on your system. Record or remember this name and use it to replace the `enX` placeholder in subsequent commands. The active connection is called `Wired connection 1` and the configuration is in the `/etc/NetworkManager/system-connections/"Wired connection 1.nmconnection"` file.

```
[student@servera ~]$ ip link  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT  
group default qlen 1000  
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

**Chapter 13 |** Manage Networking

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
  DEFAULT group default qlen 1000
    link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
[student@servera ~]$ nmcli con show --active
NAME           UUID             TYPE      DEVICE
Wired connection 1  a98933fa-25c0-36a2-b3cd-c056f41758fe  ethernet  eth0
[student@servera ~]$ ls /etc/NetworkManager/system-connections/
'Wired connection 1.nmconnection'
```

- 3. On the servera machine, switch to the root user, and then edit the /etc/NetworkManager/system-connections/"Wired connection 1.nmconnection" file to add the 10.0.1.1/24 address.
- 3.1. Use the sudo -i command to switch to the root user.

```
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- 3.2. Edit the configuration file. Add the 10.0.1.1/24 address as the second address below the first address in the file.

```
[root@servera ~]# vim /etc/NetworkManager/system-connections/"Wired connection
  1.nmconnection"
..output omitted...
[ipv4]
address1=172.25.250.10/24,172.25.250.254
address2=10.0.1.1/24
...output omitted...
```

- 4. Activate the new network address with the nmcli command.

- 4.1. Reload the configuration changes for NetworkManager to read the changes.

```
[root@servera ~]# nmcli con reload
```

- 4.2. Activate the connection with the changes.

```
[root@servera ~]# nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/2)
```

- 5. Verify that the new IP address is assigned successfully.

```
[root@servera ~]# ip -br addr show enx
eth0:      UP      172.25.250.10/24 10.0.1.1/24 fe80::6fed:5a11:4ad4:1bcf/64
```

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

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

- 7. On the **serverb** machine, edit the **/etc/NetworkManager/system-connections/"Wired connection 1.nmconnection"** file to add an address of **10.0.1.2/24** and load the new configuration.

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

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

7.2. Edit the configuration file. Add the **10.0.1.2/24** address as the second address below the first address in the file.

```
[root@serverb ~]# vim /etc/NetworkManager/system-connections/"Wired connection
1.nmconnection"
address1=172.25.250.11/24,172.25.250.254
address2=10.0.1.2/24
```

7.3. Reload the configuration changes for NetworkManager to read the changes.

```
[root@serverb ~]# nmcli con reload
```

7.4. Activate the connection with the changes.

```
[root@serverb ~]# nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/2)
```

7.5. Verify that the new IP address is assigned successfully.

```
[root@serverb ~]# ip -br addr show enX
eth0      UP      172.25.250.11/24 10.0.1.2/24 fe80::6be8:6651:4280:892c/64
```

- 8. Test connectivity between the **servera** and **serverb** machines by using the new network addresses.

8.1. From the **serverb** machine, ping the new address of the **servera** machine.

```
[root@serverb ~]# ping -c3 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=1.30 ms
```

```
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.983 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.312 ms

--- 10.0.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.312/0.864/1.297/0.410 ms
```

- 8.2. Return to the **workstation** machine as the **student** user.

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

- 8.3. Access the **servera** machine as the **student** user to ping the new address of the **serverb** machine.

```
[student@workstation ~]$ ssh student@servera ping -c3 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=0.876 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=0.289 ms

--- 10.0.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.289/0.491/0.876/0.271 ms
```

## 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 net-edit
```

This concludes the section.

# Configure Hostnames and Name Resolution

---

## Objectives

Configure a server's static hostname and its name resolution and test the results.

### Update the System Hostname

The `hostname` command displays or temporarily modifies the system's fully qualified hostname.

```
[root@host ~]# hostname
host.example.com
```

Specify a static hostname in the `/etc/hostname` file. Use the `hostnamectl` command to modify this file and view the system's fully qualified hostname. If this file does not exist, then the hostname is set by a reverse DNS query when an IP address is assigned to the interface.

```
[root@host ~]# hostnamectl set-hostname host.example.com
[root@host ~]# hostnamectl status
Static hostname: host.example.com
        Icon name: computer-vm
          Chassis: vm #
      Machine ID: 663e281edea34ffa297bd479a8f12b5
        Boot ID: 74bf3a0a48d540998a74055a0fe38821
  Virtualization: kvm
Operating System: Red Hat Enterprise Linux 9.0 (Plow)
      CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
        Kernel: Linux 5.14.0-70.el9.x86_64
    Architecture: x86-64
  Hardware Vendor: Red Hat
  Hardware Model: OpenStack Compute
[root@host ~]# cat /etc/hostname
host.example.com
```



#### Important

In Red Hat Enterprise Linux 7 and later versions, the static hostname is stored in the `/etc/hostname` file. Red Hat Enterprise Linux 6 and earlier versions store the hostname as a variable in the `/etc/sysconfig/network` file.

## Configure Name Resolution

The stub resolver converts hostnames to IP addresses or the reverse. It determines where to look based on the configuration of the `/etc/nsswitch.conf` file. By default, it attempts to resolve the query by first using the `/etc/hosts` file.

```
[root@host ~]# cat /etc/hosts
127.0.0.1      localhost localhost.localdomain localhost4 localhost4.localdomain4
::1            localhost localhost.localdomain localhost6 localhost6.localdomain6
172.25.254.254 classroom.example.com
172.25.254.254 content.example.com
```

The `getent hosts hostname` command tests hostname resolution with the `/etc/hosts` file. If an entry is not found in the `/etc/hosts` file, then the stub resolver uses a DNS name server to look up the hostname. The `/etc/resolv.conf` file controls how this query is performed:

- **search** : A list of domain names to try with a short hostname. Either `search` or `domain` should be set in the same file; if they are both set, then only the last entry takes effect. See `resolv.conf(5)` for details.
- **nameserver** : the IP address of a name server to query. Up to three name server directives can be given to provide backups if one name server is down.

```
[root@host ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager uses DNS settings in the connection configuration files to update the `/etc/resolv.conf` file. Use the `nmcli` command to modify the connections.

```
[root@host ~]# nmcli con mod ID ipv4.dns IP
[root@host ~]# nmcli con down ID
[root@host ~]# nmcli con up ID
[root@host ~]# cat /etc/sysconfig/network-scripts/ifcfg-ID
...output omitted...
DNS1=8.8.8.8
...output omitted...
```

The default behavior of the `nmcli con mod ID ipv4.dns IP` command is to replace any previous DNS settings with the new IP list that is provided. A plus (+) or minus (-) character in front of the `nmcli` command `ipv4.dns` option adds or removes an individual entry, respectively.

```
[root@host ~]# nmcli con mod ID +ipv4.dns IP
```

To add the DNS server with an IPv6 IP address of `2001:4860:4860::8888` to the list of name servers on the `static-ens3` connection:

```
[root@host ~]# nmcli con mod static-ens3 +ipv6.dns 2001:4860:4860::8888
```



### Note

Static IPv4 and IPv6 DNS settings become `nameserver` directives in `/etc/resolv.conf`. On a dual-stack system, keep at least one IPv4-reachable and an IPv6 name server listed (assuming a dual-stack system), in case of networking issues with either stack.

## Test DNS Name Resolution

The host *HOSTNAME* command can test DNS server connectivity.

```
[root@host ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
[root@host ~]# host 172.25.254.254
254.254.25.172.in-addr.arpa domain name pointer classroom.example.com.
```



### Important

DHCP automatically rewrites the `/etc/resolv.conf` file when interfaces are started, unless you specify `PEERDNS=no` in the relevant interface configuration files. Set this entry by using the `nmcli` command.

```
[root@host ~]# nmcli con mod "static-ens3" ipv4.ignore-auto-dns yes
```

Use the `dig` *HOSTNAME* command to test DNS server connectivity.

```
[root@host ~]# dig classroom.example.com

; <>> DiG 9.16.23-RH <>> classroom.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3451
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 947ea2a936353423c3bc0d5f627cc1ae7147460e10d2777c (good)
;; QUESTION SECTION:
;classroom.example.com. IN A

;; ANSWER SECTION:
classroom.example.com. 85326 IN A 172.25.254.254
...output omitted...
```

Both the `host` and `dig` commands do not view the configuration in the `/etc/hosts` file. To test the `/etc/hosts` file, use the `getent hosts` *HOSTNAME* command.

```
[root@host ~]# getent hosts classroom.example.com
172.25.254.254 classroom.example.com
```



### References

`nmcli(1)`, `hostnamectl(1)`, `hosts(5)`, `getent(1)`, `host(1)`, `dig(1)`, `getent(1)`, and `resolv.conf(5)` man pages

For more information, refer to the *Configuring and Managing Networking Guide* at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/configuring\\_and\\_managing\\_networking/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/index)

## ► Guided Exercise

# Configure Hostnames and Name Resolution

In this exercise, you manually configure the system's static hostname, /etc/hosts file, and DNS name resolver.

## Outcomes

- Set a customized hostname.
- Configure name resolution settings.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that all required resources are available.

```
[student@workstation ~]$ lab start net-hostnames
```

## Instructions

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

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

- 2. View the current hostname settings.

- 2.1. Display the current hostname.

```
[root@servera ~]# hostname
servera.lab.example.com
```

- 2.2. Display the hostname status. Note the transient hostname that is obtained from DHCP or mDNS.

```
[root@servera ~]# hostnamectl status
Static hostname: n/a
Transient hostname: servera.lab.example.com
Icon name: computer-vm
Chassis: vm
Machine ID: 63b272eae8d5443ca7aaa5593479b25f
Boot ID: ef299e0e957041ee81d0617fc98ce5ef
```

```
Virtualization: kvm
Operating System: Red Hat Enterprise Linux 9.0 (Plow)
  CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
    Kernel: Linux 5.14.0-70.el9.x86_64
  Architecture: x86-64
Hardware Vendor: Red Hat
  Hardware Model: OpenStack Compute
```

► **3.** Set a static hostname to match the current transient hostname.

3.1. Change the hostname and the hostname configuration file.

```
[root@servera ~]# hostnamectl set-hostname \
servera.lab.example.com
```

3.2. View the content of the /etc/hostname file, which provides the hostname at network start.

```
servera.lab.example.com
```

3.3. Display the hostname status. The transient hostname is not shown, now that a static hostname is configured.

```
[root@servera ~]# hostnamectl status
Static hostname: servera.lab.example.com
  Icon name: computer-vm
  Chassis: vm
  Machine ID: 63b272eae8d5443ca7aaa5593479b25f
  Boot ID: ef299e0e957041ee81d0617fc98ce5ef
  Virtualization: kvm
Operating System: Red Hat Enterprise Linux 9.0 (Plow)
  CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
    Kernel: Linux 5.14.0-70.el9.x86_64
  Architecture: x86-64
Hardware Vendor: Red Hat
  Hardware Model: OpenStack Compute
```

► **4.** Temporarily change the hostname to testname.

4.1. Change the hostname.

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

4.2. Display the current hostname.

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

4.3. View the content of the /etc/hostname file, which provides the hostname at network start.

```
servera.lab.example.com
```

- 4.4. Reboot the system.

```
[root@servera ~]# systemctl reboot  
Connection to servera closed by remote host.  
Connection to servera closed.  
[student@workstation ~]$
```

- 4.5. Log in to **servera** as the **student** user and switch to **root** user.

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

- 4.6. Display the current hostname.

```
[root@servera ~]# hostname  
servera.lab.example.com
```

- 5. Add **class** as a local nickname for the classroom server, and ensure that you can ping the server with that nickname.

- 5.1. Look up the IP address of **classroom.example.com**.

```
[root@servera ~]# host classroom.example.com  
classroom.example.com has address 172.25.254.254
```

- 5.2. Update the **/etc/hosts** file to add **class** to access the IP address 172.25.254.254. The following example shows the expected content of the **/etc/hosts** file.

```
[root@servera ~]# vim /etc/hosts  
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
172.25.254.254 classroom.example.com classroom class
```

- 5.3. Look up the IP address of **class**.

```
[root@servera ~]# host class  
Host class not found: 3(NXDOMAIN)  
[root@servera ~]# getent hosts class  
172.25.254.254 classroom.example.com classroom class
```

- 5.4. Use the **ping** command to send packets to the **class** server.

```
[root@servera ~]# ping -c3 class
PING classroom.example.com (172.25.254.254) 56(84) bytes of data.
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=1 ttl=63 time=1.21
ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=2 ttl=63 time=0.688
ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=3 ttl=63 time=0.559
ms

--- classroom.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.559/0.820/1.214/0.283 ms
```

5.5. 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, 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 net-hostnames
```

This concludes the section.

## ► Lab

# Manage Networking

In this lab, you configure networking settings on a Red Hat Enterprise Linux server.

## Outcomes

- Configure two static IPv4 addresses for the primary network interface.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

- Log in to the `serverb` machine as the `student` user. Switch to the `root` user.
- Create a connection with a static network configuration by using the settings in the table.

Parameter	Setting
Connection name	lab
Interface name	enX (might vary; use the interface with 52:54:00:00:fa:0b as its MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

- Configure the new connection to start automatically. Other connections should not start automatically.
- Modify the new connection so that it also uses the IP address 10.0.1.1/24.
- Configure the `hosts` file so that you can reference the 10.0.1.1 IP address with the `private` name.
- Reboot the system.
- Verify that the `serverb` machine is initialized.

## 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 net-review
```

## 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 net-review
```

This concludes the section.

## ► Solution

# Manage Networking

In this lab, you configure networking settings on a Red Hat Enterprise Linux server.

## Outcomes

- Configure two static IPv4 addresses for the primary network interface.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

- Log in to the `serverb` machine as the `student` user. Switch to the `root` user.
  - Log in to the `serverb` machine as the `student` user and switch to the `root` user.

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

- Create a connection with a static network configuration by using the settings in the table.

Parameter	Setting
Connection name	lab
Interface name	enX (might vary; use the interface with 52:54:00:00:fa:0b as its MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

Determine the interface name and the current active connection's name. The solution assumes that the interface name is `eth0` and that the connection name is `System eth0`.

```
[root@serverb ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:00:fa:0b brd ff:ff:ff:ff:ff:ff
        altname enp0s3
        altname ens3
[root@serverb ~]# nmcli con show --active
NAME           UUID             TYPE      DEVICE
System eth0   5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
```

Create the lab connection profile based on the table information in the instructions.

Associate the profile with your network interface name, as listed in the output of the previous `ip link` command.

```
[root@serverb ~]# nmcli con add con-name lab iface eth0 type ethernet \
ipv4.method manual \
ipv4.addresses 172.25.250.11/24 ipv4.gateway 172.25.250.254
[root@serverb ~]# nmcli con mod "lab" ipv4.dns 172.25.250.254
```

- Configure the new connection to start automatically. Other connections should not start automatically.

```
[root@serverb ~]# nmcli con mod "lab" connection.autoconnect yes
[root@serverb ~]# nmcli con mod "System eth0" connection.autoconnect no
```

- Modify the new connection so that it also uses the IP address 10.0.1.1/24.

```
[root@serverb ~]# nmcli con mod "lab" +ipv4.addresses 10.0.1.1/24
```

Or alternately edit the configuration file to add the 10.0.1.1/24 address as the second address.

```
[root@serverb ~]# vim /etc/NetworkManager/system-connections/lab.nmconnection
address2=10.0.1.1/24
```

- Configure the hosts file so that you can reference the 10.0.1.1 IP address with the private name.

```
[root@serverb ~]# echo "10.0.1.1 private" >> /etc/hosts
```

- Reboot the system.

```
[root@serverb ~]# systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

7. Verify that the **serverb** machine is initialized.

```
[student@workstation ~]$ ping -c3 serverb
PING serverb.lab.example.com (172.25.250.11) 56(84) bytes of data.
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=1 ttl=64
time=0.478 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
time=0.504 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64
time=0.513 ms
--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 78ms
rtt min/avg/max/mdev = 0.478/0.498/0.513/0.023 ms
```

## 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 net-review
```

## 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 net-review
```

This concludes the section.

# Summary

---

- The `NetworkManager` daemon monitors and manages network configuration.
- The `nmcli` command is a command-line tool for configuring network settings with the `NetworkManager` daemon.
- Starting in Red Hat Enterprise Linux 9, the default location for network configurations is the `/etc/NetworkManager/system-connections` directory.
- The system's static hostname is stored in the `/etc/hostname` file.
- The `hostnamectl` command modifies or views the status of the system's hostname and related settings.

## Chapter 14

# Access Network-Attached Storage

### Goal

Access network-attached storage with the NFS protocol.

### Objectives

- Identify NFS export information, create a directory to use as a mount point, mount an NFS export with the `mount` command or by configuring the `/etc/fstab` file, and unmount an NFS export with the `umount` command.
- Describe the benefits of using the automounter, and automount NFS exports by using direct and indirect maps.

### Sections

- Manage Network-Attached Storage with NFS (and Guided Exercise)
- Automount Network-Attached Storage (and Guided Exercise)

### Lab

Access Network-Attached Storage

# Manage Network-Attached Storage with NFS

## Objectives

Identify NFS export information, create a directory to use as a mount point, mount an NFS export with the `mount` command or by configuring the `/etc/fstab` file, and unmount an NFS export with the `umount` command.

## Accessing Exported NFS Directories

The *Network File System* (NFS) 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 9 uses NFS version 4.2. RHEL fully supports both NFSv3 and NFSv4 protocols. NFSv3 could use either a TCP or a UDP transport protocol, but NFSv4 allows 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 in multiple ways:

- **Manually** by using the `mount` command.
- **Persistently at boot** by configuring entries in the `/etc/fstab` file.
- **On demand** by configuring an automounter method.

The automounter methods, which include the `autofs` service and the `systemd.automount` facility, are discussed in the `Automount Network-Attached Storage` section. You must install the `nfs-utils` package to obtain the client tools for manually mounting, or for automounting, to obtain exported NFS directories.

```
[root@host ~]# dnf install nfs-utils
```

RHEL also supports mounting *shared* directories from Microsoft Windows systems by using the same methods as for 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.

## Query a Server's 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 used the RPC protocol, which requires a file server that supports NFSv3 connections to run the `rpcbind` service. An NFSv3 client connects to the `rpcbind` service at port 111 on the server to request NFS service. 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 eliminated the use of the legacy RPC protocol for NFS transactions. Use of the `showmount` command on a server that supports only NFSv4 times out without receiving a response, because the `rpcbind` service is not running on the server. However, querying an NFSv4 server is simpler than querying an NFSv3 server.

NFSv4 introduced an *export tree* that contains all of the paths for the server's exported directories. To view all of the exported directories, mount the root (/) of the server's export tree. Mounting the export tree's root provides browseable paths for all exported directories, as children of the tree's root directory, but does not mount ("bind") any of the exported directories.

```
[root@host ~]# mkdir /mountpoint
[root@host ~]# mount server:/ /mountpoint
[root@host ~]# ls /mountpoint
```

To mount an NFSv4 export while browsing the mounted export tree, change directory into 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 while browsing an export tree, even though you can view the export's path name. Mounting Kerberos-protected shares requires additional server configuration and using 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 identifying the NFS export to mount, create a local mount point if it does not yet exist. The `\mnt` directory is available for use as a temporary mount point, but recommended practice is not to 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, 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. The `-o sync` option specifies that all transactions to the exported file system are performed synchronously, which is strongly recommended for all production network mounts where transactions must be completed or else return as failed.

Using a manual `mount` command is not persistent. When the system reboots, that NFS export will not still be mounted. Manual mounts are useful for providing temporary access to an exported directory, or for test mounting an NFS export before persistently mounting it.

## Persistently Mount Exported NFS Directories

To persistently mount an NFS export, edit the `/etc/fstab` file and add the mount entry with similar syntax to manual mounting.

```
[root@host ~]# vim /etc/fstab
...
server:/export /mountpoint nfs rw,soft 0 0
```

Then, you can mount the NFS export by using only the mount point. The `mount` command obtains the NFS server and mount options from the matching entry in the `/etc/fstab` file.

```
[root@host ~]# mount /mountpoint
```

## Unmount Exported NFS Directories

As a privileged user, unmount an NFS export with the `umount` command. Unmounting a share does not remove its entry in the `/etc/fstab` file, if that file exists. Entries in the `/etc/fstab` file are persistent and are remounted during boot.

```
[root@host ~]# umount /mountpoint
```

A mounted directory can sometimes fail to unmount, and returns an error that the device is busy. The device is busy because either an application is keeping a file open within the file system, or some user's shell has a working directory in the mounted file-system's root directory or below it.

To resolve the error, check your own 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` (*list open files*) command to query the mount point. The `lsof` command returns a list of open file names and the process which is keeping the file open.

```
[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 -f` option to force the unmount, which can cause 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

## ► Guided Exercise

# Manage Network-Attached Storage with NFS

### Performance Checklist

In this exercise, you modify the `/etc/fstab` file to persistently mount an NFS export at boot time.

### Outcomes

- Test an NFS server with the `mount` command.
- Configure NFS exports in the `/etc/fstab` configuration file to save changes even after a system reboots.

### Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start netstorage-nfs
```

### Instructions

A shipping company uses a central NFS server, `serverb`, to host various exported documents and directories. Users on `servera`, who are all members of the `admin` group, need access to the persistently mounted NFS export.

Environment Characteristics:

- The `serverb` machine exports the `/shares/public` directory, which contains some text files.
- Members of the `admin` group (`admin1`, `sysmanager1`) have read and write access to the `/shares/public` exported directory.
- The mount point on `servera` must be the `/public` directory.
- All user passwords are set to `redhat`.
- The `nfs-utils` package is already installed.

#### ► 1. Log in to `servera` as the `student` user and switch to the `root` user.

- 1.1. Log in to `servera` 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. Test the NFS server on `serverb` with `servera` as the NFS client.

- 2.1. Create the `/public` mount point on the `servera` machine.

```
[root@servera ~]# mkdir /public
```

- 2.2. On `servera`, verify that the `/share/public` NFS export from `serverb` successfully mounts to the `/public` directory.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/public /public
```

- 2.3. List the contents of the mounted NFS export.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r--. 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root admin 27 Apr  8 22:36 Trackings.txt
```

- 2.4. Explore the `mount` command options for the mounted NFS export.

```
[root@servera ~]# mount | grep public
serverb.lab.example.com:/shares/public on /public type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync
,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. Configure `servera` so that the `/share/public` export is persistently mounted.

- 3.1. Edit the `/etc/fstab` file.

```
[root@servera ~]# vim /etc/fstab
```

Add the following line to the end of the file:

```
serverb.lab.example.com:/shares/public /public nfs rw,sync 0 0
```

- 3.2. Mount the exported directory.

```
[root@servera ~]# mount /public
```

- 3.3. List the contents of the exported directory.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r-- 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r-- 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r-- 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r-- 1 root admin 27 Apr  8 22:36 Trackings.txt
```

3.4. Reboot the servera machine.

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

- ▶ 4. After servera has finished rebooting, log in to servera as the admin1 user and test the persistently mounted NFS export.

4.1. Log in to servera as the admin1 user.

```
[student@workstation ~]$ ssh admin1@servera
[admin1@servera ~]$
```

4.2. Test the NFS export that is mounted on the /public directory.

```
[admin1@servera ~]$ ls -l /public
total 16
-rw-r--r-- 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r-- 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r-- 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r-- 1 root admin 27 Apr  8 22:36 Trackings.txt
[admin1@servera ~]$ cat /public/NOTES.txt
###In this file you can log all your notes###
[admin1@servera ~]$ echo "This is a test" > /public/Test.txt
[admin1@servera ~]$ cat /public/Test.txt
This is a test
```

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

```
[admin1@servera ~]$ exit
logout
Connection to servera closed.
```

## 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 netstorage-nfs
```

This concludes the section.

# Automount Network-Attached Storage

## Objectives

Describe the benefits of using the automounter, and automount NFS exports by using direct and indirect maps.

## Mount NFS Exports with the Automounter

The *automounter* is a service (*autofs*) that automatically mounts file systems and NFS exports on demand, and automatically unmounts file systems and NFS exports when the mounted resources are no longer in current use.

The automounter function was created to solve the problem that unprivileged users do not have sufficient permissions to use the *mount* command. Without use of the *mount* command, normal 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 time by using the */etc/fstab* configuration, then a normal user is unable to mount and access those unmounted file systems.

The automounter configuration files are populated with file system mount information, in a similar way to */etc/fstab* entries. Although */etc/fstab* file systems mount during system boot and remain mounted until system shutdown or other intervention, automounter file systems do not necessarily mount during system boot. Instead, automounter-controlled file systems mount on demand, when a user or application attempts to enter the file system mount point to access files.

## Automounter Benefits

Resource use for automounter file systems is equivalent to file systems that are mounted at boot, because a file system uses resources only when a program is reading and writing open files. Mounted but idle file systems and unmounted file systems use the same amount of resources: almost none.

The automounter advantage is that by unmounting the file system each time it is no longer in use, the file system is protected from unexpected corruption while it is open. When the file system is directed to mount again, the *autofs* service uses the most current mount configuration, unlike an */etc/fstab* mount, 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 *autofs* Service Method

The *autofs* 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 identical behavior to file systems that are mounted by using */etc/fstab*. 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 while 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 time out.

## Direct and Indirect Map Use Cases

The automounter supports both direct and indirect mount-point mapping, to handle the two types of demand mounting. A *direct* mount is when a file system mounts to an unchanging, known mount point location. Almost all the file system mounts that you configured, before learning about the automounter, are examples of direct mounts. A *direct* mount point exists as a permanent directory, the same as other normal directories.

An *indirect* mount is when the mount point location is not known until the mount demand occurs. An example of an indirect mount is the configuration for remote-mounted home directories, where a user's home directory includes their username in the directory path. The user's remote file system is mounted to their home directory, only after the automounter learns which user specified to mount their home directory, and determines the mount point location to use. Although *indirect* mount points appear to exist, the `autofs` service creates them when the mount demand occurs, and deletes them again when the demand has ended and the file system is unmounted.

## Configure the Automounter Service

The process to configure an automount has many steps.

First, you must install the `autofs` and `nfs-utils` packages.

```
[user@host ~]$ sudo dnf install autofs nfs-utils
```

These packages contain all requirements to use the automounter for NFS exports.

## Create a Master Map

Next, add a master map file to `/etc/auto.master.d`. This file identifies the base directory for mount points and identifies the mapping file to create the automounts.

```
[user@host ~]$ sudo vim /etc/auto.master.d/demo.autofs
```

The name of the master map file is mostly arbitrary (although typically meaningful), and it must have an extension of `.autofs` 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.

Add the master map entry for indirectly mapped mounts, in this case:

```
/shares  /etc/auto.demo
```

This entry uses the `/shares` directory as the base for indirect automounts. The `/etc/auto.demo` file contains the mount details. Use an absolute file name. The `auto.demo` file must be created before starting the `autofs` service.

## Create an Indirect Map

Now, create the mapping files. Each mapping file identifies the mount point, mount options, and source location to mount for a set of automounts.

```
[user@host ~]$ sudo vim /etc/auto.demo
```

The mapping file-naming convention is `/etc/auto.name`, where *name* reflects the content of the map.

```
work -rw, sync serverb:/shares/work
```

The format of an entry is *mount point*, *mount options*, and *source location*. This example shows a basic indirect mapping entry. Direct maps and indirect maps that use wildcards are covered later in this section.

Known as the *key* in the man pages, the *mount point* is created and removed automatically by the `autofs` service. In this case, the fully qualified mount point is `/shares/work` (see the master map file). The `/shares` and `/shares/work` directories are created and removed as needed by the `autofs` service.

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 `-fstype=` and `-strict`. Use `fstype` to specify the file-system type, for example `nfs4` or `xfs`, and use `strict` to treat errors when mounting file systems as fatal.

The source location for NFS exports follows the host:/pathname pattern, in this example `serverb:/shares/work`. For this automount to succeed, the NFS server, `serverb`, must *export* the directory with read/write access, and the user that requests access must have standard Linux file permissions on the directory. If `serverb` exports the directory with read/only access, then the client gets read/only access even if it requested read/write access.

## Wildcards in an Indirect Map

When an NFS server exports multiple subdirectories within a directory, then the automounter can be configured to access any of those subdirectories with a single mapping entry.

Continuing the previous example, if `serverb:/shares` exports two or more subdirectories, and they are accessible with the same mount options, then the content for the `/etc/auto.demo` file might appear as follows:

```
* -rw, sync serverb:/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 `/shares/work`, the `*` key (which is `work` in this example) replaces the ampersand in the source location and `serverb:/exports/work` is mounted. As with the indirect example, the `autofs` service creates and removes the `work` directory automatically.

## Create a Direct Map

A direct map is used to map 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, the master map file might appear as follows:

```
/ - /etc/auto.direct
```

All direct map entries use `/ -` as the base directory. In this case, the mapping file that contains the mount details is `/etc/auto.direct`.

The content for the `/etc/auto.direct` file might appear as follows:

```
/mnt/docs -rw, sync serverb:/shares/docs
```

The mount point (or key) is always an absolute path. The rest of the mapping file uses the same structure.

In this example, the `/mnt` directory exists, and it is not managed by the `autofs` service. The `autofs` service creates and removed the full `/mnt/docs` directory automatically.

## Start the Automounter Service

Lastly, use the `systemctl` command to start and enable the `autofs` service.

```
[user@host ~]$ sudo systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /usr/
lib/systemd/system/autofs.service.
```

## The Alternative `systemd.automount` Method

The `systemd` daemon can automatically create unit files for entries in `/etc/fstab` that include the `x-systemd.automount` option. Use the `systemctl daemon-reload` command after modifying an entry's mount options, to generate a new unit file, and then use the `systemctl start unit.automount` command to enable that 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 `autofs` direct maps.



### References

`autofs(5)`, `automount(8)`, `auto.master(5)`, `mount.nfs(8)`, and `systemd.automount(5)` man pages

## ► Guided Exercise

# Automount Network-Attached Storage

### Performance Checklist

In this exercise, you create direct-mapped and indirect-mapped automount-managed mount points that mount NFS file systems.

### Outcomes

- Install required packages for the automounter.
- Configure direct and indirect automounter maps, with resources from a preconfigured NFSv4 server.
- Describe the difference between direct and indirect automounter maps.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This start script determines whether `servera` and `serverb` are reachable on the network. The script alerts you if they are not available. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. The script also creates users and groups that are needed on both `servera` and `serverb`.

```
[student@workstation ~]$ lab start netstorage-autofs
```

### Instructions

An internet service provider uses a central server, `serverb`, to host shared directories with important documents that must be available on demand. When users log in to `servera`, they need access to the automounted shared directories.

Environment Characteristics:

- The `serverb` machine exports the `/shares/indirect` directory, which in turn contains the `west`, `central`, and `east` subdirectories.
- The `serverb` machine also exports the `/shares/direct/external` directory.
- The `operators` group consists of the `operator1` and `operator2` users. They have read and write access to the `/shares/indirect/west`, `/shares/indirect/central`, and `/shares/indirect/east` exported directories.
- The `contractors` group consists of the `contractor1` and `contractor2` users. They have read and write access to the `/shares/direct/external` exported directory.
- The expected mount points for `servera` are `/external` and `/internal`.
- The `/shares/direct/external` exported directory is automounted on `servera` with a `direct` map on `/external`.
- The `/shares/indirect/west` exported directory is automounted on `servera` with an `indirect` map on `/internal/west`.
- The `/shares/indirect/central` exported directory is automounted on `servera` with an `indirect` map on `/internal/central`.

- The `/shares/indirect/east` exported directory is automounted on `servera` with an `indirect` map on `/internal/east`.
- All user passwords are set to `redhat`.
- The `nfs-utils` package is already installed.

► 1. Log in to `servera` and install the required packages.

1.1. Log in to `servera` 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 ~]#
```

1.2. Install the `autofs` package.

```
[root@servera ~]# dnf install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

► 2. Configure an automounter direct map on `servera` with exports from `serverb`. Create the direct map with files that are named `/etc/auto.master.d/direct.autofs` for the master map and `/etc/auto.direct` for the mapping file. Use the `/external` directory as the main mount point on `servera`.

2.1. Test the NFS server and export before you configure the automounter.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/direct/external /mnt
[root@servera ~]# ls -l /mnt
total 4
-rw-r--r-- 1 root contractors 22 Apr  7 23:15 README.txt
[root@servera ~]# umount /mnt
```

2.2. Create a master map file named `/etc/auto.master.d/direct.autofs`, insert the following content, and save the changes.

```
/- /etc/auto.direct
```

2.3. Create a direct map file named `/etc/auto.direct`, insert the following content, and save the changes.

```
/external -rw, sync, fstype=nfs4 serverb.lab.example.com:/shares/direct/external
```

► 3. Configure an automounter indirect map on `servera` with exports from `serverb`. Create the indirect map with files that are named `/etc/auto.master.d/indirect.autofs` for the master map and `/etc/auto.indirect` for the mapping file. Use the `/internal` directory as the main mount point on `servera`.

- 3.1. Test the NFS server and export before you configure the automounter.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/indirect /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrws--- 2 root operators 24 Apr  7 23:34 central
drwxrws--- 2 root operators 24 Apr  7 23:34 east
drwxrws--- 2 root operators 24 Apr  7 23:34 west
[root@servera ~]# umount /mnt
```

- 3.2. Create a master map file named /etc/auto.master.d/indirect.autofs, insert the following content, and save the changes.

```
/internal /etc/auto.indirect
```

- 3.3. Create an indirect map file named /etc/auto.indirect, insert the following content, and save the changes.

```
* -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/indirect/&
```

- 4. Start the autofs service on servera and enable it to start automatically at boot time. Reboot servera to determine whether the autofs service starts automatically.

- 4.1. Start and enable the autofs service on servera.

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

- 4.2. Reboot the servera machine.

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

- 5. Test the direct automounter map as the contractor1 user. When done, exit from the contractor1 user session on servera.

- 5.1. After the servera machine is finished booting, log in to servera as the student user.

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

- 5.2. Switch to the contractor1 user.

```
[student@servera ~]$ su - contractor1
Password: redhat
```

- 5.3. List the /external mount point.

```
[contractor1@servera ~]$ ls -l /external
total 4
-rw-r--r--. 1 root contractors 22 Apr  7 23:34 README.txt
```

- 5.4. Review the content and test the access to the /external mount point.

```
[contractor1@servera ~]$ cat /external/README.txt
###External Folder###
[contractor1@servera ~]$ echo testing-direct > /external/testing.txt
[contractor1@servera ~]$ cat /external/testing.txt
testing-direct
```

- 5.5. Exit from the contractor1 user session.

```
[contractor1@servera ~]$ exit
logout
[student@servera ~]$
```

- ▶ 6. Test the indirect automounter map as the operator1 user. When done, log out from servera.

- 6.1. Switch to the operator1 user.

```
[student@servera ~]$ su - operator1
Password: redhat
```

- 6.2. List the /internal mount point.

```
[operator1@servera ~]$ ls -l /internal
total 0
```



### Note

With an automounter indirect map, you must access each exported subdirectory for them to mount. With an automounter direct map, after you access the mapped mount point, you can immediately view and access the subdirectories and content in the exported directory.

- 6.3. Test the /internal/west automounter exported directory access.

```
[operator1@servera ~]$ ls -l /internal/west/
total 4
-rw-r--r--. 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/west/README.txt
###West Folder###
[operator1@servera ~]$ echo testing-1 > /internal/west/testing-1.txt
[operator1@servera ~]$ cat /internal/west/testing-1.txt
testing-1
```

```
[operator1@servera ~]$ ls -l /internal  
total 0  
drwxrws---. 2 root operators 24 Apr 7 23:34 west
```

6.4. Test the /internal/central automounter exported directory access.

```
[operator1@servera ~]$ ls -l /internal/central  
total 4  
-rw-r--r--. 1 root operators 21 Apr 7 23:34 README.txt  
[operator1@servera ~]$ cat /internal/central/README.txt  
###Central Folder###  
[operator1@servera ~]$ echo testing-2 > /internal/central/testing-2.txt  
[operator1@servera ~]$ cat /internal/central/testing-2.txt  
testing-2  
[operator1@servera ~]$ ls -l /internal  
total 0  
drwxrws---. 2 root operators 24 Apr 7 23:34 central  
drwxrws---. 2 root operators 24 Apr 7 23:34 west
```

6.5. Test the /internal/east automounter exported directory access.

```
[operator1@servera ~]$ ls -l /internal/east  
total 4  
-rw-r--r--. 1 root operators 18 Apr 7 23:34 README.txt  
[operator1@servera ~]$ cat /internal/east/README.txt  
###East Folder###  
[operator1@servera ~]$ echo testing-3 > /internal/east/testing-3.txt  
[operator1@servera ~]$ cat /internal/east/testing-3.txt  
testing-3  
[operator1@servera ~]$ ls -l /internal  
total 0  
drwxrws---. 2 root operators 24 Apr 7 23:34 central  
drwxrws---. 2 root operators 24 Apr 7 23:34 east  
drwxrws---. 2 root operators 24 Apr 7 23:34 west
```

6.6. Test the /external automounter exported directory access.

```
[operator1@servera ~]$ ls -l /external  
ls: cannot open directory '/external': Permission denied
```

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

```
[operator1@servera ~]$ exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.
```

## 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 netstorage-autofs
```

This concludes the section.

## ► Lab

# Access Network-Attached Storage

### Performance Checklist

In this lab, you configure the automounter with an indirect map, using exports from an NFSv4 server.

### Outcomes

- Install required packages to set up the automounter.
- Configure an automounter indirect map, with resources from a preconfigured NFSv4 server.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This start script determines whether the `servera` and `serverb` systems are reachable on the network. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. The script also creates users and groups that are needed on both `servera` and `serverb` systems.

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

### Instructions

An IT support company uses a central server, `serverb`, to host some exported directories on `/shares` for their groups and users. Users must be able to log in and have their exported directories mounted on demand and ready to use, under the `/remote` directory on `servera`.

Environment Characteristics:

- The `serverb` machine is sharing the `/shares` directory, which in turn contains the `management`, `production`, and `operation` subdirectories.
- The `managers` group consists of the `manager1` and `manager2` users. They have read and write access to the `/shares/management` exported directory.
- The `production` group consists of the `dbuser1` and `sysadmin1` users. They have read and write access to the `/shares/production` exported directory.
- The `operators` group consists of the `contractor1` and `consultant1` users. They have read and write access to the `/shares/operation` exported directory.
- The main mount point for `servera` is the `/remote` directory.
- Use the `/etc/auto.master.d/shares.autofs` file as the master map file and the `/etc/auto.shares` file as the indirect map file.
- The `/shares/management` exported directory is automounted on `/remote/management` on `servera`.

- The `/shares/production` exported directory is automounted on `/remote/production` on `servera`.
  - The `/shares/operation` exported directory is automounted on `/remote/operation` on `servera`.
  - All user passwords are set to `redhat`.
1. Log in to `servera` and install the required packages.
  2. Configure an automounter indirect map on `servera` with exports from `serverb`. Create an indirect map with files that are named `/etc/auto.master.d/shares.autofs` for the master map and `/etc/auto.shares` for the mapping file. Use the `/remote` directory as the main mount point on `servera`. Reboot `servera` to determine whether the `autofs` service starts automatically.
  3. Test the `autofs` configuration with the various users. When done, log out from `servera`.

## Evaluation

On the `workstation` machine, use the `lab` command to confirm success of this exercise.

```
[student@workstation ~]$ lab grade netstorage-review
```

## 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 netstorage-review
```

This concludes the section.

## ► Solution

# Access Network-Attached Storage

### Performance Checklist

In this lab, you configure the automounter with an indirect map, using exports from an NFSv4 server.

### Outcomes

- Install required packages to set up the automounter.
- Configure an automounter indirect map, with resources from a preconfigured NFSv4 server.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This start script determines whether the `servera` and `serverb` systems are reachable on the network. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. The script also creates users and groups that are needed on both `servera` and `serverb` systems.

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

### Instructions

An IT support company uses a central server, `serverb`, to host some exported directories on `/shares` for their groups and users. Users must be able to log in and have their exported directories mounted on demand and ready to use, under the `/remote` directory on `servera`.

Environment Characteristics:

- The `serverb` machine is sharing the `/shares` directory, which in turn contains the `management`, `production`, and `operation` subdirectories.
- The `managers` group consists of the `manager1` and `manager2` users. They have read and write access to the `/shares/management` exported directory.
- The `production` group consists of the `dbuser1` and `sysadmin1` users. They have read and write access to the `/shares/production` exported directory.
- The `operators` group consists of the `contractor1` and `consultant1` users. They have read and write access to the `/shares/operation` exported directory.
- The main mount point for `servera` is the `/remote` directory.
- Use the `/etc/auto.master.d/shares.autofs` file as the master map file and the `/etc/auto.shares` file as the indirect map file.
- The `/shares/management` exported directory is automounted on `/remote/management` on `servera`.

- The /shares/production exported directory is automounted on /remote/production on servera.
- The /shares/operation exported directory is automounted on /remote/operation on servera.
- All user passwords are set to redhat.

- Log in to servera and install the required packages.

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

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

- Install the autoofs package.

```
[root@servera ~]# dnf install autoofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- Configure an automounter indirect map on servera with exports from serverb. Create an indirect map with files that are named /etc/auto.master.d/shares.autoofs for the master map and /etc/auto.shares for the mapping file. Use the /remote directory as the main mount point on servera. Reboot servera to determine whether the autoofs service starts automatically.

- Test the NFS server before you configure the automounter.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrwx--- 2 root managers 25 Apr  4 01:13 management
drwxrwx--- 2 root operators 25 Apr  4 01:13 operation
drwxrwx--- 2 root production 25 Apr  4 01:13 production
[root@servera ~]# umount /mnt
```

- Create a master map file named /etc/auto.master.d/shares.autoofs, insert the following content, and save the changes.

```
/remote /etc/auto.shares
```

- Create an indirect map file named /etc/auto.shares, insert the following content, and save the changes.

```
* -rw, sync, fstype=nfs4 serverb.lab.example.com:/shares/ &
```

2.4. Start and enable the `autofs` service on `servera`.

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

2.5. Reboot the `servera` machine.

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

3. Test the `autofs` configuration with the various users. When done, log out from `servera`.3.1. After the `servera` machine is finished booting, log in to `servera` as the `student` user.

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

3.2. Switch to the `manager1` user and test access.

```
[student@servera ~]$ su - manager1
Password: redhat
[manager1@servera ~]$ ls -l /remote/management/
total 4
-rw-r--r--. 1 root managers 46 Apr  4 01:13 Welcome.txt
[manager1@servera ~]$ cat /remote/management>Welcome.txt
###Welcome to Management Folder on SERVERB###
[manager1@servera ~]$ echo TEST1 > /remote/management/Test.txt
[manager1@servera ~]$ cat /remote/management/Test.txt
TEST1
[manager1@servera ~]$ ls -l /remote/operation/
ls: cannot open directory '/remote/operation/': Permission denied
[manager1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[manager1@servera ~]$ exit
logout
[student@servera ~]$
```

3.3. Switch to the `dbuser1` user and test access.

```
[student@servera ~]$ su - dbuser1
Password: redhat
[dbuser1@servera ~]$ ls -l /remote/production/
total 4
-rw-r--r--. 1 root production 46 Apr  4 01:13 Welcome.txt
[dbuser1@servera ~]$ cat /remote/production>Welcome.txt
###Welcome to Production Folder on SERVERB###
[dbuser1@servera ~]$ echo TEST2 > /remote/production/Test.txt
[dbuser1@servera ~]$ cat /remote/production/Test.txt
TEST2
[dbuser1@servera ~]$ ls -l /remote/operation/
```

```
ls: cannot open directory '/remote/operation/': Permission denied
[dbuser1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
[dbuser1@servera ~]$ exit
logout
[student@servera ~]$
```

3.4. Switch to the contractor1 user and test access.

```
[student@servera ~]$ su - contractor1
Password: redhat
[contractor1@servera ~]$ ls -l /remote/operation/
total 4
-rw-r--r-- 1 root operators 45 Apr  4 01:13 Welcome.txt
[contractor1@servera ~]$ cat /remote/operation/Welcome.txt
###Welcome to Operation Folder on SERVERB###
[contractor1@servera ~]$ echo TEST3 > /remote/operation/Test.txt
[contractor1@servera ~]$ cat /remote/operation/Test.txt
TEST3
[contractor1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
[contractor1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[contractor1@servera ~]$ exit
logout
[student@servera ~]$
```

3.5. Explore the mount options for the NFS automounted export.

```
[student@servera ~]$ mount | grep nfs
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
serverb.lab.example.com:/shares/management on /remote/management type nfs4
(rw,relatime,vers=4.2,rsize=262144,wszie=262144,namlen=255,
sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/operation on /remote/operation type nfs4
(rw,relatime,vers=4.2,rszie=262144,wszie=262144,namlen=255,
sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/production on /remote/production type nfs4
(rw,relatime,vers=4.2,rszie=262144,wszie=262144,namlen=255,
sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
local_lock=none,addr=172.25.250.11)
```

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

```
[student@servera ~]$ exit
logout
[student@workstation ~]$
```

## Evaluation

On the workstation machine, use the lab command to confirm success of this exercise.

```
[student@workstation ~]$ lab grade netstorage-review
```

## 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 netstorage-review
```

This concludes the section.

# Summary

---

- You can temporarily mount and unmount NFS shares from the command line.
- You can mount and unmount NFS shares persistently by updating the /etc/fstab file.
- You can configure the automounter service to automatically mount NFS shares with direct and indirect maps.



## Chapter 15

# Manage Network Security

### Goal

Control network connections to services using the system firewall.

### Objectives

- Accept or reject network connections to system services with firewalld rules.

### Sections

- Manage Server Firewalls (and Guided Exercise)

### Lab

- Manage Network Security

# Manage Server Firewalls

## Objectives

Accept or reject network connections to system services with `firewalld` rules.

## 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 `netfilter` framework includes hooks for kernel modules to interact with network packets as they traverse a system's network stack. Fundamentally, `netfilter` hooks are kernel routines that intercept events (for example, a packet entering an interface) and run other related routines (for example, firewall rules).

## The `nftables` Framework

The `nftables` packet classification framework builds upon the `netfilter` framework to apply firewall rules to network traffic. In Red Hat Enterprise Linux 9, `nftables` is the system firewall core, and it replaces the deprecated `iptables` framework.

The `nftables` framework provides numerous advantages over `iptables`, including improved usability and more efficient rule sets. For example, the `iptables` framework required a rule for each protocol, but `nftables` rules can apply to both IPv4 and IPv6 traffic simultaneously. The `iptables` framework required using different tools, such as `iptables`, `ip6tables`, `arptables`, and `ebtables`, for each protocol, but the `nftables` framework uses the single `nft` user-space utility to manage all protocols through a single interface.



### Note

Convert legacy `iptables` configuration files into their `nftables` equivalents by using the `iptables-translate` and `ip6tables-translate` utilities.

## The `firewalld` Service

The `firewalld` service is a dynamic firewall manager, and is the recommended front end to the `nftables` framework. The Red Hat Enterprise Linux 9 distribution 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 laptops or other machines that regularly change networks, the `NetworkManager` service can automatically set the firewall zone for a connection. This is useful when switching between home, work, and 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 checks the source address for every packet coming into the system. If that 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 `firewalld` 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 `firewalld` sends the packet to the default zone.

The default zone is not a separate zone but rather a designation assigned to an existing zone. Initially, the `firewalld` service designates the `public` zone as default, and maps the `lo` loopback interface to the `trusted` zone.

Most zones allow traffic through the firewall, which matches a list of particular ports and protocols, such as `631/udp`, or a predefined service configuration, such as `ssh`. Normally, if the traffic does not match a permitted port and protocol or service, then it is rejected. The `trusted` zone, which permits all traffic by default, is an exception.

## Predefined Zones

The `firewalld` service uses predefined zones, which you can customize. By default, all zones permit any incoming traffic which is part of an existing session initiated by the system, and also all outgoing traffic. The following table details the initial zone configuration.

### Default Configuration of Firewalld Zones

Zone name	Default configuration
<code>trusted</code>	Allow all incoming traffic.
<code>home</code>	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.
<code>internal</code>	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 <code>home</code> zone to start with).
<code>work</code>	Reject incoming traffic unless related to outgoing traffic or matching the <code>ssh</code> , <code>ipp-client</code> , or <code>dhcpcv6-client</code> predefined services.
<code>public</code>	Reject incoming traffic unless related to outgoing traffic or matching the <code>ssh</code> or <code>dhcpcv6-client</code> predefined services. <i>The default zone for newly added network interfaces.</i>

Zone name	Default configuration
external	Reject incoming traffic unless related to outgoing traffic or matching the ssh predefined service. Outgoing IPv4 traffic forwarded through this zone is <i>masqueraded</i> to look like 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 a number of predefined configurations for common services, to simplify setting firewall rules. For example, instead of researching the relevant ports for an NFS server, use the predefined nfs configuration create rules for the correct ports and protocols. The following table lists the predefined service configurations that the `firewalld` service uses in its default configuration.

### Selected Predefined Firewalld Services

Service name	Configuration
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 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.

The `firewalld` package includes many predefined service configurations. You can list the services with the `firewall-cmd --get-services` command.

```
[root@host ~]# firewall-cmd --get-services
RH-Satellite-6 RH-Satellite-6-capsule amanda-client amanda-k5-client amqp amqps
apcupsd audit bacula bacula-client bb bgp bitcoin bitcoin-rpc bitcoin-testnet
bitcoin-testnet-rpc bittorrent-lsd ceph ceph-mon cfengine 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 additional ports and protocols.

## Configure the firewalld Daemon

Among others, two common ways that system administrators use to interact with the `firewalld` service are:

- The web console graphical interface
- The `firewall-cmd` command-line tool

## Configure Firewall Services Using the Web Console

To manage firewall services with the web console, you need to log in and escalate privileges. You can 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. As a security reminder, remember to toggle back to limited access mode once you perform on your system the task that requires administrative privileges.

Click the **Networking** option in the left navigation menu to display the **Firewall** section in the main networking page. Click the **Edit rules and zones** button zones to navigate to the **Firewall** page.

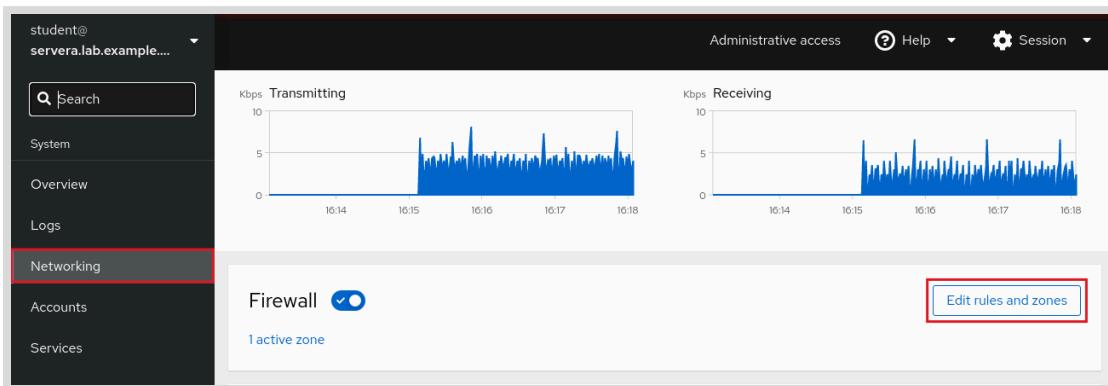


Figure 15.1: The web console networking page

The **Firewall** page displays active zones and their allowed services. Click the arrow (>) button 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.

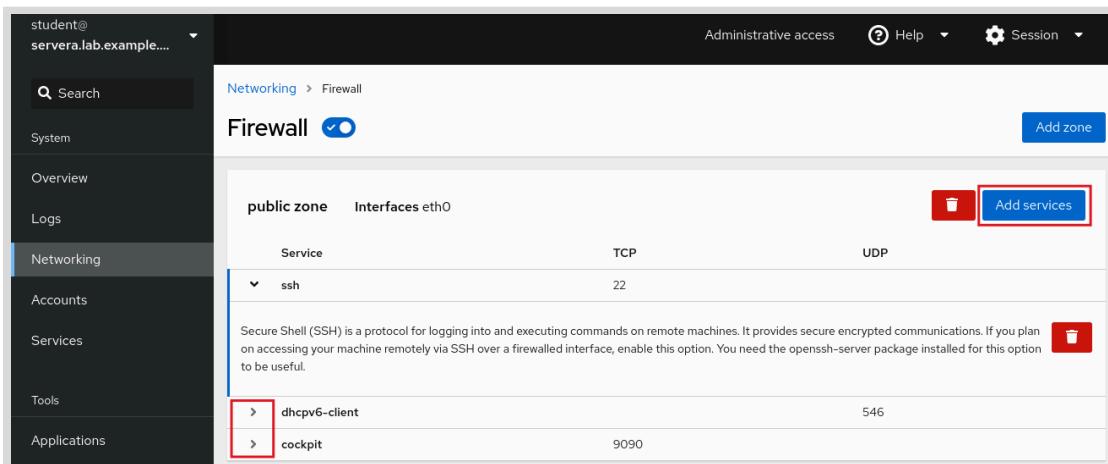


Figure 15.2: The web console firewall page

The Add Services page displays the available predefined services.

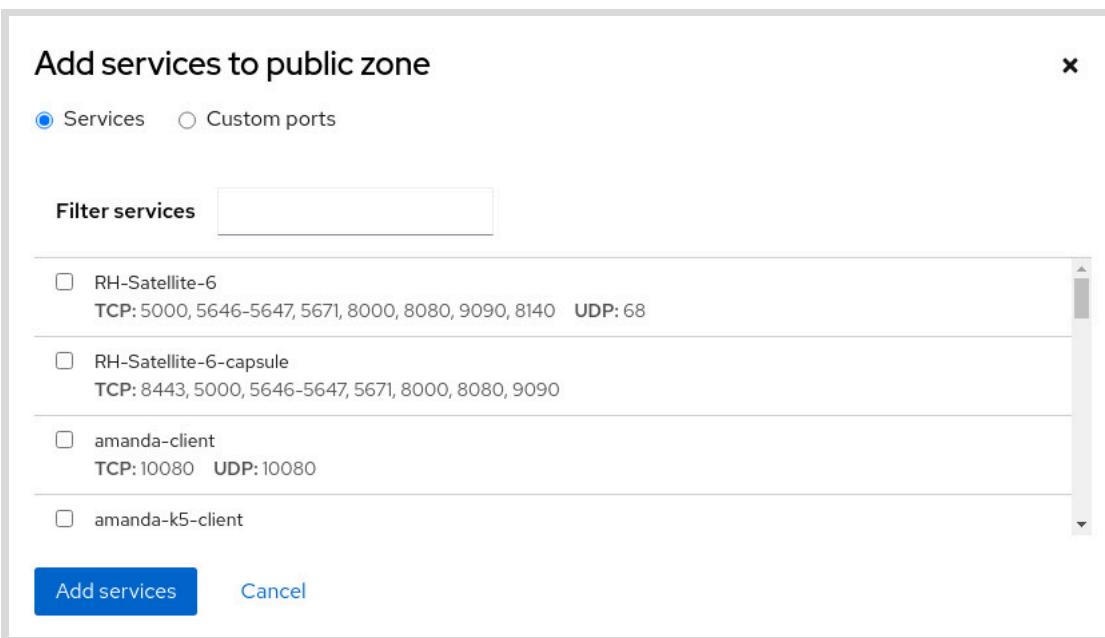


Figure 15.3: The web console add services menu

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 web related services. Select the check box to the left of the service to allow it through the firewall. Click the **Add services** button to complete the process.

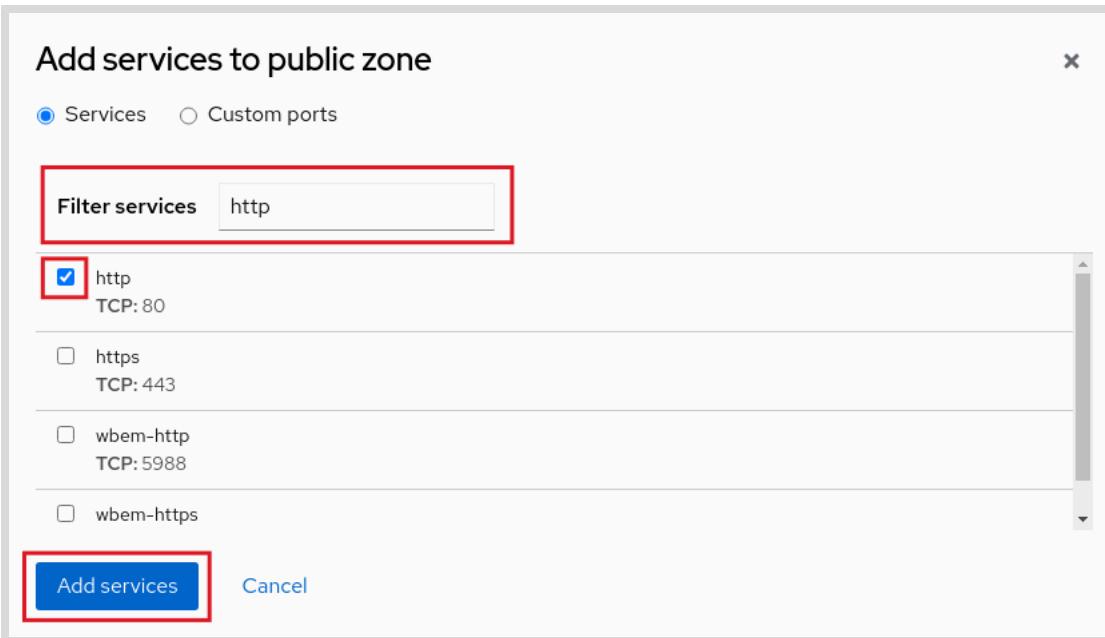


Figure 15.4: The web console add services menu options

The interface returns to the Firewall page, where you can review the updated allowed services list.

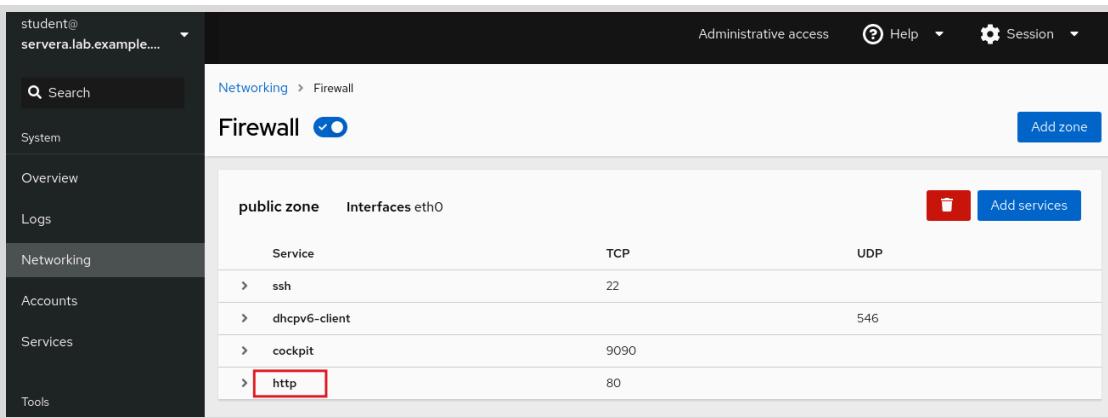


Figure 15.5: The web console firewall overview

## Configure the Firewall from the Command Line

The `firewall-cmd` command interfaces 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, for working on systems without a graphical environment, or for scripting a firewall set up.

The following table lists frequently used `firewall-cmd` commands, along with an explanation. Note that unless otherwise specified, almost all 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 --reload` command, which reads the current permanent configuration and applies it as the new runtime configuration. Many of the commands listed take the `--zone=ZONE` option to determine which zone they affect. Where a netmask is required, use CIDR notation, such as 192.168.1/24.

firewall-cmd commands	Explanation
<code>--get-default-zone</code>	Query the current default zone.
<code>--set-default-zone=ZONE</code>	Set the default zone. This changes both the runtime and the permanent configuration.
<code>--get-zones</code>	List all available zones.
<code>--get-active-zones</code>	List all zones currently in use (have an interface or source tied to them), along with their interface and source information.
<code>--add-source=CIDR [ --zone=ZONE]</code>	Route all traffic coming from the IP address or network/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 routing all traffic from the zone coming from the IP address or network. If no <code>--zone=</code> option is provided, then the default zone is used.
<code>--add-interface=INTERFACE [ --zone=ZONE]</code>	Route all traffic coming from <i>INTERFACE</i> to the specified zone. If no <code>--zone=</code> option is provided, then the default zone is used.

firewall-cmd commands	Explanation
--change-interface=INTERFACE [--zone=ZONE]	Associate the interface with ZONE instead of its current zone. If no --zone= option is provided, then the default zone is used.
--list-all [--zone=ZONE]	List all configured interfaces, sources, services, and ports for ZONE. If no --zone= option is provided, then the default zone is used.
--list-all-zones	Retrieve all information for all zones (interfaces, sources, ports, services).
--add-service=SERVICE [--zone=ZONE]	Allow traffic to SERVICE. If no --zone= option is provided, then the default zone is used.
--add-port=PORT/PROTOCOL [--zone=ZONE]	Allow traffic to the PORT/PROTOCOL port(s). If no --zone= option is provided, then the default zone is used.
--remove-service=SERVICE [--zone=ZONE]	Remove SERVICE from the allowed list for the zone. If no --zone= option is provided, then the default zone is used.
--remove-port=PORT/PROTOCOL [--zone=ZONE]	Remove the PORT/PROTOCOL port(s) from the allowed list for the zone. If no --zone= option is provided, then the default zone is used.
--reload	Drop the runtime configuration and apply the persistent configuration.

The following example sets the default zone to dmz, assigns all traffic coming from the 192.168.0.0/24 network to the internal zone, and opens the network ports for the mysql service on the internal zone.

```
[root@host ~]# firewall-cmd --set-default-zone=dmz
[root@host ~]# firewall-cmd --permanent --zone=internal \
--add-source=192.168.0.0/24
[root@host ~]# firewall-cmd --permanent --zone=internal --add-service=mysql
[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 also use Direct Configuration rules (raw nft syntax mixed in with firewalld rules). These advanced configurations are beyond the scope of this chapter.



## References

`firewall-cmd(1)`, `firewalld(1)`, `firewalld.zone(5)`, `firewalld.zones(5)`,  
and `nft(8)` man pages

## ► Guided Exercise

# Manage Server Firewalls

In this exercise, you control access to system services by adjusting system firewall rules with `firewalld`.

### Outcomes

- Configure firewall rules to control access to services.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start netsecurity-firewalls
```

### 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. 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!
```

- 3. Create the `/var/www/html/index.html` file. Add one line of text that reads: 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. Return to the workstation machine as the student user.

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

- 6. From workstation, attempt to access the web server on servera by using both the clear-text port 80/TCP and the SSL encapsulated port 443/TCP. Both attempts should fail.

- 6.1. The curl command should fail.

```
[student@workstation ~]$ curl http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80: No route to host
```

- 6.2. The curl command with the -k option for insecure connections should also fail.

```
[student@workstation ~]$ curl -k https://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 443: No route to host
```

- 7. Verify that the firewalld service on servera is enabled and running.

```
[student@workstation ~]$ ssh student@servera 'systemctl status firewalld'
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor
  preset: enabled)
  Active: active (running) since Wed 2022-04-13 11:22:50 EDT; 7min ago
    Docs: man:firewalld(1)
  Main PID: 768 (firewalld)
     Tasks: 2 (limit: 10798)
    Memory: 39.9M
       CPU: 584ms
      CGroup: /system.slice/firewalld.service
              └─768 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

Apr 13 11:22:49 servera.lab.example.com systemd[1]: Starting firewalld - dynamic
firewall daemon...
Apr 13 11:22:50 servera.lab.example.com systemd[1]: Started firewalld - dynamic
firewall daemon.
```

- 8. From workstation, open Firefox and log in to the web console running on servera to add the https service to the public firewall zone.

- For use by Rajkamal J rajkamal@redhat.com veerammarish89@gmail.com Copyright © 2022 Red Hat, Inc.
- 8.1. Open Firefox and browse to `https://servera.lab.example.com:9090` to access the web console. Click **Advanced** and **Accept the Risk and Continue** to accept the self-signed certificate.
  - 8.2. Log in as the **student** user and provide **student** as the password.
  - 8.3. Click **Turn on administrative access** and enter the **student** password again.
  - 8.4. Click **Networking** in the left navigation bar.
  - 8.5. Click **Edit rules and zones** in **Firewall** section of the **Networking** page.
  - 8.6. Click **Add services** located in the upper right corner of the **public zone** section.
  - 8.7. In the **Add services** interface, scroll down or use **Filter services** to locate and select the check box next to the **https** service.
  - 8.8. Click **Add services** to apply the change.
- ▶ 9. Return to a terminal on **workstation** and verify your work by attempting to access the **servera** web server.
- 9.1. The `curl` command to the standard port 80 should fail.

```
[student@workstation ~]$ curl http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80: No route to host
```

- 9.2. The `curl` command with the `-k` option to the port 443 should succeed.

```
[student@workstation ~]$ curl -k https://servera.lab.example.com
I am servera.
```

## 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 netsecurity-firewalls
```

This concludes the section.

## ► Lab

# Manage Network Security

In this lab, you configure firewall and SELinux settings to allow access to multiple web servers running on the same host.

## Outcomes

- Configure firewall and SELinux settings on a web server host.

## Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

Your company has decided to run a new web app. This application listens on ports 80/TCP and 1001/TCP. You should also make available port 22/TCP for ssh access. All changes you make should persist across a reboot.



### Important

Red Hat Online Learning environment needs port 5900/TCP to remain available to use the graphical interface. This port is also known under the `vnc-server` service. If you accidentally lock yourself out from the `serverb` machine, then you can either attempt to recover by using the `ssh` command to your `serverb` machine from your `workstation` machine, or reset your `serverb` machine. If you elect to reset your `serverb` machine, then you should run the setup scripts for this lab again. The configuration on your machines already includes a custom zone called `ROL` that opens these ports.

- From the `workstation` machine, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.
- Log in to the `serverb` machine to determine what is preventing access to the web servers.
- Configure SELinux to allow the `httpd` service to listen on port 1001/TCP.
- From `workstation`, test again access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.
- Log in to the `serverb` machine to determine whether the correct ports are assigned to the firewall.
- Add port 1001/TCP to the permanent configuration for the `public` network zone. Confirm your configuration.

7. From **workstation**, confirm that the default web server at **serverb.lab.example.com** returns SERVER\_B and the virtual host at **serverb.lab.example.com:1001** 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, 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 netsecurity-review
```

This concludes the section.

## ► Solution

# Manage Network Security

In this lab, you configure firewall and SELinux settings to allow access to multiple web servers running on the same host.

### Outcomes

- Configure firewall and SELinux settings on a web server host.

### Before You Begin

As the student user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

### Instructions

Your company has decided to run a new web app. This application listens on ports 80/TCP and 1001/TCP. You should also make available port 22/TCP for ssh access. All changes you make should persist across a reboot.



#### Important

Red Hat Online Learning environment needs port 5900/TCP to remain available to use the graphical interface. This port is also known under the `vnc-server` service. If you accidentally lock yourself out from the `serverb` machine, then you can either attempt to recover by using the `ssh` command to your `serverb` machine from your `workstation` machine, or reset your `serverb` machine. If you elect to reset your `serverb` machine, then you should run the setup scripts for this lab again. The configuration on your machines already includes a custom zone called `ROL` that opens these ports.

- From the `workstation` machine, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.
  - Test access to the `http://serverb.lab.example.com` web server. The test currently fails. Ultimately, the web server should return SERVER B.

```
[student@workstation ~]$ curl http://serverb.lab.example.com
curl: (7) Failed to connect to serverb.lab.example.com port 80: Connection refused
```

- Test access to the `http://serverb.lab.example.com:1001` virtual host. The test currently fails. Ultimately, the virtual host should return VHOST 1.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No route to host
```

2. Log in to the **serverb** machine to determine what is preventing access to the web servers.

- 2.1. Log in to the **serverb** machine as **student** user.

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

- 2.2. Determine whether the **httpd** service is active.

```
[student@serverb ~]$ systemctl is-active httpd
inactive
```

- 2.3. Enable and start the **httpd** service. The **httpd** service fails to start.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
[sudo] password for student: student
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. Investigate the reasons why the **httpd** service fails to start.

```
[student@serverb ~]$ systemctl status httpd.service
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
   preset: disabled)
     Active: failed (Result: exit-code) since Wed 2022-04-13 06:55:01 EDT; 2min
      52s ago
       Docs: man:httpd.service(8)
     Process: 1640 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
    status=1/FAILURE)
       Main PID: 1640 (code=exited, status=1/FAILURE)
          Status: "Reading configuration..."
            CPU: 31ms

Apr 13 06:55:01 serverb.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: (13)Permission denied:
AH00072: make_sock: could not bind to address [::]:1001
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: (13)Permission denied:
AH00072: make_sock: could not bind to address 0.0.0.0:1001
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: no listening sockets
available, shutting down
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: AH00015: Unable to open logs
```

```
Apr 13 06:55:01 serverb.lab.example.com systemd[1]: httpd.service: Main process
exited, code=exited, status=1/FAILURE
Apr 13 06:55:01 serverb.lab.example.com systemd[1]: httpd.service: Failed with
result 'exit-code'.
Apr 13 06:55:01 serverb.lab.example.com systemd[1]: Failed to start The Apache
HTTP Server.
```

2.5. Check whether SELinux is blocking the `httpd` service from binding to port 1001/TCP.

```
[student@serverb ~]$ sudo 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.

***** Plugin catchall (1.49 confidence) suggests *****

...output omitted...
```

3. Configure SELinux to allow the `httpd` service to listen on port 1001/TCP.

3.1. Use the `semanage` command to find the correct port type.

```
[student@serverb ~]$ sudo 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 port 1001/TCP to the `http_port_t` type.

```
[student@serverb ~]$ sudo semanage port -a -t http_port_t -p tcp 1001
```

3.3. Confirm that port 1001/TCP is bound to the `http_port_t` port type.

```
[student@serverb ~]$ sudo 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.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
```

- 3.5. Verify the running state of the `httpd` service.

```
[student@serverb ~]$ systemctl is-active httpd  
active  
[student@serverb ~]$ systemctl is-enabled httpd  
enabled
```

- 3.6. Return to the `workstation` machine as the `student` user.

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

4. From `workstation`, test again access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.

- 4.1. Test access to the `http://serverb.lab.example.com` web server. The web server should return SERVER B.

```
[student@workstation ~]$ curl http://serverb.lab.example.com  
SERVER B
```

- 4.2. Test access to the `http://serverb.lab.example.com:1001` virtual host. The test continues to fail.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001  
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No route to host
```

5. Log in to the `serverb` machine to determine whether the correct ports are assigned to the firewall.

- 5.1. Log in to the `serverb` machine as the `student` user.

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

- 5.2. Verify that the default firewall zone is set to the `public` zone.

```
[student@serverb ~]$ firewall-cmd --get-default-zone  
public
```

- 5.3. If the previous step does not return `public` as the default zone, then correct it with the following command:

```
[student@serverb ~]$ sudo firewall-cmd --set-default-zone public
```

- 5.4. Determine the open ports listed in the `public` network zone.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --list-all
[sudo] password for student: student
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client http ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

6. Add port 1001/TCP to the permanent configuration for the public network zone. Confirm your configuration.

- 6.1. Add port 1001/TCP to the public network zone.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public \
--add-port=1001/tcp
success
```

- 6.2. Reload the firewall configuration.

```
[student@serverb ~]$ sudo firewall-cmd --reload
success
```

- 6.3. Verify your configuration.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client http ssh
  ports: 1001/tcp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

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

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

7. From **workstation**, confirm that the default web server at `serverb.lab.example.com` returns SERVER B and the virtual host at `serverb.lab.example.com:1001` returns VHOST 1.

7.1. Test access to the `http://serverb.lab.example.com` web server.

```
[student@workstation ~]$ curl http://serverb.lab.example.com  
SERVER B
```

7.2. Test access to the `http://serverb.lab.example.com:1001` virtual host.

```
[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, 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 netsecurity-review
```

This concludes the section.

# Summary

---

- With the **netfilter** framework, kernel modules can inspect every packet that traverses the system, including all incoming, outgoing, or forwarded network packets.
- The **firewalld** service simplifies management by classifying all network traffic into zones. Each zone has its own list of ports and services. The public zone is set as the default zone.
- The **firewalld** service ships with various predefined services. You can list them by using the `firewall-cmd --get-services` command.



## Chapter 16

# Run Containers

### Goal

Obtain, run, and manage simple lightweight services as containers on a single Red Hat Enterprise Linux server.

### Objectives

- Explain container concepts and the core technologies for building, storing, and running containers.
- Discuss container management tools for using registries to store and retrieve images, and for deploying, querying, and accessing containers.
- Provide persistent storage for container data by sharing storage from the container host, and configure a container network.
- Configure a container as a `systemd` service, and configure a container service to start at boot time.

### Sections

- Container Concepts (and Quiz)
- Deploy Containers (and Guided Exercise)
- Manage Container Storage and Network Resources (and Guided Exercise)
- Manage Containers as System Services (and Guided Exercise)

### Lab

Run Containers

# Container Concepts

## Objectives

Explain container concepts and the core technologies for building, storing, and running containers.

## Container Technology

Software applications typically depend on system libraries, configuration files, or services that their runtime environment provides. Traditionally, the runtime environment for a software application is installed in an operating system that runs on a physical host or virtual machine. Administrators then install application dependencies on top of the operating system.

In Red Hat Enterprise Linux, packaging systems such as RPM help administrators to manage application dependencies. When you install the `httpd` package, the RPM system ensures that the correct libraries and other dependencies for that package are also installed.

The major drawback to traditionally deployed software applications is that these dependencies are entangled with the runtime environment. An application might require older or newer versions of supporting software than the software that is provided with the operating system. Similarly, two applications on the same system might require different and incompatible versions of the same software.

One way to resolve these conflicts is to package and deploy the application as a *container*. A container is a set of one or more processes that are isolated from the rest of the system. Software containers provide a way to package applications and to simplify their deployment and management.

Think of a physical shipping container. A shipping container is a standard way to package and ship goods. It is labeled, loaded, unloaded, and transported from one location to another as a single box. The container's contents are isolated from the contents of other containers so that they do not affect each other. These underlying principles also apply to software containers.

Red Hat Enterprise Linux supports containers by using the following core technologies:

- *Control Groups (cgroups)* for resource management.
- *Namespaces* for process isolation.
- SELinux and Seccomp (Secure Computing mode) to enforce security boundaries.



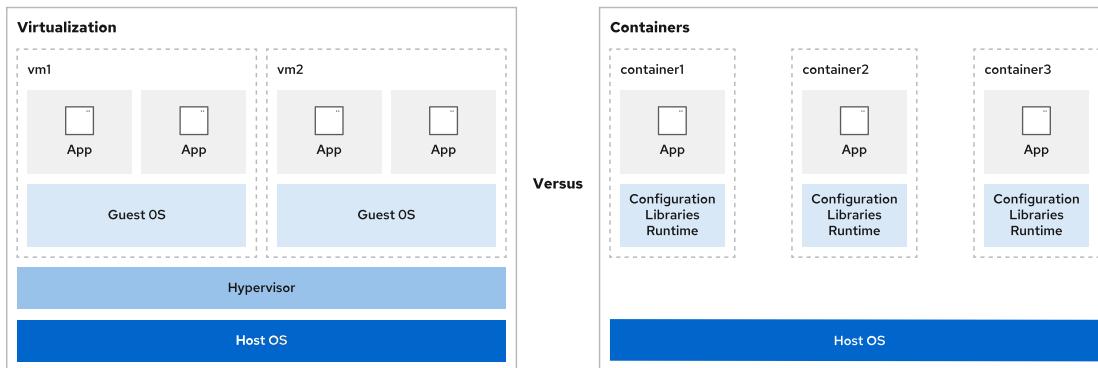
### Note

For a more in-depth discussion of container architecture and security, refer to the "Ten layers of container security" [<https://www.redhat.com/en/resources/container-securityOpenshiftCloudDevOpsWhitepaper>] white paper.

## Differences Between Containers and Virtual Machines

Containers provide many of the same benefits as virtual machines, such as security, storage, and network isolation.

Both technologies isolate their application libraries and runtime resources from the host operating system or hypervisor and vice versa.



**Figure 16.1: Comparison between virtualization and containerization**

Containers and virtual machines interact differently with hardware and the underlying operating system.

A virtual machine has the following characteristics:

- Enables multiple operating systems to run simultaneously on a single hardware platform.
- Uses a hypervisor to divide hardware into multiple virtual hardware systems.
- Requires a complete operating system environment to support the application.

A container has the following characteristics:

- Runs directly on the host operating system, and it shares resources with all containers on the system.
- Shares the host's kernel, but it isolates the application processes from the rest of the system.
- Requires far fewer hardware resources than virtual machines, so containers are also quicker to start.
- Includes all dependencies such as system and programming dependencies, and configuration settings.



### Note

Some applications might not be suitable to run as a container. For example, applications that access low-level hardware information might need more direct hardware access than containers generally provide.

## Rootless and Rootful Containers

On the container host, you can run containers as the root user or as a regular, unprivileged user. Containers that a privileged user runs are called *rootful containers*. Containers that non-privileged users run are called *rootless containers*.

A rootless container is not allowed to use system resources that are usually reserved for privileged users, such as access to restricted directories, or to publish network services on restricted ports (those ports below 1024). This feature prevents a possible attacker from gaining root privileges on the container host.

You can run containers directly as `root` if necessary, but this scenario weakens the security of the system if a bug allows an attacker to compromise the container.

## Design a Container-based Architecture

Containers are an efficient way to reuse hosted applications and to make them portable. Containers can be easily moved from one environment to another, such as from development to production. You can save multiple versions of a container and quickly access each one as needed.

Containers are typically temporary, or *ephemeral*. You can permanently save in persistent storage the data that a running container generates, but the containers themselves usually run when needed, and then stop and are removed. A new container process is started the next time that particular container is needed.

You could install a complex software application with multiple services in a single container. For example, a web server might need to use a database and a messaging system. However, using one container for multiple services is hard to manage.

A better design runs in separate containers each component, the web server, the database, and the messaging system. This way, updates and maintenance to individual application components do not affect other components or the application stack.

## Container Management Tools

Red Hat Enterprise Linux provides a set of container tools that you can use to run containers in a single server.

- `podman` manages containers and container images.
- `skopeo` inspects, copies, deletes, and signs images.
- `buildah` creates container images.

These tools are compatible with the Open Container Initiative (OCI). With these tools, you can manage any Linux containers that are created by OCI-compatible container engines, such as Podman or Docker. These tools are specifically designed to run containers under Red Hat Enterprise Linux on a single-node container host.

In this chapter, you use the `podman` and `skopeo` utilities to run and manage containers and existing container images.



### Note

Using `buildah` to construct your own container images is beyond the scope of this course, but is covered in the Red Hat Training course *Red Hat OpenShift I: Containers & Kubernetes* (DO180).

## Container Images and Registries

To run containers, you must use a *container image*. A container image is a static file that contains codified steps, and it serves as a blueprint to create containers. The container images package an application with all its dependencies, such as its system libraries, programming language runtimes and libraries, and other configuration settings.

Container images are built according to specifications, such as the Open Container Initiative (OCI) image format specification. These specifications define the format for container images, as well as the metadata about the container host operating systems and hardware architectures that the image supports.

## Chapter 16 | Run Containers

A *container registry* is a repository for storing and retrieving container images. A developer *pushes* or uploads container images to a container registry. You can *pull* or download container images from a registry to a local system to run containers.

You might use a public registry that contains third-party images, or you might use a private registry that your organization controls. The source of your container images matters. As with any other software package, you must know whether you can trust the code in the container image. Policies vary between registries about whether and how they provide, evaluate, and test container images that are submitted to them.

Red Hat distributes certified container images through two main container registries that you can access with your Red Hat login credentials.

- `registry.redhat.io` for containers that are based on official Red Hat products.
- `registry.connect.redhat.com` for containers that are based on third-party products.

The Red Hat Container Catalog (<https://access.redhat.com/containers>) provides a web-based interface to search these registries for certified content.



### Note

Red Hat provides the *Universal Base Image (UBI)* image as an initial layer to build containers. The UBI image is a minimized container image that can be a first layer for an application build.

You need a Red Hat Developer account to download an image from the Red Hat registries. You can use the `podman login` command to authenticate to the registries. If you do not provide a registry URL to the `podman login` command, then it authenticates to the default configured registry.

```
[user@host ~]$ podman login registry.lab.example.com
Username: RH134
Password: EXAMPLEPASSWORD
Login Succeeded!
```

You can also use the `podman login` command `--username` and `--password-stdin` options, to specify the user and password to log in to the registry. The `--password-stdin` option reads the password from stdin. Red Hat does not recommend to use the `--password` option to provide the password directly, as this option stores the password in the log files.

```
[user@host ~]# echo $PASSWORDVAR | podman login --username RH134 \
--password-stdin registry.access.redhat.com
```

To verify that you are logged in to a registry, use the `podman login` command `--get-login` option.

```
[user01@rhel-vm ~]$ podman login registry.access.redhat.com --get-login
RH134
[user01@rhel-vm ~]$ podman login quay.io --get-login
Error: not logged into quay.io
```

In the preceding output, the podman utility is authenticated to the `registry.access.redhat.com` registry with the RH134 user credentials, but the podman utility is not authenticated to the `quay.io` registry.

## Configure Container Registries

The default configuration file for container registries is the `/etc/containers/registries.conf` file.

```
[user@host ~]$ cat /etc/containers/registries.conf
...output omitted...
[registries.search]
registries = ['registry.redhat.io', 'quay.io', 'docker.io']

# If you need to access insecure registries, add the registry's fully-qualified
# name.
# An insecure registry is one that does not have a valid SSL certificate or only
# does HTTP.
[registries.insecure]
registries = []
...output omitted...
```

Because Red Hat recommends to use a non-privileged user to manage containers, you can create a `registries.conf` file for container registries in the `$HOME/.config/containers` directory. The configuration file in this directory overrides the settings in the `/etc/containers/registries.conf` file.

The list of registries to look for containers is configured in the `[registries.search]` section of this file. If you specify the fully qualified name of a container image from the command line, then the container utility does not search in this section.

Insecure registries are listed in the `[registries.insecure]` section of the `registries.conf` file. If a registry is listed as insecure, then connections to that registry are not protected with TLS encryption. If a registry is both searchable and insecure, then it can be listed in both `[registries.search]` and `[registries.insecure]`.



### Note

This classroom runs a private insecure registry based on Red Hat Quay to provide container images. This registry meets the classroom need; however, you would not expect to work with insecure registries in real-world scenarios. For more information about this software, see <https://access.redhat.com/products/red-hat-quay>.

## Container Files to Build Container Images

A *container file* is a text file with the instructions to build a container image. A container file usually has a *context* that defines the path or URL where its files and directories are located. The resulting container image consists of read-only layers, where each layer represents an instruction from the container file.

The following output is an example of a container file that uses the UBI image from the `registry.access.redhat.com` registry, installs the `python3` package, and prints the `hello` string to the console.

```
[user@host ~]$ cat Containerfile
FROM registry.access.redhat.com/ubi8/ubi:latest
RUN dnf install -y python3
CMD ["/bin/bash", "-c", "echo hello"]
```

**Note**

Creating a container file and its usage instructions are out of scope for this course. For more information about container files, refer to the DO180 course.

## Container Management at Scale

New applications increasingly use containers to implement functional components. Those containers provide services that other parts of the application consume. In an organization, managing a growing number of containers might quickly become an overwhelming task.

Deploying containers at scale in production requires an environment that can adapt to the following challenges:

- The platform must ensure the availability of containers that provide essential services.
- The environment must respond to application usage spikes by increasing or decreasing the number of running containers and load balancing the traffic.
- The platform must detect the failure of a container or a host and react accordingly.
- Developers might need an automated workflow to deliver new application versions transparently and securely.

*Kubernetes* is an orchestration service that deploys, manages, and scales container-based applications across a cluster of container hosts. Kubernetes redirects traffic to your containers with a load balancer, so that you can scale the number of containers that provide a service. Kubernetes also supports user-defined health checks to monitor your containers and to restart them if they fail.

Red Hat provides a distribution of Kubernetes called *Red Hat OpenShift*. Red Hat OpenShift is a set of modular components and services that are built on top of the Kubernetes infrastructure. It provides additional features, such as remote web-based management, multitenancy, monitoring and auditing, advanced security features, application lifecycle management, and self-service instances for developers.

Red Hat OpenShift is beyond the scope of this course, but you can learn more about it at <https://www.openshift.com>.

**Note**

In the enterprise, individual containers are not generally run from the command line. Instead, it is preferable to run containers in production with a Kubernetes-based platform, such as Red Hat OpenShift.

However, you might need to use commands to manage containers and images manually or at a small scale. This chapter focuses on this use case to improve your grasp of the core concepts behind containers, how they work, and how they can be useful.



## References

cgroups(7), namespaces(7), seccomp(2) man pages.

### Open Container Initiative (OCI) Image Specification

<https://github.com/opencontainers/image-spec/blob/master/spec.md>

For more information, refer to the *Starting with containers* chapter in the *Red Hat Enterprise Linux 9 Building, Running, and Managing Containers Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/building\\_running\\_and\\_managing\\_containers/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/index)

## ► Quiz

# Container Concepts

Choose the correct answers to the following questions:

► 1. **Which Red Hat Enterprise Linux tool runs containers?**

- a. buildah
- b. container
- c. podman
- d. skopeo

► 2. **Which two statements describe container technology? (Choose two.)**

- a. Containers package complete operating systems, with the addition of library dependencies.
- b. Containers run processes that are isolated from the rest of the system.
- c. Each container includes its own kernel version and libraries.
- d. Containers provide a standard way to package applications to ease deployment and management.

► 3. **Which two statements are true about container images? (Choose two.)**

- a. Container images package an application with all of its needed runtime dependencies.
- b. Container images that work with Docker cannot work with Podman.
- c. Container images can run only on a container host with the same installed software version in the image.
- d. Container images serve as blueprints for creating containers.

► 4. **Which three core technologies are used to implement containers in Red Hat Enterprise Linux? (Choose three.)**

- a. Hypervisor code for hosting VMs.
- b. Control Groups (cgroups) for resource management.
- c. Namespaces for process isolation.
- d. Full operating system for compatibility with the container's host.
- e. SELinux and Seccomp for security.

► 5. **Which sentence is true about container files?**

- a. A container file is an executable file that runs a container.
- b. A container file is an executable file that builds a container image.
- c. A container file is a compressed file that contains libraries and configuration for a container.
- d. A container file is a text file with the instructions to build a container.
- e. A container file is a text file with the instructions to build a container image.

## ► Solution

# Container Concepts

Choose the correct answers to the following questions:

► 1. **Which Red Hat Enterprise Linux tool runs containers?**

- a. buildah
- b. container
- c. podman
- d. skopeo

► 2. **Which two statements describe container technology? (Choose two.)**

- a. Containers package complete operating systems, with the addition of library dependencies.
- b. Containers run processes that are isolated from the rest of the system.
- c. Each container includes its own kernel version and libraries.
- d. Containers provide a standard way to package applications to ease deployment and management.

► 3. **Which two statements are true about container images? (Choose two.)**

- a. Container images package an application with all of its needed runtime dependencies.
- b. Container images that work with Docker cannot work with Podman.
- c. Container images can run only on a container host with the same installed software version in the image.
- d. Container images serve as blueprints for creating containers.

► 4. **Which three core technologies are used to implement containers in Red Hat Enterprise Linux? (Choose three.)**

- a. Hypervisor code for hosting VMs.
- b. Control Groups (cgroups) for resource management.
- c. Namespaces for process isolation.
- d. Full operating system for compatibility with the container's host.
- e. SELinux and Seccomp for security.

► 5. **Which sentence is true about container files?**

- a. A container file is an executable file that runs a container.
- b. A container file is an executable file that builds a container image.
- c. A container file is a compressed file that contains libraries and configuration for a container.
- d. A container file is a text file with the instructions to build a container.
- e. A container file is a text file with the instructions to build a container image.

# Deploy Containers

---

## Objectives

Discuss container management tools for using registries to store and retrieve images, and for deploying, querying, and accessing containers.

## The Podman Utility

Podman is a fully featured container engine from the `container-tools` meta-package to manage *Open Container Initiative (OCI)* containers and images. The `podman` utility does not use a daemon to function, and so developers do not need a privileged user account on the system to start or stop containers. Podman provides multiple subcommands to interact with containers and images. The following list shows subcommands that are used in this section:

### Podman Commands

Command	Description
<code>podman-build</code>	Build a container image with a container file.
<code>podman-run</code>	Run a command in a new container.
<code>podman-images</code>	List images in local storage.
<code>podman-ps</code>	Print information about containers.
<code>podman-inspect</code>	Display configuration of a container, image, volume, network, or pod.
<code>podman-pull</code>	Download an image from a registry.
<code>podman-cp</code>	Copy files or folders between a container and the local file system.
<code>podman-exec</code>	Execute a command in a running container.
<code>podman-rm</code>	Remove one or more containers.
<code>podman-rmi</code>	Remove one or more locally stored images.
<code>podman-search</code>	Search a registry for an image.

To cover the topics in this lecture, imagine the following scenario.

As a system administrator, you are tasked to run a container that is based on the RHEL 8 UBI container image called `python38` with the `python-38` package. You are also tasked to create a container image from a container file and run a container called `python36` from that container image. The container image that is created with the container file must have the `python36:1.0` tag. Identify the differences between the two containers. Also ensure that the installed `python` packages in the containers do not conflict with the installed Python version in your local machine.

## Install Container Utilities

The `container-tools` meta-package contains required utilities to interact with containers and container images. To download, run, and compare containers on your system, you install the `container-tools` meta-package with the `dnf install` command. Use the `dnf info` command to view the version and contents of the `container-tools` package.

```
[root@host ~]# dnf install container-tools
...output omitted...
[user@host ~]$ dnf info container-tools
...output omitted...
Summary      : A meta-package which container tools such as podman, buildah,
               : skopeo, etc.
License       : MIT
Description   : Latest versions of podman, buildah, skopeo, runc, common, CRIU,
               : Uidica, etc as well as dependencies such as container-selinux
               : built and tested together, and updated.
...output omitted...
```

The `container-tools` meta-package provides the needed `podman` and `skopeo` utilities to achieve the assigned tasks.

## Download a Container Image from a Registry

First, you ensure that the `podman` utility is configured to search and download containers from the `registry.redhat.io` registry. The `podman info` command displays configuration information of the `podman` utility, including the configured registries.

```
[user@host ~]$ podman info
...output omitted...
insecure registries:
  registries: []
registries:
  registries:
    - registry.redhat.io
    - quay.io
    - docker.io
...output omitted...
```

The `podman search` command searches for a matching image in the list of configured registries. By default, Podman searches in all unqualified-search registries. Depending on the Docker distribution API that is implemented with the registry, some registries might not support the search feature.

Use the `podman search` command to display a list of images on the configured registries that contain the `python-38` package.

```
[user@host ~]$ podman search python-38
NAME                                     DESCRIPTION
registry.access.redhat.com/ubi7/python-38   Python 3.8 platform for building and
                                             running applications
registry.access.redhat.com/ubi8/python-38   Platform for building and running
                                             Python 3.8 applications
...output omitted...
```

The `registry.access.redhat.com/ubi8/python-38` image seems to match the criteria for the required container.

You can use the `skopeo inspect` command to examine different container image formats from a local directory or a remote registry without downloading the image. This command output displays a list of the available version tags, exposed ports of the containerized application, and metadata of the container image. You use the `skopeo inspect` command to verify that the image contains the required `python-38` package.

```
[user@host ~]$ skopeo inspect docker://registry.access.redhat.com/ubi8/python-38
{
    "Name": "registry.access.redhat.com/ubi8/python-38",
    "Digest":
    "sha256:c6e522cba2cf2b3ae4a875d5210fb94aa1e7ba71b6cebd902a4f4df73cb090b8",
    "RepoTags": [
        ...output omitted...
        "1-68",
        "1-77-source",
        "latest"
    ...output omitted...
    "name": "ubi8/python-38",
    "release": "86.1648121386",
    "summary": "Platform for building and running Python 3.8 applications",
    ...output omitted...
    ...output omitted...
```

The `registry.access.redhat.com/ubi8/python-38` image contains the required package and it is based on the required image. You use the `podman pull` command to download the selected image to the local machine. You can use the fully qualified name of the image from the preceding output to avoid ambiguity on container versions or registries.

```
[user@host ~]$ podman pull registry.access.redhat.com/ubi8/python-38
Trying to pull registry.access.redhat.com/ubi8/python-38:latest...
Getting image source signatures
Checking if image destination supports signatures
Copying blob c530010fb61c done
...output omitted...
```

Then, you use the `podman images` command to display the local images.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
registry.access.redhat.com/ubi8/python-38	latest	a33d92f90990	1 hour ago	901 MB

## Create a Container Image from a Container File

You are provided with the following container file to create the container image in the `python36-app` directory:

```
[user@host python36-app]$ cat Containerfile
FROM registry.access.redhat.com/ubi8/ubi:latest
RUN dnf install -y python36
CMD ["/bin/bash", "-c", "sleep infinity"]
```

The previous container file uses the `registry.access.redhat.com/ubi8/ubi:latest` image as the base image. The container file then installs the `python36` package and runs the `sleep infinity` bash command to prevent the container from exiting.

Normally, a container runs a process, and then exits after that process is complete. The `sleep infinity` command prevents the container from exiting, because the process never completes. You can then test, develop, and debug inside the container.

After examining the container file, you use the `podman build` command to build the image. The `podman build` command builds a container image by using instructions from one or more container files. You must be in the directory with the container file to build the image with the `podman build` command. You can use the `podman build` command `-t` option to provide the name and `python36:1.0` tag for the new image.

```
[user@host python36-app]$ podman build -t python36:1.0 .
STEP 1/3: FROM registry.access.redhat.com/ubi8/ubi:latest
STEP 2/3: RUN dnf install -y python36
...output omitted...
STEP 3/3: CMD ["/bin/bash", "-c", "sleep infinity"]
COMMIT python36:1.0
--> 35ab820880f
Successfully tagged localhost/python36:1.0
35ab820880f1708fa310f835407ffc94cb4b4fe2506b882c162a421827b156fc
```

The last line of the preceding output shows the container image ID. Most Podman commands use the first 12 characters of the container image ID to refer to the container image. You can use this short ID or the name of a container or a container image as arguments for most Podman commands.



### Note

If a version number is not specified in the tag, then the image is created with the `:latest` tag. If an image name is not specified, then the image and tag fields show the `<none>` string.

You use the `podman images` command to verify that the image is created with the defined name and tag.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/python36	1.0	35ab820880f1	3 minute ago	266 MB
registry.access.redhat.com/ubi8/python-38	latest	a33d92f90990	1 hour ago	901 MB

You then use the `podman inspect` command to view the low-level information of the container image and verify that its content matches the requirements for the container.

```
[user@host ~]$ podman inspect localhost/python36:1.0
...output omitted...
    "Cmd": [
        "/bin/bash",
        "-c",
        "sleep infinity"
    ],
...output omitted...
{
    "created": "2022-04-18T19:47:52.708227513Z",
    "created_by": "/bin/sh -c dnf install -y python36",
    "comment": "FROM registry.access.redhat.com/ubi8/ubi:latest"
},
...output omitted...
```

The output of the `podman inspect` command shows the `registry.access.redhat.com/ubi8/ubi:latest` base image, the `dnf` command to install the `python36` package, and the `sleep infinity` bash command that is executed at runtime to prevent the container from exiting.

**Note**

The `podman inspect` command output varies from the `python-38` image to the `python36` image, because you created the `/python36` image by adding a layer with changes to the existing `registry.access.redhat.com/ubi8/ubi:latest` base image, whereas the `python-38` image is itself a base image.

## Run Containers

Now that you have the required container images, you can use them to run containers. A container can be in one of the following states:

**Created**

A container that is created but is not started.

**Running**

A container that is running with its processes.

**Stopped**

A container with its processes stopped.

**Paused**

A container with its processes paused. Not supported for rootless containers.

**Deleted**

A container with its processes in a dead state.

The `podman ps` command lists the running containers on the system. Use the `podman ps -a` command to view all containers (created, stopped, paused, or running) in the machine.

You use the `podman create` command to create the container to run later. To create the container, you use the ID of the `localhost/python36` container image. You also use the `--`

name option to set a name to identify the container. The output of the command is the long ID of the container.

```
[user@host ~]$ podman create --name python36 dd6ca291f097  
c54c7ee281581c198cb96b07d78a0f94be083ae94dacbae69c05bd8cd354bbec
```

**Note**

If you do not set a name for the container with the `podman create` or `podman run` command with the `--name` option, then the `podman` utility assigns a random name to the container.

You then use the `podman ps` and `podman ps -a` commands to verify that the container is created but is not started. You can see information about the `python36` container, such as the short ID, name, and the status of the container, the command that the container runs when started, and the image to create the container.

```
[user@host ~]$ podman create --name python36 dd6ca291f097  
c54c7ee281581c198cb96b07d78a0f94be083ae94dacbae69c05bd8cd354bbec  
[user@host ~]$ podman ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
[user@host ~]$ podman ps -a  
CONTAINER ID IMAGE COMMAND CREATED STATUS  
PORTS NAMES  
c54c7ee28158 localhost/python36:1.0 /bin/bash -c slee... 5 seconds ago Created  
python36
```

Now that you verified that the container is created correctly, you decide to start the container, so you run the `podman start` command. You can use the name or the container ID to start the container. The output of this command is the name of the container.

```
[user@host ~]$ podman start python36  
python36  
[user@host ~]$ podman ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS  
PORTS NAMES  
c54c7ee28158 localhost/python36:1.0 /bin/bash -c slee... 6 minutes ago Up 3  
seconds ago python36
```

## Run a Container from a Remote Repository

You can use the `podman run` command to create and run the container in one step. The `podman run` command runs a process inside a container, and this process starts the new container.

You use the `podman run` command `-d` option to run a container in *detached mode*, which runs the container in the background instead of in the foreground of the session. In the example of the `python36` container, you do not need to provide a command for the container to run, because the `sleep infinity` command was already provided in the container file that created the image for that container.

To create the `python38` container, you decide to use the `podman run` command and to refer to the `registry.access.redhat.com/ubi8/python-38` image. You also decide to use the `sleep infinity` command to prevent the container from exiting.

```
[user@host ~]$ podman run -d --name python38 \
registry.access.redhat.com/ubi8/python-38 \
sleep infinity
a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462
[user@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
c54c7ee28158 localhost/python36:1.0 /bin/bash -c
slee... 37 minutes ago Up 30 minutes ago python36
a60f71a1dc1b registry.access.redhat.com/ubi8/python-38:latest sleep infinity
32 seconds ago Up 33 seconds ago python38
```



### Important

If you run a container by using the fully qualified image name, but the image is not yet stored locally, then the `podman run` command first pulls the image from the registry and then runs.

## Environment Isolation in Containers

Containers isolate the environment of an application. Each container has its own file system, networking, and processes. You can notice the isolation feature when you look at the output of the `ps` command and compare it between the host machine and a running container.

You first run the `ps -ax` command on the local machine and the command returns an expected result with many processes.

```
[root@host ~]# ps -ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:01 /usr/lib/systemd/systemd --switched-root --system --
deseriali
    2 ?        S      0:00 [kthreadd]
    3 ?        I<    0:00 [rcu_gp]
    4 ?        I<    0:00 [rcu_par_gp]
...output omitted...
```

The `podman exec` command executes a command inside a running container. The command takes the name or ID of the container as the first argument and the following arguments as commands to run inside the container. You use the `podman exec` command to view the running processes in the `python36` container. The output from the `ps aux` command looks different, because it is running different processes from the local machine.

```
[student@host ~]$ podman exec python38 ps -ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 /usr/bin/coreutils --coreutils-prog-shebang=sleep /
usr/bin/sleep infinity
    7 ?        R      0:00 ps -ax
```

## Chapter 16 | Run Containers

You can use the `sh -c` command to encapsulate the command to execute in the container. In the following example, the `ps -ax > /tmp/process-data.log` command is interpreted as the command to be executed in the container. If you do not encapsulate the command, then Podman might interpret the greater-than character (`>`) as part of the podman command instead of as an argument to the `podman exec` option.

```
[student@host ~]$ podman exec python38 sh -c 'ps -ax > /tmp/process-data.log'
PID TTY      STAT   TIME COMMAND
 1 ?        Ss      0:00 /usr/bin/coreutils --coreutils-prog-shebang=sleep /
/usr/bin/sleep infinity
 7 ?        R      0:00 ps -ax
```

You decide to compare the installed python version on the host system with the installed python version on the containers.

```
[user@host ~]$ python3 --version
Python 3.9.10
[user@host ~]$ podman exec python36 python3 --version
Python 3.6.8
[user@host ~]$ podman exec python38 python3 --version
Python 3.8.8
```

## File-system Isolation in Containers

Developers can use the file-system isolation feature to write and test applications for different versions of programming languages without the need to use multiple physical or virtual machines.

You create a simple bash script that displays `hello world` on the terminal in the `/tmp` directory.

```
[user@host ~]$ echo "echo 'hello world'" > /tmp/hello.sh
```

The `/tmp/hello.sh` file exists on the host machine, and does not exist on the file system inside the containers. If you try to use the `podman exec` to execute the script, then it gives an error, because the `/tmp/hello.sh` script does not exist in the container.

```
[user@host ~]$ stat /tmp/hello.sh
  File: /tmp/hello.sh
  Size: 19          Blocks: 8          IO Block: 4096   regular file
Device: fc04h/64516d  Inode: 17655599      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ user)    Gid: ( 1000/ user)
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2022-04-19 21:47:40.101601412 -0400
Modify: 2022-04-19 21:47:36.497558132 -0400
Change: 2022-04-19 21:47:36.497558132 -0400
 Birth: 2022-04-19 21:45:24.785976758 -0400

[user@host ~]$ podman exec python38 stat /tmp/hello.sh
stat: cannot statx '/tmp/hello.sh': No such file or directory
```

The `podman cp` command copies files and folders between host and container file systems. You can copy the `/tmp/hello.sh` file to the `python38` container with the `podman cp` command.

```
[user@host ~]$ podman cp /tmp/hello.sh python38:/tmp/hello.sh

[user@host ~]$ podman exec python38 stat /tmp/hello.sh
  File: /tmp/hello.sh
  Size: 19          Blocks: 8          IO Block: 4096   regular file
Device: 3bh/59d Inode: 12280058      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/ default)  Gid: (     0/    root)
Access: 2022-04-20 01:47:36.000000000 +0000
Modify: 2022-04-20 01:47:36.000000000 +0000
Change: 2022-04-20 02:02:04.732982187 +0000
 Birth: 2022-04-20 02:02:04.732982187 +0000
```

After the script is copied to the container file system, it can be executed from within the container.

```
[user@host ~]$ podman exec python38 bash /tmp/hello.sh
hello world
```

## Remove Containers and Images

You can remove containers and images by using the `podman rm` and `podman rmi` commands, respectively. Before you remove a container image, any existing running containers from that image must be removed.

You decide to remove the `python38` container and its related image. If you attempt to remove the `registry.access.redhat.com/ubi8/python-38` image while the `python38` container exists, then it gives an error.

```
[user@host ~]$ podman rmi registry.access.redhat.com/ubi8/python-38
Error: Image used by
a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462: image is in use
by a container
```

You must stop the container before you can remove it. To stop a container, use the `podman stop` command.

```
[user@host ~]$ podman stop python38
```

After you stop the container, use the `podman rm` command to remove the container.

```
[user@host ~]$ podman rm python38
a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462
```

When the container no longer exists, the `registry.access.redhat.com/ubi8/python-38` can be removed with the `podman rmi` command.

```
[user@host ~]$ podman rmi registry.access.redhat.com/ubi8/python-38
Untagged: registry.access.redhat.com/ubi8/python-38:latest
Deleted: a33d92f90990c9b1bad9aa98fe017e48f30c711b49527dcc797135352ea57d12
```



## References

`podman(1)`, `podman-build(1)`, `podman-cp(1)`, `podman-exec(1)`, `podman-images(1)`, `podman-inspect(1)`, `podman-ps(1)`, `podman-pull(1)`, `podman-rm(1)`, `podman-rmi(1)`, `podman-run(1)`, `podman-search(1)`, and `podman-stop(1)` man pages.

For more information, refer to the Starting with containers chapter in the Building, running, and managing Linux containers on Red Hat Enterprise Linux 9 Guide at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/building\\_running\\_and\\_managing\\_containers/index#starting-with-containers\\_building-running-and-managing-containers](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/index#starting-with-containers_building-running-and-managing-containers)

## ► Guided Exercise

# Deploy Containers

In this exercise, you use container management tools to build an image, run a container, and query the running container environment.

### Outcomes

- Configure a container image registry and create a container from an existing image.
- Create a container by using a container file.
- Copy a script from a host machine into containers and run the script.
- Delete containers and images.

### Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start containers-deploy
```

### Instructions

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

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

- 2. Install the `container-tools` meta-package.

```
[student@servera ~]$ sudo dnf install container-tools  
[sudo] password for student: student  
...output omitted...  
Is this ok [y/N]: y  
...output omitted...  
Complete!
```

- 3. Configure the `registry.lab.example.com` classroom registry in your home directory.  
Log in to the container registry with `admin` as the user and `redhat321` as the password.

- 3.1. Create the `/home/student/.config/containers` directory.

```
[student@servera ~]$ mkdir -p /home/student/.config/containers
```

- 3.2. Create the /home/student/.config/containers/registries.conf file with the following contents:

```
unqualified-search-registries = ['registry.lab.example.com']

[[registry]]
location = "registry.lab.example.com"
insecure = true
blocked = false
```

- 3.3. Verify that the classroom registry is added.

```
[student@servera ~]$ podman info
...output omitted...
registries:
  registry.lab.example.com:
    Blocked: false
    Insecure: true
    Location: registry.lab.example.com
    MirrorByDigestOnly: false
    Mirrors: null
    Prefix: registry.lab.example.com
  search:
    - registry.lab.example.com
...output omitted...
```

- 3.4. Log in to the classroom registry.

```
[student@servera ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

- 4. Run the python38 container in detached mode from an image with the python 3.8 package and based on the ubi8 image. The image is hosted on a remote registry.

- 4.1. Search for a python-38 container in the registry.lab.example.com registry.

```
[student@servera ~]$ podman search registry.lab.example.com/
NAME                                     DESCRIPTION
...output omitted...
registry.lab.example.com/ubi8/python-38
registry.lab.example.com/ubi8/httpd-24
registry.lab.example.com/rhel8/php-74
```

- 4.2. Inspect the image.

```
[student@servera ~]$ skopeo inspect \
docker://registry.lab.example.com/ubi8/python-38
...output omitted...
  "description": "Python 3.8 available as container is a base platform for
building and running various Python 3.8 applications and frameworks.
...output omitted...
```

4.3. Pull the python-38 container image.

```
[student@servera ~]$ podman pull registry.lab.example.com/ubi8/python-38
Trying to pull registry.lab.example.com/ubi8/python-38:latest...
...output omitted...
671cc3cb42984e338733ebb5a9a68e69e267cb7f9cb802283d3bc066f6321617
```

4.4. Verify that the container is downloaded to the local image repository.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
registry.lab.example.com/ubi8/python-38	latest	671cc3cb4298	5 days ago	901 MB

4.5. Start the python38 container.

```
[student@servera ~]$ podman run -d --name python38 \
registry.lab.example.com/ubi8/python-38 sleep infinity
004756b52d3d3326545f5075594cffa858afd474b903288723a3aa299e72b1af
```

4.6. Verify that the container was created.

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS NAMES
004756b52d3d	registry.lab.example.com/ubi8/python-38:latest	sleep infinity
About a minute ago	Up About a minute ago	python38

- 5. Build a container image called python39:1.0 from a container file, and use the image to create a container called python39.

5.1. Examine the container file in the /home/student/python39 directory.

```
[student@servera ~]$ cat /home/student/python39/Containerfile
FROM registry.lab.example.com/ubi9-beta/ubi:latest
RUN echo -e '[rhel-9.0-for-x86_64-baseos-rpms]\nbaseurl = http://
content.example.com/rhel9.0/x86_64/dvd/BaseOS\nenabled = true\npgpcheck =
false\nname = Red Hat Enterprise Linux 9.0 BaseOS (dvd)\n[rhel-9.0-for-x86_64-
appstream-rpms]\nbaseurl = http://content.example.com/rhel9.0/x86_64/dvd/AppStream
\nenabled = true\npgpcheck = false\nname = Red Hat Enterprise Linux 9.0 Appstream
(dvd)'/>/etc/yum.repos.d/rhel_dvd.repo
RUN yum install --disablerepo=* --enablerepo=rhel-9.0-for-x86_64-baseos-rpms --
enablerepo=rhel-9.0-for-x86_64-appstream-rpms -y python3
```

## 5.2. Create the container image from the container file.

```
[student@servera ~]$ podman build -t python39:1.0 /home/student/python39/.
STEP 1/4: FROM registry.lab.example.com/ubi9-beta/ubi:latest
...output omitted...
STEP 2/4: RUN echo -e '[rhel-9.0-for-x86_64-baseos-rpms] ...
...output omitted...
STEP 3/4: RUN yum install --disablerepo=* --enablerepo=rhel-9.0-for-x86_64-baseos-
rpms --enablerepo=rhel-9.0-for-x86_64-appstream-rpms -y python3
...output omitted...
STEP 4/4: CMD ["/bin/bash", "-c", "sleep infinity"]
...output omitted...
Successfully tagged localhost/python39:1.0
80e68c195925beafe3b2ad7a54fe1e5673993db847276bc62d5f9d109e9eb499
```

## 5.3. Verify that the container image exists in the local image repository.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/python39	1.0	80e68c195925	3 minutes ago	266 MB
registry.lab.example.com/ubi8/python-38	latest	671cc3cb4298	5 days ago	901 MB
registry.lab.example.com/ubi9-beta/ubi	latest	fca12da1dc30	4 months ago	235 MB

## 5.4. Inspect the python39 container.

```
[student@servera ~]$ podman inspect localhost/python39:1.0
...output omitted...
  "comment": "FROM registry.lab.example.com/ubi9-beta/ubi:latest"
...output omitted...
  "created_by": "/bin/sh -c yum install --disablerepo=*
--enablerepo=rhel-9.0-for-x86_64-baseos-rpms --enablerepo=rhel-9.0-for-x86_64-
appstream-rpms -y python3"
...output omitted...
  "created_by": "/bin/sh -c #(nop) CMD [\"/bin/bash\", \"-c\", \"sleep
infinity\"]",
...output omitted...
```

## 5.5. Create the python39 container.

```
[student@servera ~]$ podman create --name python39 localhost/python39:1.0
3db4eabe9043224a7bdf195ab5fd810bf95db98dc29193392cef7b94489e1aae
```

## 5.6. Start the python39 container.

```
[student@servera ~]$ podman start python39
python39
```

## 5.7. Verify that the container is running.

```
[student@servera ~]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
004756b52d3d registry.lab.example.com/ubi8/python-38:latest sleep infinity
33 minutes ago Up 33 minutes ago python38
3db4eabe9043 localhost/python39:1.0 /bin/bash -c
slee... About a minute ago Up 42 seconds ago python39
```

- 6. Copy the /home/student/script.py script into the /tmp directory of the running containers and run the script on each container.

- 6.1. Copy the /home/student/script.py python script into the /tmp directory in both containers.

```
[student@servera ~]$ podman cp /home/student/script.py python39:/tmp/script.py
[student@servera ~]$ podman cp /home/student/script.py python38:/tmp/script.py
```

- 6.2. Run the Python script in both containers, and then run the Python script on the host.

```
[student@servera ~]$ podman exec -it python39 python3 /tmp/script.py
This script was not run on the correct version of Python
Expected version of Python is 3.8
Current version of python is 3.9
[student@servera ~]$ podman exec -it python38 python3 /tmp/script.py
This script was correctly run on Python 3.8
[student@servera ~]$ python3 /home/student/script.py
This script was not run on the correct version of Python
Expected version of Python is 3.8
Current version of python is 3.9
```

- 7. Delete containers and images. Return to workstation.

- 7.1. Stop both containers.

```
[student@servera ~]$ podman stop python39 python38
...output omitted...
python38
python39
```

- 7.2. Remove both containers.

```
[student@servera ~]$ podman rm python39 python38
3db4eabe9043224a7bd195ab5fd810bf95db98dc29193392cef7b94489e1aae
004756b52d3d3326545f5075594cffa858afd474b903288723a3aa299e72b1af
```

- 7.3. Remove both container images.

```
[student@servera ~]$ podman rmi localhost/python39:1.0 \
registry.lab.example.com/ubi8/python-38:latest \
registry.lab.example.com/ubi9-beta/ubi
Untagged: localhost/python39:1.0
Untagged: registry.lab.example.com/ubi8/python-38:latest
Deleted: 80e68c195925beafe3b2ad7a54fe1e5673993db847276bc62d5f9d109e9eb499
Deleted: 219e43f6ff96fd11ea64f67cd6411c354dacbc5cbe296ff1fdbf5b717f01d89a
Deleted: 671cc3cb42984e338733ebb5a9a68e69e267cb7f9cb802283d3bc066f6321617
```

7.4. Return to workstation.

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

## 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 containers-deploy
```

This concludes the section.

# Manage Container Storage and Network Resources

## Objectives

Provide persistent storage for container data by sharing storage from the container host, and configure a container network.

## Manage Container Resources

You can use containers to run a simple process and exit. You can also configure a container to run a service continuously, such as a database server. In this scenario, you might consider adding more resources to the container, such as persistent storage or DNS resolution for other containers.

You can use different strategies to configure persistent storage for containers. In an enterprise container platform, such as Red Hat OpenShift, you can use sophisticated storage solutions to provide storage to your containers without knowing the underlying infrastructure.

For small deployments where you use only a single container host, and without a need to scale, you can create persistent storage from the container host by creating a directory to mount on the running container.

When a container, such as a web server or database server, serves content for clients outside the container host, you must set up a communication channel for those clients to access the content of the container. You can configure *port mapping* to enable communication to a container. With port mapping, the requests that are destined for a port on the container host are forwarded to a port inside the container.

To cover the topics in this lecture, imagine the following scenario.

As a system administrator, you are tasked to create the db01 containerized database, based on MariaDB, to use the local machine to allow port 3306 traffic with the appropriate firewall configuration. The db01 container must use persistent storage with the appropriate SELinux context. Add the appropriate network configuration so that the client01 container can communicate with the db01 container with DNS.

## Environment Variables for Containers

Some container images allow passing environment variables to customize the container at creation time. You can use environment variables to set parameters to the container to tailor for your environment without the need to create your own custom image. Usually, you would not modify the container image, because it would add layers to the image, which might be harder to maintain.

You use the `podman run -d registry.lab.example.com/rhel8/mariadb-105` command to run a containerized database, but you notice that the container fails to start.

```
[user@host ~]$ podman run -d registry.lab.example.com/rhel8/mariadb-105 \
--name db01
20751a03897f14764fb0e7c58c74564258595026124179de4456d26c49c435ad
[user@host ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
20751a03897f registry.lab.example.com/rhel8/mariadb-105:latest run-mysqld
29 seconds ago Exited (1) 29 seconds ago db01
```

You use the `podman container logs` command to investigate the reason of the container status.

```
[user@host ~]$ podman container logs db01
...output omitted...
You must either specify the following environment variables:
  MYSQL_USER (regex: '^[_a-zA-Z0-9_-]+$')
  MYSQL_PASSWORD (regex: '^[_a-zA-Z0-9_-!@#$%^&*()-=<>,_.?;:_]+$')
  MYSQL_DATABASE (regex: '^[_a-zA-Z0-9_-]+$')
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[_a-zA-Z0-9_-!@#$%^&*()-=<>,_.?;:_]+$')
Or both.
...output omitted...
```

From the preceding output, you determine that the container did not continue to run because the required environment variables were not passed to the container. So you inspect the `mariadb-105` container image to find more information about the environment variables to customize the container.

```
[user@host ~]$ skopeo inspect docker://registry.lab.example.com/rhel8/mariadb-105
...output omitted...
{
  "name": "rhel8/mariadb-105",
  "release": "40.1647451927",
  "summary": "MariaDB 10.5 SQL database server",
  "url": "https://access.redhat.com/containers/#/registry.access.redhat.com/rhel8/mariadb-105/images/1-40.1647451927",
  "usage": "podman run -d -e MYSQL_USER=user -e MYSQL_PASSWORD=pass -e MYSQL_DATABASE=db -p 3306:3306 rhel8/mariadb-105",
  "vcs-ref": "c04193b96a119e176ada62d779bd44a0e0edf7a6",
  "vcs-type": "git",
  "vendor": "Red Hat, Inc.",
  ...output omitted...
```

The `usage` label from the output provides an example of how to run the image. The `url` label points to a web page in the Red Hat Container Catalog that documents environment variables and other information about how to use the container image.

The documentation for this image shows that the container uses the 3306 port for the database service. The documentation also shows that the following environment variables are available to configure the database service:

## Environment Variables for the mariadb Image

Variable	Description
MYSQL_USER	Username for the MySQL account to be created
MYSQL_PASSWORD	Password for the user account
MYSQL_DATABASE	Database name
MYSQL_ROOT_PASSWORD	Password for the root user (optional)

After examining the available environment variables for the image, you use the `podman run` command `-e` option to pass environment variables to the container, and use the `podman ps` command to verify that it is running.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
registry.lab.example.com/rhel8/mariadb-105
[user@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
4b8f01be7fd6 registry.lab.example.com/rhel8/mariadb-105:latest run-mysqld 6
seconds ago Up 6 seconds ago db01
```

## Container Persistent Storage

By default, the storage that a container uses is ephemeral. The ephemeral nature of container storage means that its contents are lost after you remove the container. You must consider file-system level permissions when you mount a persistent volume in a container.

In the MariaDB image, the `mysql` user must own the `/var/lib/mysql` directory, the same as if MariaDB was running on the host machine. The directory that you intend to mount into the container must have `mysql` as the user and group owner (or the UID/GID of the `mysql` user, if MariaDB is not installed on the host machine). If you run a container as the `root` user, then the UIDs and GIDs on your host machine match the UIDs and GIDs inside the container.

The UID and GID matching configuration does not occur the same way in a rootless container. In a rootless container, the user has root access from within the container, because Podman launches a container inside the user namespace.

You can use the `podman unshare` command to run a command inside the user namespace. To obtain the UID mapping for your user namespace, use the `podman unshare cat` command.

```
[user@host ~]$ podman unshare cat /proc/self/uid_map
0      1000      1
1      100000    65536
[user@host ~]$ podman unshare cat /proc/self/gid_map
0      1000      1
1      100000    65536
```

The preceding output shows that in the container, the root user (UID and GID of 0) maps to your user (UID and GID of 1000) on the host machine. In the container, the UID and GID of 1 maps to the UID and GID of 100000 on the host machine. Every UID and GID after 1 increments by 1. For example, the UID and GID of 30 inside a container maps to the UID and GID of 100029 on the host machine.

You use the `podman exec` command to view the `mysql` user UID and GID inside the container that is running with ephemeral storage.

```
[user@host ~]$ podman exec -it db01 grep mysql /etc/passwd
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
```

You decide to mount the `/home/user/db_data` directory into the `db01` container to provide persistent storage on the `/var/lib/mysql` directory of the container. You then create the `/home/user/db_data` directory and use the `podman unshare` command to set the user namespace UID and GID of 27 as the owner of the directory.

```
[user@host ~]$ mkdir /home/user/db_data
[user@host ~]$ podman unshare chown 27:27 /home/user/db_data
```

The UID and GID of 27 in the container maps to the UID and GID of 100026 on the host machine. You can verify the mapping by viewing the ownership of the `/home/user/db_data` directory with the `ls` command.

```
[student@workstation ~]$ ls -l /home/user/
total 0
drwxrwxr-x. 3 100026 100026 18 May 5 14:37 db_data
...output omitted...
```

Now that the correct file-system level permissions are set, you use the `podman run` command `-v` option to mount the directory.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql \
registry.lab.example.com/rhel8/mariadb-105
```

You notice that the `db01` container is not running.

CONTAINER ID	IMAGE	COMMAND	
CREATED	STATUS	PORTS	NAMES
dfdc20cf9a7e	registry.lab.example.com/rhel8/mariadb-105:latest	run-mysqld	db01
29 seconds ago	Exited (1)	29 seconds ago	

The `podman container logs` command shows a permission error for the `/var/lib/mysql` data directory.

```
[user@host ~]$ podman container logs db01
...output omitted...
--> 16:41:25      Initializing database ...
--> 16:41:25      Running mysql_install_db ...
mkdir: cannot create directory '/var/lib/mysql/data': Permission denied
Fatal error Can't create database directory '/var/lib/mysql/data'
```

This error happens because of the incorrect SELinux context that is set on the /home/user/db\_data directory on the host machine.

## SELinux Contexts for Container Storage

You must set the `container_file_t` SELinux context type before you can mount the directory as persistent storage to a container. If the directory does not have the `container_file_t` SELinux context, then the container cannot access the directory. You can append the `Z` option to the argument of the `podman run` command `-v` option to automatically set the SELinux context on the directory.

So you use the `podman run -v /home/user/dbfiles:/var/lib/mysql:Z` command to set the SELinux context for the `/home/user/dbfiles` directory when you mount it as persistent storage for the `/var/lib/mysql` directory.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
registry.lab.example.com/rhel8/mariadb-105
```

You then verify that the correct SELinux context is set on the `/home/user/dbfiles` directory with the `ls` command `-Z` option.

```
[user@host ~]$ ls -Z /home/user/
system_u:object_r:container_file_t:s0:c81,c1009 dbfiles
...output omitted...
```

## Assign a Port Mapping to Containers

To provide network access to containers, clients must connect to ports on the container host that pass the network traffic through to ports in the container. When you map a network port on the container host to a port in the container, network traffic that is sent to the host network port is received by the container.

For example, you can map the 13306 port on the container host to the 3306 port on the container for communication with the MariaDB container. Therefore, traffic that is sent to the container host port 13306 would be received by MariaDB that is running in the container.

You use the `podman run` command `-p` option to set a port mapping from the 13306 port from the container host to the 3306 port on the db01 container.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
-p 13306:3306 \
registry.lab.example.com/rhel8/mariadb-105
```

Use the `podman port` command `-a` option to show all container port mappings in use. You can also use the `podman port db01` command to show the mapped ports for the `db01` container.

```
[user@host ~]$ podman port -a
1c22fd905120 3306/tcp -> 0.0.0.0:13306
[user@host ~]$ podman port db01
3306/tcp -> 0.0.0.0:13306
```

You use the `firewall-cmd` command to allow port 13306 traffic into the container host machine so that it can be redirected to the container.

```
[root@host ~]# firewall-cmd --add-port=13306/tcp --permanent
[root@host ~]# firewall-cmd --reload
```



### Important

A rootless container cannot open a privileged port (ports below 1024) on the container. That is, the `podman run -p 80:8080` command does not normally work for a running rootless container. To map a port on the container host below 1024 to a container port, you must run Podman as root or make other adjustments to the system.

You can map a port above 1024 on the container host to a privileged port on the container, even if you are running a rootless container. The `8080:80` mapping works if the container provides service listening on port 80.

## DNS Configuration in a Container

Podman v4.0 supports two network back ends for containers, Netavark and CNI. Starting with RHEL 9, systems use Netavark by default. To verify which network back end is used, run the following `podman info` command.

```
[user@host ~]$ podman info --format {{.Host.NetworkBackend}}
netavark
```

**Note**

The `container-tools` meta-package includes the `netavark` and `aardvark-dns` packages. If Podman was installed as a stand-alone package, or if the `container-tools` meta-package was installed later, then the result of the previous command might be `cni`. To change the network back end, set the following configuration in the `/usr/share/containers/containers.conf` file:

```
[network]
...output omitted...
network_backend = "netavark"
```

Existing containers on the host that use the default Podman network cannot resolve each other's hostnames, because DNS is not enabled on the default network.

Use the `podman network create` command to create a DNS-enabled network. You use the `podman network create` command to create the network called `db_net`, and specify the subnet as `10.87.0.0/16` and the gateway as `10.87.0.1`.

```
[user@host ~]$ podman network create --gateway 10.87.0.1 \
--subnet 10.87.0.0/16 db_net
db_net
```

If you do not specify the `--gateway` or `--subnet` options, then they are created with the default values.

The `podman network inspect` command displays information about a specific network. You use the `podman network inspect` command to verify that the gateway and subnet were correctly set and that the new `db_net` network is DNS-enabled.

```
[user@host ~]$ podman network inspect db_net
[
  {
    "name": "db_net",
    ...output omitted...
    "subnets": [
      {
        "subnet": "10.87.0.0/16",
        "gateway": "10.87.0.1"
      }
    ],
    ...output omitted...
    "dns_enabled": true,
    ...output omitted...
  ]
]
```

You can add the DNS-enabled `db_net` network to a new container with the `podman run` command `--network` option. You use the `podman run` command `--network` option to create the `db01` and `client01` containers that are connected to the `db_net` network.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
```

```
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
-p 13306:3306 \
--network db_net \
registry.lab.example.com/rhel8/mariadb-105
[user@host ~]$ podman run -d --name client01 \
--network db_net \
registry.lab.example.com/ubi8/ubi:latest \
sleep infinity
```

Because containers are designed to have only the minimum required packages, the containers might not have the required utilities to test communication such as the ping and ip commands. You can install these utilities in the container by using the podman exec command.

```
[user@host ~]$ podman exec -it db01 dnf install -y iputils iproute
...output omitted...
[user@host ~]$ podman exec -it client01 dnf install -y iputils iproute
...output omitted...
```

The containers can now ping each other by container name. You test the DNS resolution with the podman exec command. The names resolve to IPs within the subnet that was manually set for the db\_net network.

```
[user@host ~]$ podman exec -it db01 ping -c3 client01
PING client01.dns.podman (10.87.0.4) 56(84) bytes of data.
64 bytes from 10.87.0.4 (10.87.0.4): icmp_seq=1 ttl=64 time=0.049 ms
...output omitted...
--- client01.dns.podman ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.049/0.060/0.072/0.013 ms

[user@host ~]$ podman exec -it client01 ping -c3 db01
PING db01.dns.podman (10.87.0.3) 56(84) bytes of data.
64 bytes from 10.87.0.3 (10.87.0.3): icmp_seq=1 ttl=64 time=0.021 ms
...output omitted...
--- db01.dns.podman ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.021/0.040/0.050/0.013 ms
```

You verify that the IP addresses in each container match the DNS resolution with the podman exec command.

```
[user@host ~]$ podman exec -it db01 ip a | grep 10.8
inet 10.87.0.3/16 brd 10.87.255.255 scope global eth0
    inet 10.87.0.4/16 brd 10.87.255.255 scope global eth0
[user@host ~]$ podman exec -it client01 ip a | grep 10.8
    inet 10.87.0.3/16 brd 10.87.255.255 scope global eth0
    inet 10.87.0.4/16 brd 10.87.255.255 scope global eth0
```

## Multiple Networks to a Single Container

Multiple networks can be connected to a container at the same time to help to separate different types of traffic.

You use the `podman network create` command to create the backend network.

```
[user@host ~]$ podman network create backend
```

You then use the `podman network ls` command to view all the Podman networks.

```
[user@host ~]$ podman network ls
NETWORK ID      NAME      DRIVER
a7fea510a6d1   backend   bridge
fe680efc5276   db01     bridge
2f259bab93aa   podman   bridge
```

The subnet and gateway were not specified with the `podman network create` command `--gateway` and `--subnet` options.

You use the `podman network inspect` command to obtain the IP information of the backend network.

```
[user@host ~]$ podman network inspect backend
[
  {
    "name": "backend",
    ...output omitted...
    "subnets": [
      {
        "subnet": "10.89.1.0/24",
        "gateway": "10.89.1.1"
      }
    ]
  }
]
```

You can use the `podman network connect` command to connect additional networks to a container when it is running. You use the `podman network connect` command to connect the backend network to the `db01` and `client01` containers.

```
[user@host ~]$ podman network connect backend db01
[user@host ~]$ podman network connect backend client01
```



### Important

If a network is not specified with the `podman run` command, then the container connects to the default network. The default network uses the `slirp4netns` network mode, and the networks that you create with the `podman network create` command use the `bridge` network mode. If you try to connect a bridge network to a container by using the `slirp4netns` network mode, then the command fails:

```
Error: "slirp4netns" is not supported: invalid network mode
```

You use the podman inspect command to verify that both networks are connected to each container and to display the IP information.

```
[user@host ~]$ podman inspect db01
...output omitted...
    "backend": {
        "EndpointID": "",
        "Gateway": "10.89.1.1",
        "IPAddress": "10.89.1.4",
    },
    "db_net": {
        "EndpointID": "",
        "Gateway": "10.87.0.1",
        "IPAddress": "10.87.0.3",
    }
...output omitted...
[user@host ~]$ podman inspect client01
...output omitted...
    "backend": {
        "EndpointID": "",
        "Gateway": "10.89.1.1",
        "IPAddress": "10.89.1.5",
    },
    "db_net": {
        "EndpointID": "",
        "Gateway": "10.87.0.1",
        "IPAddress": "10.87.0.4",
    }
...output omitted...
```

The client01 container can now communicate with the db01 container on both networks. You use the podman exec command to ping both networks on the db01 container from the client01 container.

```
[user@host ~]$ podman exec -it client01 ping -c3 10.89.1.4 | grep 'packet loss'
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
[user@host ~]$ podman exec -it client01 ping -c3 10.87.0.3 | grep 'packet loss'
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
```



## References

podman(1), podman-exec(1), podman-info(1), podman-network(1), podman-network-create(1), podman-network-inspect(1), podman-network-ls(1), podman-port(1), podman-run(1), and podman-unshare(1) man pages

For more information, refer to the Working with containers chapter in the Building, running, and managing Linux containers on Red Hat Enterprise Linux 9 Guide at [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/building\\_running\\_and\\_managing\\_containers/assembly\\_working-with-containers\\_building-running-and-managing-containers](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/assembly_working-with-containers_building-running-and-managing-containers)

## ► Guided Exercise

# Manage Container Storage and Network Resources

In this exercise, you pass environment variables to a container during creation, mount persistent storage to a container, create and connect multiple container networks, and expose container ports from the host machine.

## Outcomes

- Create container networks and connect them to containers.
- Troubleshoot failed containers.
- Pass environment variables to containers during creation.
- Create and mount persistent storage to containers.
- Map host ports to ports inside containers.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start containers-resources
```

## Instructions

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

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

- 2. Create the `frontend` container network. Create the `db_client` and `db_01` containers and connect them to the `frontend` network.

- 2.1. Use the `podman network create` command `--subnet` and `--gateway` options to create the `frontend` network with the `10.89.1.0/24` subnet and the `10.89.1.1` gateway.

```
[student@servera ~]$ podman network create --subnet 10.89.1.0/24 \
--gateway 10.89.1.1 frontend
frontend
```

- 2.2. Log in to the `registry.lab.example.com` registry.

```
[student@servera ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

- 2.3. Create the db\_client container that is connected to the frontend network. Mount the /etc/yum.repos.d DNF repositories directory inside the container at /etc/yum.repos.d to allow package installation.

```
[student@servera ~]$ podman run -d --name db_client \
--network frontend \
-v /etc/yum.repos.d:/etc/yum.repos.d \
registry.lab.example.com/ubi9-beta/ubi \
sleep infinity
e20dfed7e392abe4b7bea3c25e9cb17ef95d16af9cedd50d68f997a663ba6c15
```

- 2.4. Create the db\_01 container that is connected to the frontend network.

```
[student@servera ~]$ podman run -d --name db_01 --network frontend \
registry.lab.example.com/rhel8/mariadb-105
3e767ae6eea4578152a216beb5ae98c8ef03a2d66098debe2736b8b458bab405
```

- 2.5. View all containers.

```
[student@servera ~]$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND
CREATED      STATUS          PORTS     NAMES
e20dfed7e392  registry.lab.example.com/ubi8/ubi:latest    sleep infinity
56 seconds ago Up 56 seconds ago           db_client
3e767ae6eea4  registry.lab.example.com/rhel8/mariadb-105:latest run-mysqld 1
second ago   Exited (1) 1 second ago           db_01
```

- 3. Troubleshoot the db\_01 container and determine why it is not running. Re-create the db\_01 container by using the required environment variables.

- 3.1. View the container logs and determine why the container exited.

```
[student@servera ~]$ podman container logs db_01
...output omitted...
You must either specify the following environment variables:
  MYSQL_USER (regex: '^[_a-zA-Z0-9_-]+$')
  MYSQL_PASSWORD (regex: '^[_a-zA-Z0-9_-!@#$%^&*()-=<>,_.?;:_]+$')
  MYSQL_DATABASE (regex: '^[_a-zA-Z0-9_-]+$')
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[_a-zA-Z0-9_-!@#$%^&*()-=<>,_.?;:_]+$')
Or both.
...output omitted...
```

- 3.2. Remove the db\_01 container and create it again with environment variables. Provide the required environment variables.

```
[student@servera ~]$ podman rm db_01
3e767ae6eea4578152a216beb5ae98c8ef03a2d66098debe2736b8b458bab405
[student@servera ~]$ podman run -d --name db_01 \
--network frontend \
-e MYSQL_USER=dev1 \
-e MYSQL_PASSWORD=devpass \
-e MYSQL_DATABASE=devdb \
-e MYSQL_ROOT_PASSWORD=redhat \
registry.lab.example.com/rhel8/mariadb-105
948c4cd767b561432056e77adb261ab4024c1b66a22af17861aba0f16c66273b
```

3.3. View the current running containers.

COLUMN	CONTAINER ID	IMAGE	CREATED	STATUS	PORTS	NAMES
sleep infinity	e20dfed7e392	registry.lab.example.com/ubi8/ubi:latest	56 seconds ago	Up 56 seconds ago		db_client
run-mysqld	948c4cd767b5	registry.lab.example.com/rhel8/mariadb-105:latest	11 seconds ago	Up 12 seconds ago		db_01

- ▶ 4. Create persistent storage for the containerized MariaDB service, and map the local machine 13306 port to the 3306 port in the container. Allow traffic to the 13306 port on the servera machine.

4.1. Create the /home/student/database directory on the servera machine.

```
[student@servera ~]$ mkdir /home/student/databases
```

4.2. Obtain the mysql UID and GID from the db\_01 container, and then remove the db01 container.

```
[student@servera ~]$ podman exec -it db_01 grep mysql /etc/passwd
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
[student@servera ~]$ podman stop db_01
db_01
[student@servera ~]$ podman rm db_01
948c4cd767b561432056e77adb261ab4024c1b66a22af17861aba0f16c66273b
```

4.3. Run the chown command inside the container namespace, and set the user and group owner to 27 on the /home/student/database directory.

```
[student@servera ~]$ podman unshare chown 27:27 /home/student/databases/
[student@servera ~]$ ls -l /home/student/
total 0
drwxr-xr-x. 2 100026 100026 6 May 9 17:40 databases
```

4.4. Create the db\_01 container, and mount the /home/student/databases directory from the servera machine to the /var/lib/mysql directory inside the db\_01 container. Use the Z option to apply the required SELinux context.

```
[student@servera ~]$ podman run -d --name db_01 \
--network frontend \
-e MYSQL_USER=dev1 \
-e MYSQL_PASSWORD=devpass \
-e MYSQL_DATABASE=devdb \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/student/databases:/var/lib/mysql:z \
-p 13306:3306 \
registry.lab.example.com/rhel8/mariadb-105
```

- 4.5. Install the mariadb package in the db\_client container.

```
[student@servera ~]$ podman exec -it db_client dnf install -y mariadb
...output omitted...
Complete!
```

- 4.6. Create the crucial\_data table in the dev\_db database in the db\_01 container from the db\_client container.

```
[student@servera ~]$ podman exec -it db_client mysql -u dev1 -p -h db_01
Enter password: devpass
...output omitted...
MariaDB [(none)]> USE devdb;
Database changed
MariaDB [devdb]> CREATE TABLE crucial_data(column1 int);
Query OK, 0 rows affected (0.036 sec)

MariaDB [devdb]> SHOW TABLES;
+-----+
| Tables_in_devdb |
+-----+
| crucial_data    |
+-----+
1 row in set (0.001 sec)

MariaDB [devdb]> quit
Bye
```

- 4.7. Allow port 13306 traffic in the firewall on the servera machine.

```
[student@servera ~]$ sudo firewall-cmd --add-port=13306/tcp --permanent
[sudo] password for student: student
success
[student@servera ~]$ sudo firewall-cmd --reload
success
```

- 4.8. Open a second terminal on the Workstation machine and use the MariaDB client to connect to the servera machine on port 13306 to show tables inside the db\_01 container that are stored in the persistent storage.

```
[student@workstation ~]$ mysql -u dev1 -p -h servera --port 13306 \
devdb -e 'SHOW TABLES';
Enter password: devpass
+-----+
| Tables_in_devdb |
+-----+
| crucial_data     |
+-----+
```

- ▶ 5. Create a second container network called `backend`, and connect the `backend` network to the `db_client` and `db_01` containers. Test network connectivity and DNS resolution between the containers.
  - 5.1. Create the `backend` network with the `10.90.0.0/24` subnet and the `10.90.0.1` gateway.

```
[student@servera ~]$ podman network create --subnet 10.90.0.0/24 \
--gateway 10.90.0.1 backend
backend
```

- 5.2. Connect the `backend` container network to the `db_client` and `db_01` containers.

```
[student@servera ~]$ podman network connect backend db_client
[student@servera ~]$ podman network connect backend db_01
```

- 5.3. Obtain the IP addresses of the `db_01` container.

```
[student@servera ~]$ podman inspect db_01
...output omitted...
{
  "Networks": {
    "backend": {
      "EndpointID": "",
      "Gateway": "10.90.0.1",
      "IPAddress": "10.90.0.3",
      ...
    },
    "frontend": {
      "EndpointID": "",
      "Gateway": "10.89.1.1",
      "IPAddress": "10.89.1.6",
      ...
    }
  }
}
```

- 5.4. Install the `iputils` package in the `db_client` container.

```
[student@servera ~]$ podman exec -it db_client dnf install -y iputils
...output omitted...
Complete!
```

- 5.5. Ping the `db_01` container name from the `db_client` container.

```
[student@servera ~]$ podman exec -it db_client ping -c4 db_01
PING db_01.dns.podman (10.90.0.3) 56(84) bytes of data.
...output omitted...
--- db_01.dns.podman ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3048ms
rtt min/avg/max/mdev = 0.043/0.049/0.054/0.004 ms
```

5.6. Exit the servera machine.

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

## 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 containers-resources
```

This concludes the section.

# Manage Containers as System Services

## Objectives

Configure a container as a `systemd` service, and configure a container service to start at boot time.

## Manage Small Container Environments with `systemd` Units

You can run a container to complete a system task or to obtain the output of a series of commands. You also might want to run containers that run a service indefinitely, such as web servers or databases. In a traditional environment, a privileged user typically configures these services to run at system boot, and manages them with the `systemctl` command.

As a regular user, you can create a `systemd` unit to configure your rootless containers. You can use this configuration to manage your container as a regular system service with the `systemctl` command.

Managing containers based on `systemd` units is mainly useful for basic and small deployments that do not need to scale. For more sophisticated scaling and orchestration of many container-based applications and services, you can use an enterprise orchestration platform that is based on Kubernetes, such as Red Hat OpenShift Container Platform.

To discuss the topics in this lecture, imagine the following scenario.

As a system administrator, you are tasked to configure the `webserver1` container that is based on the `http24` container image to start at system boot. You must also mount the `/app-artifacts` directory for the web server content and map the 8080 port from the local machine to the container. Configure the container to start and stop with `systemctl` commands.

## Requirements for `systemd` User Services

As a regular user, you can enable a service with the `systemctl` command. The service starts when you open a session (graphical interface, text console, or SSH) and it stops when you close the last session. This behavior differs from a system service, which starts when the system boots and stops when the system shuts down.

By default, when you create a user account with the `useradd` command, the system uses the next available ID from the regular user ID range. The system also reserves a range of IDs for the user's containers in the `/etc/subuid` file. If you create a user account with the `useradd` command `--system` option, then the system does not reserve a range for the user containers. As a consequence, you cannot start rootless containers with system accounts.

You decide to create a dedicated user account to manage containers. You use the `useradd` command to create the `appdev-adm` user, and use `redhat` as the password.

```
[user@host ~]$ sudo useradd appdev-adm
myapp.service
[user@host ~]$ sudo passwd appdev-adm
Changing password for user appdev-adm.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

You then use the `su` command to switch to the `appdev-adm` user, and you start using the `podman` command.

```
[user@host ~]$ su appdev-adm
Password: redhat
[appdev-adm@host ~]$ podman info
ERRO[0000] XDG_RUNTIME_DIR directory "/run/user/1000" is not owned by the current
user
[appdev-adm@host ~]$
```

Podman is a stateless utility and requires a full login session. Podman must be used within an SSH session and cannot be used in a `sudo` or an `su` shell. So you exit the `su` shell and log in to the machine via SSH.

```
[appdev-adm@host ~]$ exit
[user@host ~]$ exit
[user@example ~]$ ssh appdev-adm@host
[appdev-adm@host ~]$
```

You then configure the container registry and authenticate with your credentials. You run the `http` container with the following command.

```
[appdev-adm@host ~]$ podman run -d --name webserver1 -p 8080:8080 -v \
~/app-artifacts:/var/www:Z registry.access.redhat.com/ubi8/httpd-24
af84e1ec33ea2f0d9787c56fbe7a62a4b9ce8ac03911be9e97f95575b306c297
[appdev-adm@host ~]$ podman ps -a
CONTAINER ID  IMAGE                                     COMMAND
              CREATED        STATUS          PORTS          NAMES
af84e1ec33ea  registry.access.redhat.com/ubi8/httpd-24:latest  /usr/bin/run-
http...  16 seconds ago  Exited (1) 15 seconds ago  0.0.0.0:8080->8080/tcp
webserver1
```

You notice that the `webserver1` container did not start, so you run the `podman logs` command to view the logs of the container.

```
[appdev-adm@host ~]$ podman container logs webserver1
=> sourcing 10-set-ppm.sh ...
=> sourcing 20-copy-config.sh ...
=> sourcing 40-ssl-certs.sh ...
---> Generating SSL key pair for httpd...
AH00526: Syntax error on line 122 of /etc/httpd/conf/httpd.conf:
DocumentRoot '/var/www/html' is not a directory, or is not readable
[appdev-adm@servera ~]$
```

The logs of the `webserver1` container show that the `/var/www/html` file is not readable. This error occurs because the container mount directory cannot find the `html` subdirectory. You delete the existing container, update the command, and run it as follows:

```
[appdev-adm@host ~]$ podman run -d --name webserver1 -p 8080:8080 -v \
~/app-artifacts:/var/www/html:Z registry.access.redhat.com/ubi8/httpd-24
cde4a3d8c9563fd50cc39de8a4873dcf15a7e881ba4548d5646760eae7a35d81
[appdev-adm@host ~]$ podman ps
CONTAINER ID  IMAGE                                     COMMAND
CREATED      STATUS          PORTS          NAMES
cde4a3d8c956  registry.access.redhat.com/ubi8/httpd-24:latest  /usr/bin/run-
http...  4 seconds ago  Up 5 seconds ago  0.0.0.0:8080->8080/tcp  webserver1
```

## Create systemd User Files for Containers

You can manually define `systemd` services in the `~/.config/systemd/user/` directory. The file syntax for user services is the same as for the system services files. For more details, review the `systemd.unit(5)` and `systemd.service(5)` man pages.

Use the `podman generate systemd` command to generate `systemd` service files for an existing container. The `podman generate systemd` command uses a container as a model to create the configuration file.

The `podman generate systemd` command `--new` option instructs the `podman` utility to configure the `systemd` service to create the container when the service starts, and to delete the container when the service stops.



### Important

Without the `--new` option, the `podman` utility configures the service unit file to start and stop the existing container without deleting it.

You use the `podman generate systemd` command with the `--name` option to display the `systemd` service file that is modeled for the `webserver1` container.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1
...output omitted...
ExecStart=/usr/bin/podman start webserver1 ①
ExecStop=/usr/bin/podman stop -t 10 webserver1 ②
ExecStopPost=/usr/bin/podman stop -t 10 webserver1
...output omitted...
```

- ① On start, the `systemd` daemon executes the `podman start` command to start the existing container.
- ② On stop, the `systemd` daemon executes the `podman stop` command to stop the container. Notice that the `systemd` daemon does not delete the container on this action.

You then use the previous command with the addition of the `--new` option to compare the `systemd` configuration.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1 --new
...output omitted...
ExecStartPre=/bin/rm -f %t/%n.ctr-id
ExecStart=/usr/bin/podman run --cidfile=%t/%n.ctr-id --cgroups=no-common --rm --
sdnotify=common --replace -d --name webserver1 -p 8080:8080 -v /home/appdev-adm/
app-artifacts:/var/www/html:Z registry.access.redhat.com/ubi8/httpd-24 ❶
ExecStop=/usr/bin/podman stop --ignore --cidfile=%t/%n.ctr-id ❷
ExecStopPost=/usr/bin/podman rm -f --ignore --cidfile=%t/%n.ctr-id ❸
...output omitted...
```

- ① On start, the `systemd` daemon executes the `podman run` command to create and then start a new container. This action uses the `podman run` command `--rm` option, which removes the container on stop.
- ② On stop, `systemd` executes the `podman stop` command to stop the container.
- ③ After `systemd` has stopped the container, `systemd` removes it using the `podman rm -f` command.

You verify the output of the `podman generate systemd` command and run the previous command with the `--files` option to create the `systemd` user file in the current directory. Because the `webserver1` container uses persistent storage, you choose to use the `podman generate systemd` command with the `--new` option. You then create the `~/.config/systemd/user/` directory and move the file to this location.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1 --new --files
/home/appdev-adm/container-webserver1.service
[appdev-adm@host ~]$ mkdir -p ~/.config/systemd/user/
[appdev-adm@host ~]$ mv container-webserver1.service ~/.config/systemd/user/
```

## Manage `systemd` User Files for Containers

Now that you created the `systemd` user file, you can use the `systemctl` command `--user` option to manage the `webserver1` container.

First, you reload the `systemd` daemon to make the `systemctl` command aware of the new user file. You use the `systemctl --user start` command to start the `webserver1` container. Use the name of the generated `systemd` user file for the container.

```
[appdev-adm@host ~]$ systemctl --user start container-webserver1.service
[appdev-adm@host ~]$ systemctl --user status container-webserver1.service
● container-webserver1.service - Podman container-webserver1.service
   Loaded: loaded (/home/appdev-adm/.config/systemd/user/container-
webserver1.service; disabled; vendor preset: disabled)
     Active: active (running) since Thu 2022-04-28 21:22:26 EDT; 18s ago
```

```

Docs: man:podman-generate-systemd(1)
Process: 31560 ExecStartPre=/bin/rm -f /run/user/1003/container-
webserver1.service.ctr-id (code=exited, status=0/SUCCESS)
Main PID: 31600 (common)
...output omitted...
[appdev-adm@host ~]$ podman ps
CONTAINER ID  IMAGE                                     COMMAND
CREATED      STATUS          PORTS          NAMES
18eb00f42324  registry.access.redhat.com/ubi8/httpd-24:latest  /usr/bin/run-
http... 28 seconds ago  Up 29 seconds ago  0.0.0.0:8080->8080/tcp  webserver1
Created symlink /home/appdev-adm/.config/systemd/user/default.target.wants/
container-webserver1.service → /home/appdev-adm/.config/systemd/user/container-
webserver1.service.

```



### Important

When you configure a container with the `systemd` daemon, the daemon monitors the container status and restarts the container if it fails. Do not use the `podman` command to start or stop these containers. Doing so might interfere with the `systemd` daemon monitoring.

The following table summarizes the different directories and commands that are used between `systemd` system and user services.

#### Comparing System and User Services

Storing custom unit files	System services	<code>/etc/systemd/system/unit.service</code>
	User services	<code>~/.config/systemd/user/unit.service</code>
Reloading unit files	System services	<code># systemctl daemon-reload</code>
	User services	<code>\$ systemctl --user daemon-reload</code>
Starting and stopping a service	System services	<code># systemctl start UNIT</code> <code># systemctl stop UNIT</code>
	User services	<code>\$ systemctl --user start UNIT</code> <code>\$ systemctl --user stop UNIT</code>
Starting a service when the machine starts	System services	<code># systemctl enable UNIT</code>
	User services	<code>\$ logindctl enable-linger</code> <code>\$ systemctl --user enable UNIT</code>

## Configure Containers to Start at System Boot

Now that the `systemd` configuration for the container is complete, you exit the SSH session. Some time after, you are notified that the container stops after you exited the session.

You can change this default behavior and force your enabled services to start with the server and stop during the shutdown by running the `logindctl enable-linger` command.

You use the `logindctl` command to configure the `systemd` user service to persist after the last user session of the configured service closes. You then verify the successful configuration with the `logindctl show-user` command.

```
[user@host ~]$ logindctl show-user appdev-adm
...output omitted...
Linger=no
[user@host ~]$ logindctl enable-linger
[user@host ~]$ logindctl show-user appdev-adm
...output omitted...
Linger=yes
```

To revert the operation, use the `logindctl disable-linger` command.

## Manage Containers as Root with `systemd`

You can also configure containers to run as root and manage them with `systemd` service files. One advantage of this approach is that you can configure the service files to work exactly like common `systemd` unit files, rather than as a particular user.

The procedure to set the service file as root is similar to the previously outlined procedure for rootless containers, with the following exceptions:

- Do not create a dedicated user for container management.
- The service file must be in the `/etc/systemd/system` directory instead of in the `~/.config/systemd/user` directory.
- You manage the containers with the `systemctl` command without the `--user` option.
- Do not run the `logindctl enable-linger` command as the `root` user.

For a demonstration, see the YouTube video from the Red Hat Videos channel that is listed in the References at the end of this section.



### References

`logindctl(1)`, `systemd.unit(5)`, `systemd.service(5)`, `subuid(5)`, and `podman-generate-systemd(1)` man pages

#### Managing Containers in Podman with Systemd Unit Files

<https://www.youtube.com/watch?v=AGkM2jGT61Y>

For more information, refer to the *Running Containers as Systemd Services with Podman* chapter in the *Red Hat Enterprise Linux 9 Building, Running, and Managing Containers Guide* at

[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/9/html-single/building\\_running\\_and\\_managing\\_containers/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/index)

## ► Guided Exercise

# Manage Containers as System Services

In this exercise, you configure a container to manage it as a `systemd` service, and use `systemctl` commands to manage that container so that it automatically starts when the host machine starts.

## Outcomes

- Create `systemd` service files to manage a container.
- Configure a container so you can manage it with `systemctl` commands.
- Configure a user account for `systemd` user services to start a container when the host machine starts.

## Before You Begin

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start containers-services
```

## Instructions

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

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

- 2. Create a user account called `contsvc` and use `redhat` as the password. Use this user account to run containers as `systemd` services.

- 2.1. Create the `contsvc` user. Set `redhat` as the password for the `contsvc` user.

```
[student@servera ~]$ sudo useradd contsvc
[sudo] password for student: student
[student@servera ~]$ sudo passwd contsvc
Changing password for user contsvc.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 2.2. To manage the `systemd` user services with the `contsvc` account, you must log in directly as the `contsvc` user. You cannot use the `su` and `sudo` commands to create a session with the `contsvc` user.

**Chapter 16 |** Run Containers

Return to the workstation machine as the student user, and then log in as the contsvc user.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$ ssh contsvc@servera  
...output omitted...  
[contsvc@servera ~]$
```

- ▶ 3. Configure access to the `registry.lab.example.com` classroom registry in your home directory. Use the `/tmp/containers-services/registries.conf` file as a template.

- 3.1. Create the `~/.config/containers/` directory.

```
[contsvc@servera ~]$ mkdir -p ~/.config/containers/
```

- 3.2. The `lab` script prepares the `registries.conf` file in the `/tmp/containers-services/` directory. Copy that file to the `~/.config/containers/` directory.

```
[contsvc@servera ~]$ cp /tmp/containers-services/registries.conf \  
~/.config/containers/
```

- 3.3. Verify that you can access the `registry.lab.example.com` registry. If everything works as expected, then the command should list some images.

```
[contsvc@servera ~]$ podman search ubi  
NAME                                     DESCRIPTION  
registry.lab.example.com/ubi7/ubi  
registry.lab.example.com/ubi8/ubi  
registry.lab.example.com/ubi9-beta/ubi
```

- ▶ 4. Use the `/home/contsvc/webcontent/html/` directory as persistent storage for the web server container. Create the `index.html` test page with the `Hello World` line inside the directory.

- 4.1. Create the `~/webcontent/html/` directory.

```
[contsvc@servera ~]$ mkdir -p ~/webcontent/html/
```

- 4.2. Create the `index.html` file and add the `Hello World` line.

```
[contsvc@servera ~]$ echo "Hello World" > ~/webcontent/html/index.html
```

- 4.3. Confirm that the permission for others is set to `r--` in the `index.html` file. The container uses a non-privileged user that must be able to read the `index.html` file.

```
[contsvc@servera ~]$ ls -ld webcontent/html/  
drwxr-xr-x. 2 contsvc contsvc 24 Aug 28 04:56 webcontent/html/  
[contsvc@servera ~]$ ls -l webcontent/html/index.html  
-rw-r--r--. 1 contsvc contsvc 12 Aug 28 04:56 webcontent/html/index.html
```

- 5. Use the `registry.lab.example.com/rhel8/httpd-24:1-105` image to run a container called `webapp` in detached mode. Redirect the `8080` port on the local host to the container `8080` port. Mount the `~/webcontent` directory from the host to the `/var/www` directory in the container.
- 5.1. Log in to the `registry.lab.example.com` registry as the `admin` user with `redhat321` as the password.

```
[contsvc@servera ~]$ podman login registry.lab.example.com  
Username: admin  
Password: redhat321  
Login Succeeded!
```

- 5.2. Use the `registry.lab.example.com/rhel8/httpd-24:1-163` image to run a container called `webapp` in detached mode. Use the `-p` option to map the `8080` port on `servera` to the `8080` port in the container. Use the `-v` option to mount the `~/webcontent` directory on `servera` to the `/var/www` directory in the container.

```
[contsvc@servera ~]$ podman run -d --name webapp -p 8080:8080 -v \  
~/webcontent:/var/www:Z registry.lab.example.com/rhel8/httpd-24:1-163  
750a681bd37cb6825907e9be4347eec2c4cd79550439110fc6d41092194d0e06  
...output omitted...
```

- 5.3. Verify that the web service is working on port `8080`.

```
[contsvc@servera ~]$ curl http://localhost:8080  
Hello World
```

- 6. Create a `systemd` service file to manage the `webapp` container with `systemctl` commands. Configure the `systemd` service so that when you start the service, the `systemd` daemon creates a container. After you finish the configuration, stop and then delete the `webapp` container. Remember that the `systemd` daemon expects that the container does not exist initially.
- 6.1. Create and change to the `~/.config/systemd/user/` directory.

```
[contsvc@servera ~]$ mkdir -p ~/.config/systemd/user/  
[contsvc@servera ~]$ cd ~/.config/systemd/user/
```

- 6.2. Create the unit file for the `webapp` container. Use the `--new` option so that `systemd` creates a container when starting the service and deletes the container when stopping the service.

```
[contsvc@servera user]$ podman generate systemd --name webapp --files --new  
/home/contsvc/.config/systemd/user/container-webapp.service
```

- 6.3. Stop and then delete the webapp container.

```
[contsvc@servera user]$ podman stop webapp
webapp
[contsvc@servera user]$ podman rm webapp
750a681bd37cb6825907e9be4347eec2c4cd79550439110fc6d41092194d0e06
[contsvc@servera user]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

- ▶ 7. Reload the `systemd` daemon configuration, and then enable and start your new container-`webapp` user service. Verify the `systemd` service configuration, stop and start the service, and display the web server response and the container status.

- 7.1. Reload the configuration to recognize the new unit file.

```
[contsvc@servera user]$ systemctl --user daemon-reload
```

- 7.2. Enable and start the `container-webapp` service.

```
[contsvc@servera user]$ systemctl --user enable --now container-webapp
Created symlink /home/contsvc/.config/systemd/user/multi-user.target.wants/
container-webapp.service → /home/contsvc/.config/systemd/user/container-
webapp.service.
Created symlink /home/contsvc/.config/systemd/user/default.target.wants/container-
webapp.service → /home/contsvc/.config/systemd/user/container-webapp.service.
```

- 7.3. Verify that the web server responds to requests.

```
[contsvc@servera user]$ curl http://localhost:8080
Hello World
```

- 7.4. Verify that the container is running.

```
[contsvc@servera user]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
3e996db98071 registry.access.redhat.com/ubi8/httpd-24:1-163 /usr/bin/run-http...
3 minutes ago Up 3 minutes ago 0.0.0.0:8080->8080/tcp webapp
```

Notice the container ID. Use this information to confirm that `systemd` creates a container when you restart the service.

- 7.5. Stop the `container-webapp` service, and confirm that the container no longer exists. When you stop the service, `systemd` stops and then deletes the container.

```
[contsvc@servera user]$ systemctl --user stop container-webapp
[contsvc@servera user]$ podman ps --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

- 7.6. Start the `container-webapp` service, and then confirm that the container is running.

**Chapter 16 |** Run Containers

The container ID is different, because the `systemd` daemon creates a container with the start instruction and deletes the container with the stop instruction.

```
[contsvc@servera user]$ systemctl --user start container-webapp
[contsvc@servera user]$ podman ps
CONTAINER ID  IMAGE                               COMMAND
CREATED      STATUS      PORTS                 NAMES
4584b4df514c  registry.access.redhat.com/ubi8/httpd-24:1-163  /usr/bin/run-http...
6 seconds ago  Up 7 seconds ago  0.0.0.0:8080->8080/tcp  webapp
```

- 8. Ensure that the services for the `contsvc` user start at system boot. When done, restart the `servera` machine.

- 8.1. Run the `logindctl enable-linger` command.

```
[contsvc@servera user]$ logindctl enable-linger
```

- 8.2. Confirm that the `Linger` option is set for the `contsvc` user.

```
[contsvc@servera user]$ logindctl show-user contsvc
...output omitted...
Linger=yes
```

- 8.3. Switch to the `root` user, and then use the `systemctl reboot` command to restart `servera`.

```
[contsvc@servera user]$ su -
Password: redhat
Last login: Fri Aug 28 07:43:40 EDT 2020 on pts/0
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- 9. When the `servera` machine is up again, log in to `servera` as the `contsvc` user. Verify that `systemd` started the `webapp` container and that the web content is available.

- 9.1. Log in to `servera` as the `contsvc` user.

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

- 9.2. Verify that the container is running.

```
[contsvc@servera ~]$ podman ps
CONTAINER ID  IMAGE                               COMMAND
CREATED      STATUS      PORTS                 NAMES
6c325bf49f84  registry.access.redhat.com/ubi8/httpd-24:1-163  /usr/bin/run-http...
2 minutes ago  Up 2 minutes ago  0.0.0.0:8080->8080/tcp  webapp
```

- 9.3. Access the web content.

```
[contsvc@servera ~]$ curl http://localhost:8080  
Hello World
```

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

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

## Finish

On the workstation machine, run the `lab finish containers-services` script to complete this exercise.

```
[student@workstation ~]$ lab finish containers-services
```

This concludes the section.

## ▶ Lab

# Run Containers

In this lab, you configure a container on your server that provides a MariaDB database service, stores its database on persistent storage, and starts automatically with the server.

## Outcomes

- Create detached containers.
- Configure port redirection and persistent storage.
- Configure `systemd` for containers to start when the host machine starts.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. On `serverb`, install the container tools package.
2. The container image registry at `registry.lab.example.com` stores the `rhel8/mariadb-103` image with several tags. Use the `podsvc` user to list the available tags and note the tag with the *lowest* version number. Use the `admin` user and `redhat321` password to authenticate to the registry. Use the `/tmp/registries.conf` file as a template for the registry configuration.
3. As the `podsvc` user, create the `/home/podsvc/db_data` directory. Configure the directory so that containers have read/write access.
4. Create the `inventorydb` detached container. Use the `rhel8/mariadb-103` image from the `registry.lab.example.com` registry, and specify the tag with the lowest version number on that image, which you found in a preceding step. Map port 3306 in the container to port 13306 on the host. Mount the `/home/podsvc/db_data` directory on the host as `/var/lib/mysql/data` in the container. Declare the following variable values for the container:

Variable	Value
<code>MYSQL_USER</code>	<code>operator1</code>
<code>MYSQL_PASSWORD</code>	<code>redhat</code>
<code>MYSQL_DATABASE</code>	<code>inventory</code>
<code>MYSQL_ROOT_PASSWORD</code>	<code>redhat</code>

You can copy and paste these parameters from the /home/podsvc/containers-review/variables file on serverb. Execute the /home/podsvc/containers-review/testdb.sh script to confirm that the MariaDB database is running.

5. Configure the `systemd` daemon so that the `inventorydb` container starts automatically when the system boots.

## 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, 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 containers-review
```

This concludes the section.

## ► Solution

# Run Containers

In this lab, you configure a container on your server that provides a MariaDB database service, stores its database on persistent storage, and starts automatically with the server.

## Outcomes

- Create detached containers.
- Configure port redirection and persistent storage.
- Configure `systemd` for containers to start when the host machine starts.

## Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

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

## Instructions

1. On `serverb`, install the container tools package.

- 1.1. Log in to `serverb` as the `student` user.

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

- 1.2. Install the `container-tools` package.

```
[student@serverb ~]$ sudo dnf install container-tools
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

2. The container image registry at `registry.lab.example.com` stores the `rhel8/mariadb-103` image with several tags. Use the `podsvc` user to list the available tags and note the tag with the *lowest* version number. Use the `admin` user and `redhat321` password to authenticate to the registry. Use the `/tmp/registries.conf` file as a template for the registry configuration.

- 2.1. Return to the `workstation` machine as the `student` user.

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

- 2.2. Log in to `serverb` as the `podsvc` user.

```
[student@workstation ~]$ ssh podsvc@serverb
...output omitted...
[podsvc@serverb ~]$
```

- 2.3. Configure access to the `registry.lab.example.com` classroom registry in your home directory. Use the `/tmp/registries.conf` file as a template.

```
[podsvc@serverb ~]$ mkdir -p ~/.config/containers/
[podsvc@serverb ~]$ cp /tmp/registries.conf \
~/.config/containers/
```

- 2.4. Log in to the container registry with the `podman login` command.

```
[podsvc@serverb ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

- 2.5. View information about the `registry.lab.example.com/rhel8/mariadb-103` image.

```
[podsvc@serverb ~]$ skopeo inspect \
docker://registry.lab.example.com/rhel8/mariadb-103
{
  "Name": "registry.lab.example.com/rhel8/mariadb-103",
  "Digest": "sha256:a95b...4816",
  "RepoTags": [
    "1-86",
    "1-102",
    "latest"
  ],
  ...output omitted...
```

The lowest version tag is the `1-86` version.

3. As the `podsvc` user, create the `/home/podsvc/db_data` directory. Configure the directory so that containers have read/write access.

- 3.1. Create the `/home/podsvc/db_data` directory.

```
[podsvc@serverb ~]$ mkdir /home/podsvc/db_data
```

- 3.2. Set the access mode of the directory to 777 so that everyone has read/write access.

```
[podsvc@serverb ~]$ chmod 777 /home/podsvc/db_data
```

4. Create the `inventorydb` detached container. Use the `rhel8/mariadb-103` image from the `registry.lab.example.com` registry, and specify the tag with the lowest version number on that image, which you found in a preceding step. Map port 3306 in the container to port 13306 on the host. Mount the `/home/podsvc/db_data` directory on the host as `/var/lib/mysql/data` in the container. Declare the following variable values for the container:

Variable	Value
<code>MYSQL_USER</code>	<code>operator1</code>
<code>MYSQL_PASSWORD</code>	<code>redhat</code>
<code>MYSQL_DATABASE</code>	<code>inventory</code>
<code>MYSQL_ROOT_PASSWORD</code>	<code>redhat</code>

You can copy and paste these parameters from the `/home/podsvc/containers-review/variables` file on `serverb`. Execute the `/home/podsvc/containers-review/testdb.sh` script to confirm that the MariaDB database is running.

- 4.1. Create the container.

```
[podsvc@serverb ~]$ podman run -d --name inventorydb -p 13306:3306 \
-e MYSQL_USER=operator1 \
-e MYSQL_PASSWORD=redhat \
-e MYSQL_DATABASE=inventory \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/podsvc/db_data:/var/lib/mysql/data:Z \
registry.lab.example.com/rhel8/mariadb-103:1-86
...output omitted...
```

- 4.2. Confirm that the database is running.

```
[podsvc@serverb ~]$ ~/containers-review/testdb.sh
Testing the access to the database...
SUCCESS
```

5. Configure the `systemd` daemon so that the `inventorydb` container starts automatically when the system boots.

- 5.1. If you used `sudo` or `su` to log in as the `podsvc` user, then exit `serverb` and use the `ssh` command to directly log in to `serverb` as the `podsvc` user. Remember, the `systemd` daemon requires the user to open a direct session from the console or through SSH. Omit this step if you already logged in to the `serverb` machine as the `podsvc` user by using SSH.

```
[student@workstation ~]$ ssh podsvc@serverb
...output omitted...
[podsvc@serverb ~]$
```

5.2. Create the `~/.config/systemd/user/` directory.

```
[podsvc@serverb ~]$ mkdir -p ~/.config/systemd/user/
```

5.3. Create the `systemd` unit file from the running container.

```
[podsvc@serverb ~]$ cd ~/.config/systemd/user/
[podsvc@serverb user]$ podman generate systemd --name inventorydb --files --new
/home/podsvc/.config/systemd/user/container-inventorydb.service
```

5.4. Stop and then delete the `inventorydb` container.

```
[podsvc@serverb user]$ podman stop inventorydb
inventorydb
[podsvc@serverb user]$ podman rm inventorydb
0d28f0e0a4118ff019691e34afe09b4d28ee526079b58d19f03b324bd04fd545
```

5.5. Instruct the `systemd` daemon to reload its configuration, and then enable and start the `container-inventorydb` service.

```
[podsvc@serverb user]$ systemctl --user daemon-reload
[podsvc@serverb user]$ systemctl --user enable --now container-inventorydb.service
Created symlink /home/podsvc/.config/systemd/user/multi-user.target.wants/
container-inventorydb.service → /home/podsvc/.config/systemd/user/container-
inventorydb.service.
Created symlink /home/podsvc/.config/systemd/user/default.target.wants/
container-inventorydb.service → /home/podsvc/.config/systemd/user/container-
inventorydb.service.
```

5.6. Confirm that the container is running.

```
[podsvc@serverb user]$ ~/containers-review/testdb.sh
Testing the access to the database...
SUCCESS
[podsvc@serverb user]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED
      STATUS PORTS NAMES
3ab24e7f000d registry.lab.example.com/rhel8/mariadb-103:1-86 run-mysqld 47
seconds ago Up 46 seconds ago 0.0.0.0:13306->3306/tcp inventorydb
```

5.7. Run the `loginctl enable-linger` command for the user services to start automatically when the server starts.

```
[podsvc@serverb ~]$ loginctl enable-linger
```

5.8. Return to the `workstation` machine as the `student` user.

```
[podsvc@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 containers-review
```

## 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 containers-review
```

This concludes the section.

# Summary

---

- Containers provide a lightweight way to distribute and run an application with its dependencies so that it does not conflict with installed software on the host.
- Containers run from container images that you can download from a container registry or create yourself.
- You can use container files with instructions to build a customized container image.
- Podman, which Red Hat Enterprise Linux provides, directly runs and manages containers and container images on a single host.
- You can run containers as `root`, or as non-privileged rootless containers for increased security.
- You can map network ports on the container host to pass traffic to services that run in its containers.
- You can use environment variables to configure the software in containers at build time.
- Container storage is temporary, but you can attach persistent storage to a container by using the contents of a directory on the container host, for example.
- You can configure a `systemd` unit file to automatically run containers when the system starts.

## Chapter 17

# Comprehensive Review

### Goal

Review tasks from *RHCSA Rapid Track*

### Objectives

- Review tasks from *RHCSA Rapid Track*

### Sections

- Comprehensive Review

### Labs

- Fix Boot Issues and Maintain Servers
- Configure and Manage File Systems and Storage
- Configure and Manage Server Security
- Run Containers

# Comprehensive Review

---

## Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills learned in *RHCSA Rapid Track*.

## Reviewing RHCSA Rapid Track

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter.

You can refer to earlier sections in the textbook for extra study.

### **Chapter 1, Access Systems and Obtaining Support**

Edit text files, log in to local and remote Linux system, and investigate problem resolution methods provided through Red Hat Support and Red Hat Insights.

- Create and edit text files from the command line with the vim editor.
- Configure a user account to use key-based authentication to log in to remote systems securely without a password.
- Describe and use Red Hat Customer Portal key resources to find information from Red Hat documentation and the Knowledgebase.
- Use Red Hat Insights to analyze servers for issues, remediate or resolve them, and confirm that the solution worked.

### **Chapter 2, Manage Files from the Command Line**

Copy, move, create, delete, and organize files from the Bash shell.

- Describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.
- Make multiple file names reference the same file with hard links and symbolic (or "soft") links.
- Efficiently run commands that affect many files by using pattern matching features of the Bash shell.

### **Chapter 3, Manage Local Users and Groups**

Create, manage, and delete local users and groups, and administer local password policies.

- Describe the purpose of users and groups on a Linux system.
- Switch to the superuser account to manage a Linux system, and grant other users superuser access through the sudo command.
- Create, modify, and delete local user accounts.
- Create, modify, and delete local group accounts.

- Set a password management policy for users, and manually lock and unlock user accounts.

## **Chapter 4, Control Access to Files**

Set Linux file-system permissions on files and to interpret the security effects of different permission settings.

- Change the permissions and ownership of files with command-line tools.
- Control the default permissions of user-created files, explain the effect of special permissions, and use special and default permissions to set the group owner of files that are created in a directory.

## **Chapter 5, Manage SELinux Security**

Protect and manage server security by using SELinux.

- Explain how SELinux protects resources, change the current SELinux mode of a system, and set the default SELinux mode of a system.
- Manage the SELinux policy rules that determine the default context for files and directories with the `semanage fcontext` command and apply the context defined by the SELinux policy to files and directories with the `restorecon` command.
- Activate and deactivate SELinux policy rules with the `setsebool` command, manage the persistent value of SELinux Booleans with the `semanage boolean -l` command, and consult `man` pages that end with `_selinux` to find useful information about SELinux Booleans.
- Use SELinux log analysis tools and display useful information during SELinux troubleshooting with the `sealert` command.

## **Chapter 6, Tune System Performance**

Evaluate and control processes, set tuning parameters and adjust process scheduling priorities on a Red Hat Enterprise Linux system.

- Use commands to kill and communicate with processes, define the characteristics of a daemon process, and stop user sessions and processes.
- Define load average and determine resource-intensive server processes.
- Optimize system performance by selecting a tuning profile that the `tuned` daemon manages.
- Prioritize or deprioritize specific processes, with the `nice` and `renice` commands.

## **Chapter 7, Schedule Future Tasks**

Schedule tasks to automatically execute in the future.

- Schedule commands to run on a repeating schedule with a user's `crontab` file.
- Schedule commands to run on a repeating schedule with the system `crontab` file and directories.
- Enable and disable `systemd` timers, and configure a timer that manages temporary files.

## **Chapter 8, Install and Update Software Packages**

Download, install, update, and manage software packages from Red Hat and DNF package repositories.

**Chapter 17 | Comprehensive Review**

- Register a system to your Red Hat account and assign it entitlements for software updates and support services with Red Hat Subscription Management.
- Find, install, and update software packages with the dnf command.
- Enable and disable server use of Red Hat or third-party DNF repositories.

## **Chapter 9, Manage Basic Storage**

Create and manage storage devices, partitions, file systems, and swap spaces from the command line.

- Access the contents of file systems by adding and removing file systems in the file-system hierarchy.
- Create storage partitions, format them with file systems, and mount them for use.
- Create and manage swap spaces to supplement physical memory.

## **Chapter 10, Manage Storage Stack**

Create and manage logical volumes that contain file systems or swap spaces from the command line.

- Describe logical volume manager components and concepts, and implement LVM storage and display LVM component information.
- Analyze the multiple storage components that make up the layers of the storage stack.

## **Chapter 11, Control Services and Boot Process**

Control and monitor network services, system daemons and the boot process using `systemd`

- List system daemons and network services that were started by the systemd service and socket units.
- Control system daemons and network services with systemctl.
- Describe the Red Hat Enterprise Linux boot process, set the default target when booting, and boot a system to a non-default target.
- Log in to a system and change the root password when the current root password is lost.
- Manually repair file-system configuration or corruption issues that stop the boot process.

## **Chapter 12, Analyze and Store Logs**

Locate and accurately interpret system event logs for troubleshooting purposes.

- Describe the basic Red Hat Enterprise Linux logging architecture to record events.
- Interpret events in relevant syslog files to troubleshoot problems or review system status.
- Find and interpret entries in the system journal to troubleshoot problems or review system status.
- Configure the system journal to preserve the record of events when a server is rebooted.
- Maintain accurate time synchronization with Network Time Protocol (NTP) and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

## **Chapter 13, Manage Networking**

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

- Test and inspect the current network configuration with command-line utilities.
- Manage network settings and devices with the nmcli command.
- Modify network configuration by editing configuration files.
- Configure a server's static hostname and its name resolution and test the results.

## **Chapter 14, Access Network-Attached Storage**

Access network-attached storage with the NFS protocol.

- Identify NFS export information, create a directory to use as a mount point, mount an NFS export with the `mount` command or by configuring the `/etc/fstab` file, and unmount an NFS export with the `umount` command.
- Describe the benefits of using the automounter, and automount NFS exports by using direct and indirect maps.

## **Chapter 15, Manage Network Security**

Control network connections to services using the system firewall.

- Accept or reject network connections to system services with `firewalld` rules.

## **Chapter 16, Run Containers**

Obtain, run, and manage simple lightweight services as containers on a single Red Hat Enterprise Linux server.

- Explain container concepts and the core technologies for building, storing, and running containers.
- Discuss container management tools for using registries to store and retrieve images, and for deploying, querying, and accessing containers.
- Provide persistent storage for container data by sharing storage from the container host, and configure a container network.
- Configure a container as a `systemd` service, and configure a container service to start at boot time.

## ► Lab

# Fix Boot Issues and Maintain Servers



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you troubleshoot and repair boot problems and update the system default target. You also schedule tasks to run on a repeating schedule as a normal user.

## Outcomes

- Diagnose issues and recover the system from emergency mode.
- Change the default target from `graphical.target` to `multi-user.target`.
- Schedule recurring jobs to run as a normal user.

## Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-compreview1
```

## Specifications

- On `workstation`, run the `/tmp/rhcsa-break1` script. This script causes an issue with the boot process on `serverb` and then reboots the machine. Troubleshoot the cause and repair the boot issue. When prompted, use `redhat` as the password of the `root` user.
- On `workstation`, run the `/tmp/rhcsa-break2` script. This script causes the default target to switch from the `multi-user` target to the `graphical` target on the `serverb` machine and then reboots the machine. On `serverb`, reset the default target to use the `multi-user` target. The default target settings must persist after reboot without manual intervention. As the `student` user, use the `sudo` command for performing privileged commands. Use `student` as the password, when required.
- On `serverb`, schedule a recurring job as the `student` user that executes the `/home/student/backup-home.sh` script hourly between 7 PM and 9 PM every day except on Saturday and Sunday. Download the backup script from <http://materials.example.com/labs/backup-home.sh>. The `backup-home.sh` script backs up the `/home/student` directory from

serverb to servera in the /home/student/serverb-backup directory. Use the backup-home.sh script to schedule the recurring job as the student user.

- 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.

```
[student@workstation ~]$ lab grade rhcsa-comprevew1
```

## 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 rhcsa-comprevew1
```

This concludes the section.

## ► Solution

# Fix Boot Issues and Maintain Servers



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you troubleshoot and repair boot problems and update the system default target. You also schedule tasks to run on a repeating schedule as a normal user.

## Outcomes

- Diagnose issues and recover the system from emergency mode.
- Change the default target from `graphical.target` to `multi-user.target`.
- Schedule recurring jobs to run as a normal user.

## Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-compreview1
```

1. On `workstation`, run the `/tmp/rhcsa-break1` script.

```
[student@workstation ~]$ sh /tmp/rhcsa-break1
```

2. After the `serverb` machine boots, access the console and notice that the boot process stopped early. Take a moment to speculate about a possible cause for this behavior.
  - 2.1. Locate the icon for the `serverb` console, as appropriate for your classroom environment. Open the console and inspect the error. It might take a few seconds for the error to appear.
  - 2.2. Press `Ctrl+Alt+Del` to reboot the `serverb` machine. When the boot-loader menu appears, press any key except `Enter` to interrupt the countdown.
  - 2.3. Edit the default boot-loader entry, in memory, to log in to the emergency mode. Press `e` to edit the current entry.

- 2.4. Use the cursor keys to navigate to the line that starts with `linux`. Append `systemd.unit=emergency.target`.
- 2.5. Press `Ctrl+x` to boot with the modified configuration.
- 2.6. Log in to emergency mode. Use `redhat` as the `root` user's password.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

3. Remount the `/` file system with read and write capabilities. Use the `mount -a` command to attempt to mount all the other file systems.
  - 3.1. Remount the `/` file system with read and write capabilities to edit the file system.

```
[root@serverb ~]# mount -o remount,rw /
```

- 3.2. Attempt to mount all the other file systems. Notice that one of the file systems does not mount.

```
[root@serverb ~]# mount -a
...output omitted...
mount: /FakeMount: can't find UUID=fake.
```

- 3.3. Edit the `/etc/fstab` file to fix the issue. Remove or comment out the incorrect line.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
#UUID=fake      /FakeMount  xfs    defaults    0 0
```

- 3.4. Update the `systemd` daemon for the system to register the new `/etc/fstab` file configuration.

```
[root@serverb ~]# systemctl daemon-reload
[ 206.828912] systemd[1]: Reloading.
```

- 3.5. Verify that `/etc/fstab` file is now correct by attempting to mount all entries.

```
[root@serverb ~]# mount -a
```

- 3.6. Reboot `serverb` and wait for the boot to complete. The system should now boot without errors.

```
[root@serverb ~]# systemctl reboot
```

4. On `workstation`, run the `/tmp/rhcsa-break2` script. Wait for the `serverb` machine to reboot before proceeding.

```
[student@workstation ~]$ sh /tmp/rhcsa-break2
```

5. On **serverb**, set the **multi-user** target as the current and default target.

5.1. Log in to **serverb** as the **student** user.

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

5.2. Determine the default target.

```
[student@serverb ~]$ systemctl get-default
graphical.target
```

5.3. Switch to the **multi-user** target.

```
[student@serverb ~]$ sudo systemctl isolate multi-user.target
[sudo] password for student: student
```

5.4. Set the **multi-user** target as the default target.

```
[student@serverb ~]$ sudo 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.
```

5.5. Reboot **serverb** and verify that the **multi-user** target is set as the default target.

```
[student@serverb ~]$ sudo systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

5.6. After the system reboots, open an SSH session to **serverb** as the **student** user. Verify that the **multi-user** target is set as the default target.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ systemctl get-default
multi-user.target
```

6. On **serverb**, schedule a recurring job as the **student** user that executes the **/home/student/backup-home.sh** script hourly between 7 PM and 9 PM on all days except Saturday and Sunday. Use the **backup-home.sh** script to schedule the recurring job. Download the backup script from <http://materials.example.com/labs/backup-home.sh>.

6.1. On **serverb**, download the backup script from <http://materials.example.com/labs/backup-home.sh>. Use **chmod** to make the backup script executable.

```
[student@serverb ~]$ wget http://materials.example.com/labs/backup-home.sh
...output omitted...
[student@serverb ~]$ chmod +x backup-home.sh
```

6.2. Open the crontab file with the default text editor.

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

6.3. Edit the file to add the following line:

```
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

Save the changes and exit the editor.

6.4. Use the `crontab -l` command to list the scheduled recurring jobs.

```
[student@serverb ~]$ crontab -l
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

7. Reboot `serverb` and wait for the boot to complete before grading.

```
[student@serverb ~]$ sudo systemctl reboot
[sudo] password for student: student
Connection to serverb closed by remote host.
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 rhcsa-compreview1
```

## 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 rhcsa-compreview1
```

This concludes the section.

## ▶ Lab

# Configure and Manage File Systems and Storage



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you create a logical volume, mount a network file system, and create a swap partition that is automatically activated at boot. You also configure directories to store temporary files.

### 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.

### Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-comprevew2
```

### Specifications

- On **serverb**, configure a new 1 GiB **vol\_home** logical volume in a new 2 GiB **extra\_storage** volume group. Use the unpartitioned **/dev/vdb** disk to create the partition.
- Format the **vol\_home** logical volume with the **XFS** file-system type, and persistently mount it on the **/user-homes** directory.
- On **serverb**, persistently mount the **/share** network file system that **servera** exports on the **/local-share** directory. The **servera** machine exports the **servera.lab.example.com:/share** path.

- On **serverb**, create a new 512 MiB swap partition on the **/dev/vdc** disk. Persistently mount the swap partition.
- Create the **production** user group. Create the **production1**, **production2**, **production3**, and **production4** users with the **production** group as their supplementary group.
- On **serverb**, configure the **/run/volatile** directory to store temporary files. If the files in this directory are not accessed for more than 30 seconds, then the system automatically deletes them. Set **0700** as the octal permissions for the 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.

```
[student@workstation ~]$ lab grade rhcsa-comprevew2
```

## 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 rhcsa-comprevew2
```

This concludes the section.

## ► Solution

# Configure and Manage File Systems and Storage



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you create a logical volume, mount a network file system, and create a swap partition that is automatically activated at boot. You also configure directories to store temporary files.

## 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.

## Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-compreview2
```

1. On **serverb**, configure a new 1 GiB **vol\_home** logical volume in a new 2 GiB **extra\_storage** volume group. Use the unpartitioned **/dev/vdb** disk to create the partition.
  - 1.1. Log in to **serverb** as the **student** user and switch to the **root** user.

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

- 1.2. Create a 2 GiB partition on the /dev/vdb disk.

```
[root@serverb ~]# parted /dev/vdb mklabel msdos
...output omitted...
[root@serverb ~]# parted /dev/vdb mkpart primary 1GiB 3GiB
...output omitted...
[root@serverb ~]# parted /dev/vdb set 1 lvm on
...output omitted...
```

- 1.3. Declare the /dev/vdb1 block device as a physical volume.

```
[root@serverb ~]# pvcreate /dev/vdb1
...output omitted...
```

- 1.4. Create the extra\_storage volume group with the /dev/vdb1 partition.

```
[root@serverb ~]# vgcreate extra_storage /dev/vdb1
...output omitted...
```

- 1.5. Create the 1 GiB vol\_home logical volume.

```
[root@serverb ~]# lvcreate -L 1GiB -n vol_home extra_storage
...output omitted...
```

2. Format the vol\_home logical volume with the XFS file-system type, and persistently mount it on the /user-homes directory.

- 2.1. Create the /user-homes directory.

```
[root@serverb ~]# mkdir /user-homes
```

- 2.2. Format the /dev/extrastorage/vol\_home partition with the XFS file-system type.

```
[root@serverb ~]# mkfs -t xfs /dev/extrastorage/vol_home
...output omitted...
```

- 2.3. Persistently mount the /dev/extrastorage/vol\_home partition on the /user-homes directory. Use the partition's UUID for the /etc/fstab file entry.

```
[root@serverb ~]# lsblk -o UUID /dev/extrastorage/vol_home
UUID
988cf149-0667-4733-abca-f80c6ec50ab6
[root@serverb ~]# echo "UUID=988c...0ab6 /user-homes xfs defaults 0 0" \
>> /etc/fstab
[root@serverb ~]# mount /user-homes
```

3. On serverb, persistently mount the /share network file system that servera exports on the /local-share directory. The servera machine exports the servera.lab.example.com:/share path.

- 3.1. Create the /local-share directory.

```
[root@serverb ~]# mkdir /local-share
```

- 3.2. Append the appropriate entry to the /etc/fstab file to persistently mount the servera.lab.example.com:/share network file system.

```
[root@serverb ~]# echo "servera.lab.example.com:/share /local-share \
nfs rw,sync 0 0" >> /etc/fstab
```

- 3.3. Mount the network file system on the /local-share directory.

```
[root@serverb ~]# mount /local-share
```

4. On serverb, create a 512 MiB swap partition on the /dev/vdc disk. Activate and persistently mount the swap partition.

- 4.1. Create a 512 MiB partition on the /dev/vdc disk.

```
[root@serverb ~]# parted /dev/vdc mklabel msdos
...output omitted...
[root@serverb ~]# parted /dev/vdc mkpart primary linux-swap 1MiB 513MiB
...output omitted...
```

- 4.2. Create the swap space on the /dev/vdc1 partition.

```
[root@serverb ~]# mkswap /dev/vdc1
...output omitted...
```

- 4.3. Create an entry in the /etc/fstab file to persistently mount the swap space. Use the partition's UUID to create the /etc/fstab file entry. Activate the swap space.

```
[root@serverb ~]# lsblk -o UUID /dev/vdc1
UUID
cc18ccb6-bd29-48a5-8554-546bf3471b69
[root@serverb ~]# echo "UUID=cc18...1b69 swap swap defaults 0 0" >> /etc/fstab
[root@serverb ~]# swapon -a
```

5. Create the production user group. Then, create the production1, production2, production3, and production4 users with the production group as their supplementary group.

```
[root@serverb ~]# groupadd production
[root@serverb ~]# for i in 1 2 3 4; do useradd -G production production$i; done
```

6. On serverb, configure the /run/volatile directory to store temporary files. If the files in this directory are not accessed for more than 30 seconds, then the system automatically deletes them. Set 0700 as the octal permissions for the directory. Use the /etc/tmpfiles.d/volatile.conf file to configure the time-based deletion of the files in the /run/volatile directory.

6.1. Create the `/etc/tmpfiles.d/volatile.conf` file with the following content:

```
d /run/volatile 0700 root root 30s
```

6.2. Use the `systemd-tmpfiles --create` command to create the `/run/volatile` directory if it does not exist.

```
[root@serverb ~]# systemd-tmpfiles --create /etc/tmpfiles.d/volatile.conf
```

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

```
[student@serverb ~]$ exit  
logout  
Connection to serverb 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 rhcsa-comprevew2
```

## 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 rhcsa-comprevew2
```

This concludes the section.

## ► Lab

# Configure and Manage Server Security



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you configure SSH key-based authentication, change firewall settings, adjust the SELinux mode and an SELinux Boolean, and troubleshoot SELinux issues.

## Outcomes

- Configure SSH key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

## Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-comprevew3
```

## Specifications

- On **serverb**, generate an SSH key pair for the **student** user. Do not protect the private key with a passphrase.
- Configure the **student** user on **servera** to accept login authentication with the SSH key pair that you generated on the **serverb** machine. The **student** user on **serverb** must be able to log in to **servera** via SSH without entering a password.
- On **servera**, check the **/user-homes/production5** directory permissions. Then, configure SELinux to run in the permissive mode by default.
- On **serverb**, verify that the **/localhome** directory does not exist. Then, configure the **production5** user's home directory to mount the **/user-homes/production5** network file system. The **servera.lab.example.com** machine exports the file system as the **servera.lab.example.com:/user-homes/production5** NFS share. Use the

autofs service to mount the network share. Verify that the autofs service creates the /localhome/production5 directory with the same permissions as on servera.

- On serverb, adjust the appropriate SELinux Boolean so that the production5 user may use the NFS-mounted home directory after authenticating with an SSH key. If required, use redhat as the password of the production5 user.
- On serverb, adjust the firewall settings to reject all connection requests from the servera machine. Use the servera IPv4 address (172.25.250.10) to configure the firewall rule.
- On serverb, investigate and fix the issue with the failing Apache web service, which listens on port 30080/TCP for connections. Adjust the firewall settings appropriately so that the port 30080/TCP is open for 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.

```
[student@workstation ~]$ lab grade rhcsa-comprevew3
```

## 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 rhcsa-comprevew3
```

This concludes the section.

## ► Solution

# Configure and Manage Server Security



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

In this review, you configure SSH key-based authentication, change firewall settings, adjust the SELinux mode and an SELinux Boolean, and troubleshoot SELinux issues.

## Outcomes

- Configure SSH key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

## Before You Begin

If you did not reset your **workstation** and **server** machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the **student** user on the **workstation** machine, use the **lab** command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-comprevew3
```

- On **serverb**, generate an SSH key pair for the **student** user. Do not protect the private key with a passphrase.

- Log in to **serverb** as the **student** user.

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

- Use the **ssh-keygen** command to generate an SSH key pair. Do not protect the private key with a passphrase.

```
[student@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
```

```
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+ijpGqjEQSGBR80RNchiRTHw/URQksVdHjsHqVBXeYI student@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]---+
|+BBX+o*+o..=+.. |
|+.0.0000 .oE+o . |
|.+ . . . .+ .o |
|. . o . o |
| . .S |
|... . |
|.o. .. |
|o .o o |
|..o.... |
+---[SHA256]---+
```

2. Configure the **student** user on **servera** to accept login authentication with the SSH key pair that you generated on the **serverb** machine. The **student** user on **serverb** must be able to log in to **servera** via SSH without entering a password.
  - 2.1. Send the public key of the newly generated SSH key pair to the **student** user on the **servera** machine.

```
[student@serverb ~]$ ssh-copy-id student@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/student/.ssh/
id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be established.
ED25519 key fingerprint is SHA256:shYfoFG0Nnv42pv7j+HG+FISmCAm4Bh5jfjwwSMJbrw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
student@servera's password: student

Number of key(s) added: 1

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

- 2.2. Verify that the **student** user can log in to **servera** from **serverb** without entering a password. Do not close the connection.

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

3. On **servera**, check the **/user-homes/production5** directory permissions. Then, configure SELinux to run in the permissive mode by default.

- 3.1. Check the /user-homes/production5 directory permissions.

```
[student@servera ~]$ ls -ld /user-homes/production5
drwx----- 2 production5 production5 62 May  6 05:27 /user-homes/production5
```

- 3.2. Edit the /etc/sysconfig/selinux file to set the SELINUX parameter to the permissive value.

```
[student@servera ~]$ sudo vi /etc/sysconfig/selinux
...output omitted...
#SELINUX=enforcing
SELINUX=permissive
...output omitted...
```

- 3.3. Reboot the system.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@serverb ~]$
```

4. On serverb, verify that the /localhome directory does not exist. Then, configure the production5 user's home directory to mount the /user-homes/production5 network file system. The servera.lab.example.com machine exports the file system as the servera.lab.example.com:/user-homes/production5 NFS share. Use the autofs service to mount the network share. Verify that the autofs service creates the /localhome/production5 directory with the same permissions as on servera.

- 4.1. Verify that the /localhome directory does not exist.

```
[student@serverb ~]$ ls -ld /localhome
ls: cannot access '/localhome': No such file or directory
```

- 4.2. On serverb, switch to the root user.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 4.3. Install the autofs package.

```
[root@serverb ~]# dnf install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  autofs-1:5.1.7-27.el9.x86_64      libsss_autofs-2.6.2-2.el9.x86_64

Complete!
```

- 4.4. Create the /etc/auto.master.d/production5.autofs map file with the following content:

```
/- /etc/auto.production5
```

- 4.5. Determine the production5 user's home directory.

```
[root@serverb ~]# getent passwd production5
production5:x:5001:5001::/localhome/production5:/bin/bash
```

- 4.6. Create the /etc/auto.production5 file with the following content:

```
/localhome/production5 -rw servera.lab.example.com:/user-homes/production5
```

- 4.7. Restart the autofs service.

```
[root@serverb ~]# systemctl restart autofs
```

- 4.8. Verify that the autofs service creates the /localhome/production5 directory with the same permissions as on servera.

```
[root@serverb ~]# ls -ld /localhome/production5
drwx----- 2 production5 production5 62 May  6 05:52 /localhome/production5
```

5. On serverb, adjust the appropriate SELinux Boolean so that the production5 user may use the NFS-mounted home directory after authenticating with an SSH key. If required, use redhat as the password of the production5 user.

- 5.1. Open a new terminal window and verify from servera that the production5 user is not able to log in to serverb with SSH key-based authentication. An SELinux Boolean is preventing the user from logging in. From workstation, open a new terminal and log in to servera as the student user.

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

- 5.2. Switch to the production5 user. When prompted, use redhat as the password of the production5 user.

```
[student@servera ~]$ su - production5
Password: redhat
[production5@servera ~]$
```

- 5.3. Generate an SSH key pair.

```
[production5@servera ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/production5/.ssh/id_rsa): Enter
Created directory '/home/production5/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/production5/.ssh/id_rsa.
```

```
Your public key has been saved in /home/production5/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AbUcIBXneyiGIhr4wS1xzs3WqDvbTP+eZuSRn9HQ/cw
    production5@servera.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
| ..=++      |
| . = o      |
| . . =     . . |
| .. * + o + . . . |
| += = B S .. o o.|
| .+ + + .+ . . E|
| . . . . o o o |
| . = . +.o   |
|   ooo .=+   |
+---[SHA256]-----+
```

- 5.4. Transfer the public key of the SSH key pair to the `production5` user on the `serverb` machine. When prompted, use `redhat` as the password of the `production5` user.

```
[production5@servera ~]$ ssh-copy-id production5@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
production5/.ssh/id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ciCkaRWF4g6eR9nSdPxQ7KL8czpViXal6BousK544TY.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
production5@serverb's password: redhat

Number of key(s) added: 1

Now try logging in to the machine, with: "ssh 'production5@serverb'"
and check to make sure that only the key(s) you wanted were added.
```

- 5.5. Use SSH public key-based authentication instead of password-based authentication to log in to `serverb` as the `production5` user. This command should fail.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
production5@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-
mic,password).
```

- 5.6. On the terminal that is connected to `serverb` as the `root` user, set the `use_nfs_home_dirs` SELinux Boolean to `true`.

```
[root@serverb ~]# setsebool -P use_nfs_home_dirs true
```

- 5.7. Return to the terminal that is connected to `servera` as the `production5` user, and use SSH public key-based authentication instead of password-based authentication to log in to `serverb` as the `production5` user. This command should succeed.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
...output omitted...
[production5@serverb ~]$
```

- 5.8. Exit and close the terminal that is connected to serverb as the production5 user. Keep open the terminal that is connected to serverb as the root user.
6. On serverb, adjust the firewall settings to reject all connection requests that originate from the servera machine. Use the servera IPv4 address (172.25.250.10) to configure the firewall rule.
- 6.1. Add the IPv4 address of servera to the block zone.

```
[root@serverb ~]# firewall-cmd --add-source=172.25.250.10/32 \
--zone=block --permanent
success
```

- 6.2. Reload the changes in the firewall settings.
7. On serverb, investigate and fix the issue with the failing Apache web service, which listens on port 30080/TCP for connections. Adjust the firewall settings appropriately so that the port 30080/TCP is open for incoming connections.
- 7.1. Restart the httpd service. This command fails to restart the service.

```
[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.
```

- 7.2. Investigate why the httpd service is failing. Notice the permission error that indicates that the httpd daemon failed to bind to port 30080/TCP on startup. SELinux policies can prevent an application from binding to a non-standard port. Press q to quit the command.

```
[root@serverb ~]# systemctl status httpd.service
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor
preset: disabled)
     Active: failed (Result: exit-code) since Mon 2022-05-02 13:20:46 EDT; 29s ago
       Docs: man:httpd.service(8)
     Process: 2322 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
    Main PID: 2322 (code=exited, status=1/FAILURE)
      Status: "Reading configuration..."
        CPU: 30ms
```

```
May 02 13:20:46 serverb.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
May 02 13:20:46 serverb.lab.example.com httpd[2322]: (13)Permission denied:
AH00072: make_sock: could not bind to address [::]:30080
May 02 13:20:46 serverb.lab.example.com httpd[2322]: (13)Permission denied:
AH00072: make_sock: could not bind to address 0.0.0.0:30080
May 02 13:20:46 serverb.lab.example.com httpd[2322]: no listening sockets
available, shutting down
...output omitted...
```

- 7.3. Determine whether an SELinux policy is preventing the `httpd` service from binding to port 30080/TCP. The log messages reveal that the port 30080/TCP does not have the appropriate `http_port_t` SELinux context, which causes SELinux to prevent the `httpd` service from binding to the port. The log message also produces the syntax of the `semanage port` command, so that you can easily fix the issue.

```
[root@serverb ~]# sealert -a /var/log/audit/audit.log
...output omitted...
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port
30080.

***** Plugin bind_ports (92.2 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 30080
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 30080
    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...
```

- 7.4. Set the appropriate SELinux context on the port 30080/TCP for the `httpd` service to bind to it.

```
[root@serverb ~]# semanage port -a -t http_port_t -p tcp 30080
```

- 7.5. Restart the `httpd` service. This command should successfully restart the service.

```
[root@serverb ~]# systemctl restart httpd
```

- 7.6. Add the port 30080/TCP to the default public zone.

```
[root@serverb ~]# firewall-cmd --add-port=30080/tcp --permanent
success
[root@serverb ~]# firewall-cmd --reload
success
```

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

```
[root@serverb ~]# exit  
logout  
[student@serverb ~]$ exit  
logout  
Connection to serverb 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 rhcsa-comprevew3
```

## 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 rhcsa-comprevew3
```

This concludes the section.

## ► Lab

# Run Containers



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

## Outcomes

- Create rootless detached containers.
- Configure port mapping and persistent storage.
- Configure `systemd` for a container to manage it with `systemctl` commands.

## Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-comprevew4
```

## Specifications

- On `serverb`, configure the `podmgr` user with `redhat` as the password and set up the appropriate tools for the `podmgr` user to manage the containers for this comprehensive review. Configure the `registry.lab.example.com` as the remote registry. Use `admin` as the user and `redhat321` as the password to authenticate. You can use the `/tmp/review4/registry.conf` file to configure the registry.
- The `/tmp/review4/container-dev` directory contains two directories with development files for the containers in this comprehensive review. Copy the two directories under the `/tmp/review4/container-dev` directory to the `podmgr` home directory. Configure the `/home/podmgr/storage/database` subdirectory so that you can use it as persistent storage for a container.
- Create the production DNS-enabled container network. Use the `10.81.0.0/16` subnet and `10.81.0.1` as the gateway. Use this container network for the containers that you create in this comprehensive review.
- Create the `db-app01` detached container based on the `registry.lab.example.com/rhel8/mariadb-103` container image with the lowest tag number in the production

network. Use the /home/podmgr/storage/database directory as persistent storage for the /var/lib/mysql/data directory of the db-app01 container. Map the 13306 port on the local machine to the 3306 port in the container. Use the values of the following table to set the environment variables to create the containerized database.

Variable	Value
MYSQL_USER	developer
MYSQL_PASSWORD	redhat
MYSQL_DATABASE	inventory
MYSQL_ROOT_PASSWORD	redhat

- Create a `systemd` service file to manage the db-app01 container. Configure the `systemd` service so that when you start the service, the `systemd` daemon keeps the original container. Start and enable the container as a `systemd` service. Configure the db-app01 container to start at system boot.
- Copy the /home/podmgr/db-dev/inventory.sql script into the /tmp directory of the db-app01 container and execute it inside the container. If you executed the script locally, then you would use the `mysql -u root inventory < /tmp/inventory.sql` command.
- Use the container file in the /home/podmgr/http-dev directory to create the http-app01 detached container in the production network. The container image name must be `http-client` with the 9.0 tag. Map the 8080 port on the local machine to the 8080 port in the container.
- Use the `curl` command to query the content of the http-app01 container. Verify that the output of the command shows the container name of the client and that the status of the database is up.

## 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 rhcsa-comprevew4
```

## 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 rhcsa-comprevew4
```

This concludes the section.

## ► Solution

# Run Containers



### Note

If you plan to take the RHCSA exam, then use the following approach to maximize the benefit of this Comprehensive Review: attempt each lab without viewing the solution buttons or referring to the course content. Use the grading scripts to gauge your progress as you complete each lab.

### Outcomes

- Create rootless detached containers.
- Configure port mapping and persistent storage.
- Configure `systemd` for a container to manage it with `systemctl` commands.

### Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the last chapter, then save any work that you want to keep from earlier exercises on those machines, and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start rhcsa-compreview4
```

1. On `serverb`, configure the `podmgr` user with `redhat` as the password and set up the appropriate tools for the `podmgr` user to manage the containers for this comprehensive review. Configure the `registry.lab.example.com` as the remote registry. Use `admin` as the user and `redhat321` as the password to authenticate. You can use the `/tmp/review4/registry.conf` file to configure the registry.

- 1.1. Log in to `serverb` as the `student` user.

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

- 1.2. Install the `container-tools` meta-package.

```
[student@serverb ~]$ sudo dnf install container-tools
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 1.3. Create the podmgr user and set redhat as the password for the user.

```
[student@serverb ~]$ sudo useradd podmgr
[student@serverb ~]$ sudo passwd podmgr
Changing password for user podmgr.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 1.4. Exit the student user session. Log in to the serverb machine as the podmgr user. If prompted, use redhat as the password.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$ ssh podmgr@serverb
...output omitted...
[podmgr@serverb ~]$
```

- 1.5. Create the ~/.config/containers directory.

```
[podmgr@serverb ~]$ mkdir -p ~/.config/containers
```

- 1.6. Copy the /tmp/review4/registries.conf file to the container configuration directory in the home directory.

```
[podmgr@serverb ~]$ cp /tmp/review4/registries.conf ~/.config/containers/
```

- 1.7. Log in to the registry to verify the configuration.

```
[podmgr@serverb ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

2. The /tmp/review4/container-dev directory contains two directories with development files for the containers in this comprehensive review. Copy the two directories under the /tmp/review4/container-dev directory to the podmgr home directory. Configure the /home/podmgr/storage/database subdirectory so that you can use it as persistent storage for a container.

- 2.1. Copy the content of the /tmp/review4/container-dev directory to the podmgr home directory.

```
[podmgr@serverb ~]$ cp -r /tmp/review4/container-dev/* .
[podmgr@serverb ~]$ ls -l
total 0
drwxr-xr-x. 2 podmgr podmgr 27 May 10 21:52 db-dev
drwxr-xr-x. 2 podmgr podmgr 44 May 10 21:52 http-dev
```

- 2.2. Create the /home/podmgr/storage/database directory in the podmgr home directory. Set the appropriate permissions on the directory for the container to mount it as persistent storage.

```
[podmgr@serverb ~]$ mkdir -p storage/database
[podmgr@serverb ~]$ chmod 0777 storage/database
[podmgr@serverb ~]$ ls -l storage/
total 0
drwxrwxrwx. 2 podmgr podmgr 6 May 10 21:55 database
```

3. Create the production DNS-enabled container network. Use the 10.81.0.0/16 subnet and 10.81.0.1 as the gateway. Use this container network for the containers that you create in this comprehensive review.
- 3.1. Create the production DNS-enabled container network. Use the 10.81.0.0/16 subnet and 10.81.0.1 as the gateway.

```
[podmgr@serverb ~]$ podman network create --gateway 10.81.0.1 \
--subnet 10.81.0.0/16 production
production
```

- 3.2. Verify that the DNS feature is enabled in the production network.

```
[podmgr@serverb ~]$ podman network inspect production
[
  {
    "name": "production",
    ...output omitted...
    "subnets": [
      {
        "subnet": "10.81.0.0/16",
        "gateway": "10.81.0.1"
      }
    ],
    ...output omitted...
    "dns_enabled": true,
    ...output omitted...
  }
]
```

4. Create the db-app01 detached container based on the registry.lab.example.com/rhel8/mariadb-103 container image with the lowest tag number in the production network. Use the /home/podmgr/storage/database directory as persistent storage for the /var/lib/mysql/data directory of the db-app01 container. Map the 13306 port on the local machine to the 3306 port in the container. Use the values of the following table to set the environment variables to create the containerized database.

Variable	Value
MYSQL_USER	developer
MYSQL_PASSWORD	redhat
MYSQL_DATABASE	inventory
MYSQL_ROOT_PASSWORD	redhat

- 4.1. Search for the oldest version tag number of the `registry.lab.example.com/rhel8/mariadb` container image.

```
[podmgr@serverb ~]$ skopeo inspect \
docker://registry.lab.example.com/rhel8/mariadb-103
{
    "Name": "registry.lab.example.com/rhel8/mariadb-103",
    "Digest":
"sha256:a95b678e52bb9f4305cb696e45c91a38c19a7c2c5c360ba6c681b10717394816",
    "RepoTags": [
        "1-86",
        "1-102",
        "latest"
    ...
    .output omitted...
```

- 4.2. Use the oldest version tag number from the output of the previous step to create the detached `db-app01` container in the `production` network. Use the `/home/podmgr/storage/database` directory as persistent storage for the container. Map the 13306 port to the 3306 container port. Use the data that is provided in the table to set the environment variables for the container.

```
[podmgr@serverb ~]$ podman run -d --name db-app01 \
-e MYSQL_USER=developer \
-e MYSQL_PASSWORD=redhat \
-e MYSQL_DATABASE=inventory \
-e MYSQL_ROOT_PASSWORD=redhat \
--network production -p 13306:3306 \
-v /home/podmgr/storage/database:/var/lib/mysql/data:Z \
registry.lab.example.com/rhel8/mariadb-103:1-86
...
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID  IMAGE                                     COMMAND      CREATED
           STATUS          PORTS          NAMES
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86   run-mysqld  20
seconds ago   Up 20 seconds ago  0.0.0.0:13306->3306/tcp  db-app01
```

5. Create a `systemd` service file to manage the `db-app01` container. Configure the `systemd` service so that when you start the service, the `systemd` daemon keeps the original container. Start and enable the container as a `systemd` service. Configure the `db-app01` container to start at system boot.

- 5.1. Create the `~/.config/systemd/user/` directory for the container unit file.

```
[podmgr@serverb ~]$ mkdir -p ~/.config/systemd/user/
```

- 5.2. Create the systemd unit file for the db-app01 container, and move the unit file to the `~/.config/systemd/user/` directory.

```
[podmgr@serverb ~]$ podman generate systemd --name db-app01 --files
/home/podmgr/container-db-app01.service
[podmgr@serverb ~]$ mv container-db-app01.service ~/.config/systemd/user/
```

- 5.3. Stop the db-app01 container.

```
[podmgr@serverb ~]$ podman stop db-app01
db-app01
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED
      STATUS          PORTS NAMES
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld About
an hour ago   Exited (0) 3 seconds ago  0.0.0.0:13306->3306/tcp  db-app01
```

- 5.4. Reload the user systemd service to use the new service unit.

```
[podmgr@serverb ~]$ systemctl --user daemon-reload
```

- 5.5. Start and enable the systemd unit for the db-app01 container.

```
[podmgr@serverb ~]$ systemctl --user enable --now container-db-app01
Created symlink /home/podmgr/.config/systemd/user/default.target.wants/container-
db-app01.service → /home/podmgr/.config/systemd/user/container-db-app01.service.
[podmgr@serverb ~]$ systemctl --user status container-db-app01
● container-db-app01.service - Podman container-db-app01.service
   Loaded: loaded (/home/podmgr/.config/systemd/user/container-db-app01.service;
   disabled; vendor preset: disabled)
     Active: active (running) since Tue 2022-05-10 22:16:23 EDT; 7s ago
...output omitted...
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED
      STATUS          PORTS NAMES
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld 59
seconds ago   Up About a minute ago  0.0.0.0:13306->3306/tcp  db-app01
```

- 5.6. Use the `logindctl` command to configure the db-app01 container to start at system boot.

```
[podmgr@serverb ~]$ logindctl enable-linger
```

6. Copy the `/home/podmgr/db-dev/inventory.sql` script into the `/tmp` directory of the db-app01 container, and execute it inside the container. If you executed the script locally, then you would use the `mysql -u root inventory < /tmp/inventory.sql` command.

**Chapter 17 |** Comprehensive Review

- 6.1. Copy the /home/podmgr/db-dev/inventory.sql script into the /tmp directory of the db-app01 container.

```
[podmgr@serverb ~]$ podman cp /home/podmgr/db-dev/inventory.sql \
db-app01:/tmp/inventory.sql
```

- 6.2. Execute the inventory.sql script in the db-app01 container.

```
[podmgr@serverb ~]$ podman exec -it db-app01 sh -c 'mysql -u root inventory \
< /tmp/inventory.sql'
```

7. Use the container file in the /home/podmgr/http-dev directory to create the http-app01 detached container in the production network. The container image name must be http-client with the 9.0 tag. Map the 8080 port on the local machine to the 8080 port in the container.

- 7.1. Create the http-client:9.0 image with the container file in the /home/podmgr/http-dev directory.

```
[podmgr@serverb ~]$ podman build -t http-client:9.0 http-dev/
STEP 1/7: FROM registry.lab.example.com/rhel8/php-74:1-63
...output omitted...
```

- 7.2. Create the http-app01 detached container in the production network. Map the 8080 port from the local machine to the 8080 port in the container.

```
[podmgr@serverb ~]$ podman run -d --name http-app01 \
--network production -p 8080:8080 localhost/http-client:9.0
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID  IMAGE                                     COMMAND      CREATED
           STATUS          PORTS          NAMES
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld  20
              minutes ago   Up 20 seconds ago  0.0.0.0:13306->3306/tcp  db-app01
ee424df19621  localhost/http-client:9.0                  /bin/sh -c    4
              seconds ago   Up 4 seconds ago  0.0.0.0:8080->8080/tcp  http-app01
```

8. Query the content of the http-app01 container. Verify that it shows the container name of the client and that the status of the database is up.

- 8.1. Verify that the http-app01 container responds to http requests.

```
[podmgr@serverb ~]$ curl 127.0.0.1:8080
This is the server http-app01 and the database is up
```

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

```
[podmgr@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 rhcsa-comprevew4
```

## 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 rhcsa-comprevew4
```

This concludes the section.