



Red Hat Training and Certification

Student Workbook (ROLE)

Red Hat Enterprise Linux 9.3 RH124

Red Hat System Administration I

Edition 1



Red Hat
Learning Community

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



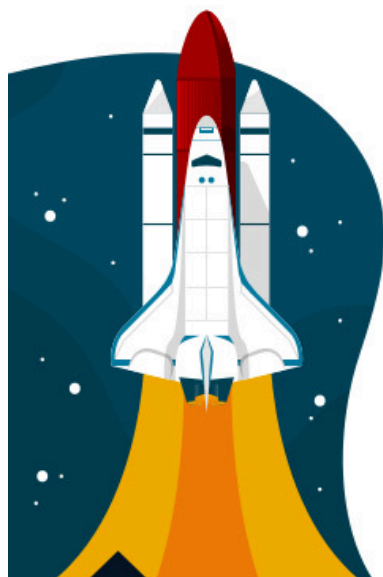
Engage in thousands of active conversations and posts



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



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.

Red Hat System Administration I

Red Hat Enterprise Linux 9.3 RH124

Red Hat System Administration I

Edition 1 20240717

Publication date 20240717

Authors: Ashish Lingayat, Alejandra Ramírez Palacios, Alex Callejas,
Antonio Marí Romero, Bernardo Gargallo, Dallas Spohn, Ed Parenti,
Jacob Pelchat, Mike Kelly, Morgan Weetman, Patrick Gomez,
Shreya Dchange

Course Architects: Philip Sweany, Steven Bonneville

DevOps Engineers: Artur Glogowski, Trey Feagle

Editors: Julian Cable, Rachel Lee

© 2024 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are © 2024 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

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

Document Conventions	xi
Admonitions	xi
Inclusive Language	xii
Introduction	xiii
Red Hat System Administration I	xiii
Orientation to the Classroom Environment	xiv
Performing Lab Exercises	xviii
1. Get Started with Red Hat Enterprise Linux	1
What Is Linux?	2
Quiz: Get Started with Red Hat Enterprise Linux	10
Summary	12
2. Access the Command Line	13
Access the Command Line	14
Quiz: Access the Command Line	19
Access the Command Line with the Desktop	23
Guided Exercise: Access the Command Line with the Desktop	28
Execute Commands with the Bash Shell	30
Quiz: Execute Commands with the Bash Shell	36
Lab: Access the Command Line	40
Summary	46
3. Manage Files from the Command Line	47
Describe Linux File System Hierarchy Concepts	48
Quiz: Describe Linux File System Hierarchy Concepts	50
Specify Files by Name	54
Quiz: Specify Files by Name	59
Manage Files with Command-line Tools	63
Guided Exercise: Manage Files with Command-line Tools	68
Make Links Between Files	72
Guided Exercise: Make Links Between Files	76
Match File Names with Shell Expansions	78
Quiz: Match File Names with Shell Expansions	83
Lab: Manage Files from the Command Line	87
Summary	97
4. Get Help in Red Hat Enterprise Linux	99
Read Manual Pages	100
Guided Exercise: Read Manual Pages	104
Lab: Get Help in Red Hat Enterprise Linux	108
Summary	115
5. Create, View, and Edit Text Files	117
Redirect Output to a File or Program	118
Quiz: Redirect Output to a File or Program	124
Edit Text Files from the Shell Prompt	128
Guided Exercise: Edit Text Files from the Shell Prompt	132
Change the Shell Environment	134
Guided Exercise: Change the Shell Environment	141
Lab: Create, View, and Edit Text Files	144
Summary	152
6. Manage Local Users and Groups	153
Describe User and Group Concepts	154
Quiz: Describe User and Group Concepts	158
Gain Superuser Access	162

Guided Exercise: Gain Superuser Access	167
Manage Local User Accounts	172
Guided Exercise: Manage Local User Accounts	175
Manage Local Group Accounts	178
Guided Exercise: Manage Local Group Accounts	181
Manage User Passwords	184
Guided Exercise: Manage User Passwords	188
Lab: Manage Local Users and Groups	192
Summary	198
7. Control Access to Files	199
Interpret Linux File System Permissions	200
Quiz: Interpret Linux File System Permissions	204
Manage File System Permissions from the Command Line	208
Guided Exercise: Manage File System Permissions from the Command Line	212
Manage Default Permissions and File Access	215
Guided Exercise: Manage Default Permissions and File Access	221
Lab: Control Access to Files	226
Summary	232
8. Monitor and Manage Linux Processes	233
Process States and Lifecycle	234
Quiz: Process States and Lifecycle	239
Control Jobs	241
Guided Exercise: Control Jobs	244
Kill Processes	249
Guided Exercise: Kill Processes	254
Monitor Process Activity	258
Guided Exercise: Monitor Process Activity	262
Lab: Monitor and Manage Linux Processes	267
Summary	278
9. Control Services and Daemons	279
Identify Automatically Started System Processes	280
Guided Exercise: Identify Automatically Started System Processes	286
Control System Services	290
Guided Exercise: Control System Services	294
Lab: Control Services and Daemons	298
Summary	302
10. Configure and Secure SSH	303
Access the Remote Command Line with SSH	304
Guided Exercise: Access the Remote Command Line	309
Configure SSH Key-based Authentication	313
Guided Exercise: Configure SSH Key-based Authentication	319
Customize OpenSSH Service Configuration	325
Guided Exercise: Customize OpenSSH Service Configuration	327
Lab: Configure and Secure SSH	333
Summary	340
11. Manage Networking	341
Describe Networking Concepts	342
Quiz: Describe Networking Concepts	355
Validate Network Configuration	359
Guided Exercise: Validate Network Configuration	365
Configure Networking from the Command Line	368
Guided Exercise: Configure Networking from the Command Line	375

Edit Network Configuration Files	380
Guided Exercise: Edit Network Configuration Files	384
Configure Hostnames and Name Resolution	388
Guided Exercise: Configure Hostnames and Name Resolution	392
Lab: Manage Networking	396
Summary	401
12. Install and Update Software Packages	403
Register Systems for Red Hat Support	404
Quiz: Register Systems for Red Hat Support	409
Explain and Investigate RPM Software Packages	411
Guided Exercise: Explain and Investigate RPM Software Packages	417
Install and Update Software Packages with DNF	421
Guided Exercise: Install and Update Software Packages with DNF	430
Enable DNF Software Repositories	435
Guided Exercise: Enable DNF Software Repositories	438
Lab: Install and Update Software Packages	441
Summary	447
13. Access Linux File Systems	449
Identify File Systems and Devices	450
Quiz: Identify File Systems and Devices	454
Mount and Unmount File Systems	456
Guided Exercise: Mount and Unmount File Systems	459
Locate Files on the System	462
Guided Exercise: Locate Files on the System	469
Lab: Access Linux File Systems	472
Summary	477
14. Analyze Servers and Get Support	479
Analyze and Manage Remote Servers	480
Guided Exercise: Analyze and Manage Remote Servers	492
Create a Diagnostics Report	496
Guided Exercise: Create a Diagnostics Report	501
Detect and Resolve Issues with Red Hat Insights	503
Quiz: Detect and Resolve Issues with Red Hat Insights	510
Summary	512
15. Comprehensive Review	513
Comprehensive Review	514
Lab: Manage Files from the Command Line	518
Lab: Manage Users and Groups, Permissions, and Processes	525
Lab: Configure and Manage a Server	532
Lab: Manage Networks	538
Lab: Mount File Systems and Find Files	544

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

These describe where to find external documentation that is relevant to a subject.



Note

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



Important

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



Warning

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

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

Red Hat System Administration I

Red Hat System Administration I (RH124) is designed for IT professionals without previous Linux system administration experience. The course provides students with Linux administration "survival skills" by focusing on core administration tasks. *Red Hat System Administration I* also provides a foundation for students who plan to become full-time Linux system administrators by introducing key command-line concepts and enterprise-level tools. These concepts are further developed in the follow-on course, *Red Hat System Administration II* (RH134).

Course Objectives

- Gain sufficient skill to perform core system administration tasks on Red Hat Enterprise Linux.
- Build foundational skills that an RHCSA-certified Red Hat Enterprise Linux system administrator needs.

Audience

- IT professionals across a broad range of disciplines who need to perform essential Linux administration tasks, including installation, establishing network connectivity, managing physical storage, and basic security administration.

Prerequisites

- This course has no formal prerequisites; however, previous system administration experience on other operating systems is beneficial.

Orientation to the Classroom Environment

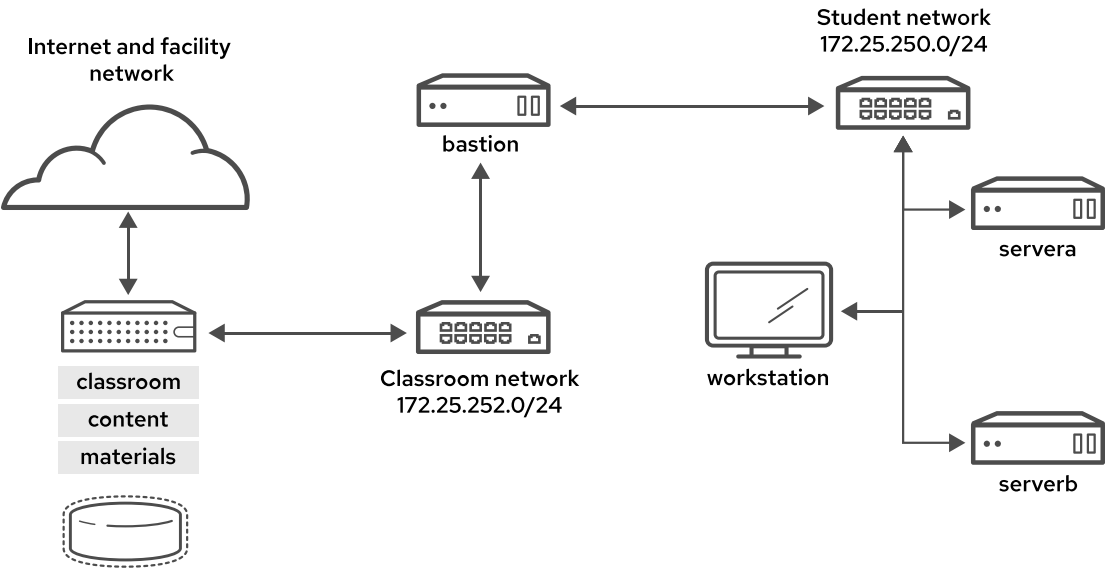


Figure 0.1: Classroom environment

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 address	Role
bastion.lab.example.com	172.25.250.254	Gateway system to connect the student private network to the classroom server (must always be running)
classroom.example.com	172.25.254.254	Server that hosts the required classroom materials
workstation.lab.example.com	172.25.250.9	Graphical workstation for student use
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 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 Your Systems

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>]. 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.

Table of ContentsCourseLab Environment★?

▶ Lab Controls

Click **CREATE** to build all of the virtual machines needed for the classroom lab environment. This may take several minutes to complete. Once created the environment can then be stopped and restarted to pause your experience.

If you **DELETE** your lab, you will remove all of the virtual machines in your classroom and lose all of your progress.

DELETESTOP ⓘ

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. All saved work on those systems' disks is lost.
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. All saved work on that system's disks is lost.

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

Get Started with Red Hat Enterprise Linux

Goal

Define open source, Linux, Linux distributions, and Red Hat Enterprise Linux.

Objectives

- Explain the purpose of open source, Linux, Linux distributions, and Red Hat Enterprise Linux.

Sections

- What Is Linux? (and Quiz)

What Is Linux?

Objectives

- Define and explain the purpose of Linux, open source, Linux distributions, and Red Hat Enterprise Linux.

Why Should You Learn about Linux?

Linux is a critical technology for IT professionals to understand.

Linux is in widespread use, worldwide. Internet users interact with Linux applications and web server systems daily, by browsing the World Wide Web and using e-commerce sites to buy and sell products.

Linux is in use for much more than the internet. Linux manages point-of-sale systems and the world's stock markets, powers smart TVs and in-flight entertainment systems, and runs most of the top 500 supercomputers in the world. Linux provides the core technologies that power the cloud revolution and the tools to build the latest generations of container-based microservices applications, software-based storage technologies, and big data solutions.

In the modern data center, Linux and Microsoft Windows are the predominant operating systems. Linux use continues to expand in enterprise, cloud, and device spaces. Due to its widespread adoption, you have many reasons to learn Linux:

- A Windows user needs to interoperate with Linux systems and applications.
- In application development, Linux commonly hosts the application and its runtime.
- In cloud computing, both private and public cloud instances use Linux as the operating system.
- Mobile applications and Internet of Things (IoT) devices commonly run on Linux.
- When looking for new IT career opportunities, Linux skills are in high demand.

What Makes Linux Great?

If someone asks you "What makes Linux great?", then you have many answers to pick from:

- Linux is *open source* software.

Being open source means that you can see all of how a program or system works. You can also experiment with changes and share them freely for others to use. The open source model means that improvements are easier to make, enabling faster innovation.

- Linux provides a *command-line interface* (CLI) for easy access and powerful scripting.

Linux is built around a basic design philosophy that users can perform all administration tasks from the CLI. It enables easier automation, deployment, and provisioning, and simplifies both local and remote system administration. Unlike many other operating systems, these capabilities were in the architecture from the start, and result in ease of use and stability.

- Linux is a *modular* operating system that is designed to easily replace or remove components.

System components can be upgraded and updated when needed. A Linux system can be a general-purpose development workstation or a purposefully minimized software appliance.

What Is Open Source Software?

Open source software is software with *source code* that anyone can use, study, modify, and share.

Source code is the set of human-readable instructions that are used to make a program. Code might be in interpretive form, such as a script, or compiled into a binary executable that the computer runs directly. Source code becomes copyrighted when created, and the copyright holder controls the terms under which the software can be copied, adapted, and distributed. Users can use the software according to its software license.

Some software uses "proprietary" or "closed source" source code that only the originating person, team, or organization can see, or change, or distribute. Proprietary licenses typically restrict the user to running the program, and provide limited or no access to the source.

Open source software is different. When a copyright holder provides software under an open source license, they grant the user the right to run the program and to view, modify, compile, and redistribute the source to others, royalty-free. Open source licensing promotes collaboration, sharing, transparency, and rapid innovation, because it encourages more people to modify and improve the software and to share enhancements more widely.

Open source software can still be provided for commercial use. Open source is a critical part of many organizations' commercial operations. Some open source licenses allow code to be reused in proprietary products. Anyone can sell open source code, but open source licensing generally allows the customer to redistribute the source code. Open source vendors such as Red Hat provide commercial support for deploying, managing, and building solutions that are based on open source products.

Open source has many benefits for the user:

- *Control*: See what the code does and improve it.
- *Training*: Learn from real-world code and develop more applications that are useful.
- *Security*: Inspect sensitive code, and fix it even without the original developers' help.
- *Stability*: Rely on code that can survive the loss of the original developer.

Types of Open Source Licenses

The developers of open source software can license their software in different ways. The software license terms control how the source can be combined with other code or reused. To be open source, licenses must allow users to freely use, view, change, compile, and distribute the code.

Two general classes of open source license are particularly important:

- *Copyleft* licenses are designed to encourage keeping the code open source.
- *Permissive* licenses are designed to maximize code reusability.

Copyleft, or "share-alike" licenses, require that anyone who distributes the source code, with or without changes, must pass along the freedom for others to also copy, change, and distribute the code. The advantage of copyleft licenses is that they help to keep existing code, and improvements to that code, open and increase the amount of available open source code. Common copyleft licenses include the *GNU General Public License* (GPL) and the *Lesser GNU Public License* (LGPL).

Permissive licenses maximize the reusability of source code. You can use the source for any purpose if the copyright and license statements are preserved, including reusing code under more restrictive or proprietary licenses. Although permissive licensing makes it easy to reuse code, it risks encouraging proprietary-only enhancements. Examples of permissive licenses include the MIT/X11 license, the Simplified BSD license, and the Apache Software License 2.0.

Who Develops Open Source Software?

Open source development today is overwhelmingly professional. Open source is no longer solely developed by a body of volunteers. Today, most open source developers work for organizations that pay them to participate with open source projects to construct and contribute the enhancements that the organization and their customers need.

Volunteers and the academic community still play a significant role and can make vital contributions, especially in emerging technology. The combination of formal and informal development provides a highly dynamic and productive environment.

What Is a Linux Distribution?

A *Linux distribution* is an installable operating system that is constructed from a Linux kernel and that supports user programs and libraries. A complete *Linux* system is developed by multiple independent development communities that work cooperatively on individual components. A distribution provides an easy method to install and manage a working Linux system.

In 1991, graduate student Linus Torvalds developed a UNIX-like kernel that he named Linux, and licensed it as open source software under the GPL. The kernel is the core of the operating system and manages hardware, memory, and the scheduling of running programs. The Linux kernel is supplemented with other open source software, including utilities and programs from the GNU Project, a graphical interface from MIT's *X Window System*. The Linux kernel also includes other open source components, such as the Sendmail mail server and the Apache HTTP web server, to become a complete open source UNIX-like operating system.

A major challenge for Linux users is to assemble all these software pieces from many sources. Early Linux developers provided a distribution of prebuilt and tested tools that users could download and install to quickly implement Linux systems.

Many Linux distributions exist, each with differing goals and support criteria. Generally, distributions have some common characteristics:

- Distributions consist of a Linux kernel and support user-space programs.
- Distributions can be small and single-purpose, or can include thousands of open source programs.
- Distributions provide a means to install and update the software and its components.
- The distribution provider supports the software, and ideally, participates in the development community.

Who Is Red Hat?

Red Hat is the world's leading provider of open source software solutions, by using a community-powered approach to reliable and high-performance cloud, Linux, middleware, storage, and virtualization technologies. The mission of Red Hat is to be the catalyst in communities of customers, contributors, and partners to create better technology the open source way.

The role of Red Hat is to help customers to connect with the open source community and their partners to effectively use open source software solutions. Red Hat actively participates in and supports the open source community. Many years of experience have convinced the company of the importance of open source to the future of the IT industry.

Red Hat is best known for its participation in the Linux community and the Red Hat Enterprise Linux distribution. Red Hat is also active in other open source communities, including middleware projects that are centered on the JBoss developer community. Red Hat also provides virtualization solutions, cloud technologies such as OpenStack and OpenShift, and the Ceph and Gluster software-based storage projects, plus others.

Red Hat Enterprise Linux Ecosystem

Red Hat Enterprise Linux (RHEL) is Red Hat's commercial production-grade Linux distribution. Red Hat develops and integrates open source software into RHEL through a multistage process.

- Red Hat *participates* in supporting individual open source projects. It contributes code, developer time, resources, and support, and often collaborates with developers from other Linux distributions, to improve the general quality of software for everyone.
- Red Hat sponsors and *integrates* open source projects into the community-driven Fedora distribution. Fedora provides a free working environment to serve as a development lab and proving ground for features to be incorporated into CentOS Stream and RHEL products.
- Red Hat *stabilizes* the CentOS Stream software to be ready for long-term support and standardization, and integrates it into RHEL, the production-ready distribution.

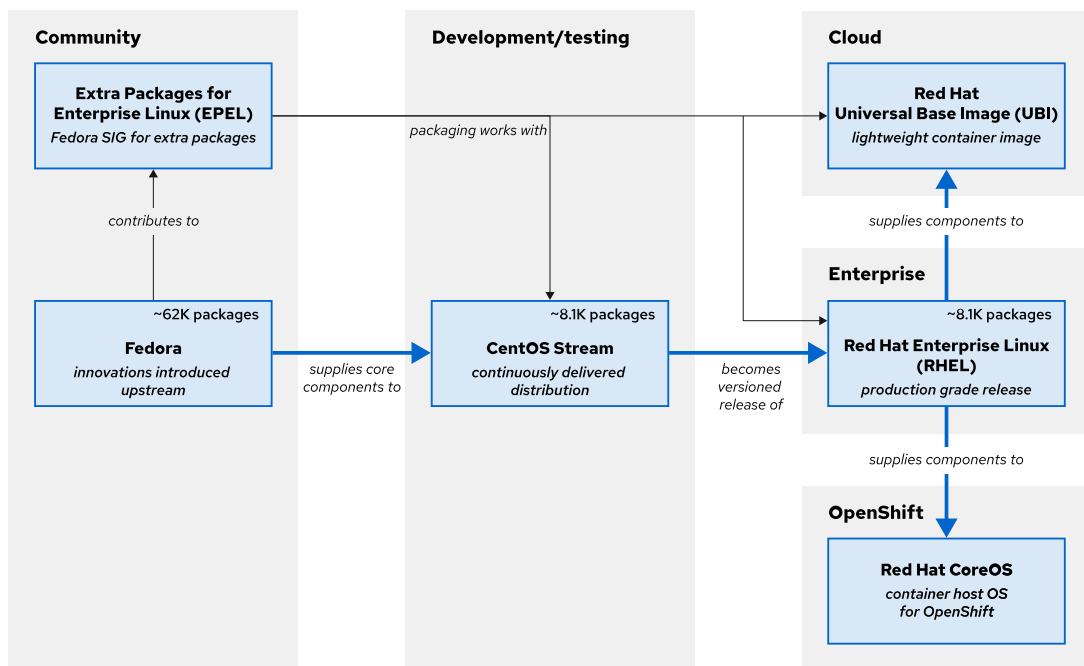


Figure 1.1: The Red Hat Enterprise Linux ecosystem

Fedora

Fedora is a community project that produces and releases a free, comprehensive Linux-based operating system. Red Hat sponsors and works with the Fedora community to integrate the latest upstream software into a fast-moving, secure distribution. The Fedora project contributes back to the open source world, and anyone can participate.

Fedora prioritizes innovation and excellence above long-term stability. Major updates occur every six months, and bring significant changes. Fedora supports releases for about a year, which means the latest two updates, making it less suited for supportable production use. Fedora remains the source of innovation for the entire Enterprise Linux ecosystem. In general, packages start out in Fedora and are included into CentOS Stream only when they are considered mature in stability, security, performance, and customer demand.

Extra Packages for Enterprise Linux

A Fedora project Special Interest Group (SIG) builds and maintains a community-supported package repository called Extra Packages for Enterprise Linux (EPEL). EPEL versions align with major RHEL releases, and enable RHEL customers to run workloads with software dependencies that are not supported in RHEL. EPEL packages are not included in Red Hat support, but are equivalent to Fedora's level of quality.

Typically, EPEL packages are built against RHEL releases. EPEL Next is an additional repository for package maintainers to build against CentOS Stream. This repository is useful when CentOS Stream contains an upcoming RHEL library rebase, or if an EPEL package has a minimum version build requirement that is already in CentOS Stream but not yet in RHEL.

CentOS Stream

CentOS Stream is the upstream project for RHEL. Development of the next RHEL version is transparent and open for community contributions that can directly influence the next release. Patches that are submitted to CentOS Stream are integrated faster to RHEL, to allow significant changes during the current RHEL version lifecycle. CentOS Stream is a continuous integration and delivery distribution, with tested and stable nightly builds.

The CentOS project welcomes contributors worldwide, to give RHEL derivatives the opportunity to contribute to CentOS Stream for their own benefit. The CentOS project also aims to promote sustainable open source software that responds faster to security exploits, emerging technologies, and changing customer requirements.



Note

Before 2019, CentOS Linux was a freely available, unsupported distribution, community-built from Red Hat's source code after each major RHEL release. Although the CentOS community enjoyed having a freely available RHEL clone, this model had disadvantages. Commonly, developer contributions to CentOS Linux were not backported to Fedora or RHEL without considerable duplicate effort. Also, significant delays occurred between a RHEL release and its corresponding CentOS distribution build, with a similar delay for critical RHEL security, driver, and tuning fixes. Red Hat switched to the CentOS Stream model to address these issues.

A benefit of CentOS Stream is that, as the source for RHEL development, it is available in all the same architectures as RHEL, including Intel/AMD x86_64, ARM64, IBM Power, and IBM Z.

Many innovative technology organizations have proven that CentOS Stream is a viable replacement for the original downstream CentOS Linux. CentOS Stream can be freely downloaded and installed for many use cases, including development and light production. For community users with use cases that are not suitable for a continuously delivered distribution with asynchronous patch releases, Red Hat provides free individual RHEL developer subscriptions for small-scale use, such as demos, prototyping, quality assurance, and limited production.

Red Hat Enterprise Linux

Red Hat Enterprise Linux (RHEL) is Red Hat's production-ready, commercially supported Linux distribution. In the computing industry, RHEL is acknowledged as the leading platform for open source computing. RHEL is extensively tested and has a worldwide ecosystem of support partners for hardware and software certifications, consulting services, training, and multi-year support and maintenance guarantees.

Red Hat builds RHEL major releases directly from the CentOS Stream continuous development project, which is sourced from Fedora. In contrast to the previous RHEL development model, the releases were constructed internally with less transparency, and the source was provided only for building as CentOS Linux after the RHEL release. Now the new CentOS Stream development model is open and available to all, for feedback and contribution, and the code is prepared to be the next major RHEL release.

RHEL uses a subscription-based support model, and does not charge license fees for open-source software. Red Hat support subscriptions provide product support, maintenance, updates, security patches, and access to the Customer Portal Knowledgebase, utilities, and downloadable releases of Red Hat products.

The following table lists some key differences between Fedora, CentOS Stream, and RHEL.

	Fedora	CentOS Stream	RHEL
Expected lifecycle	12-18 months	5 years	10 years
Software vendor certified	No	Usually not	Yes
Documentation provided by	Community	Community	Red Hat
Expert support available	No	No	Yes
Product security team	No	No	Yes
Security certifications	No	No	Yes
No-cost options	Yes	Yes	Yes
Management tools	No	No	Yes

RHEL for Edge

RHEL for Edge is an image-based variant of RHEL, with a different deployment mechanism. RHEL provides the ability to create purpose-built operating system images through a tool called Image Builder. With this mechanism, IT teams can build, deploy, and maintain these RHEL images in less time over the life of the system. Image-based deployments are optimized for various edge architectures, but are customizable for specific edge deployments.

The Edge features in RHEL include secure management and scaling capabilities, including zero-touch provisioning, system health visibility, and quick security remediations from within a single interface.

Red Hat CoreOS

RHEL CoreOS (RHCOS) is not a stand-alone operating system, but it is built from RHEL components, and is then released, upgraded, and managed as part of the Red Hat OpenShift Container Platform (RHOCP) for cloud-native applications. RHCOS is fundamentally an image-based RHEL container host, which uses the Container Runtime Interface (CRI-O)-compliant container engine that is integrated in RHOCP. To learn more about Red Hat CoreOS, begin by becoming familiar with OpenShift and containers.

Red Hat Universal Base Image

A Red Hat Universal Base Image (UBI) is essentially a freely redistributable derivative of RHEL. UBI is designed to be a foundation for cloud-native and web application use cases that are developed in containers. All UBI content is a subset of RHEL, with packages sourced from secure RHEL channels, and UBI is supported similar to RHEL when run on a Red Hat supported platforms such as OpenShift and RHEL hosts.

With UBI, developers can focus their efforts on their application in the container image. UBI is a set of base images, and a set of application images (such as python, ruby, node.js, httpd, or nginx). UBI also consists of a set of RPM repositories, from which you can update any UBI base image to include the package dependencies that your application requires.

Red Hat Enterprise Linux Continuous Development

In the Fedora upstream community, Fedora Rawhide is the continuous development environment for a regular cadence of public Fedora releases. The community tests and prepares new Linux kernel versions, device drivers, utilities, and applications for the next Fedora distribution. Major RHEL release development begins with selection of the latest Fedora release as the base for the current CentOS Stream continuous development distribution.

Before a package is formally introduced to CentOS Stream, it undergoes rigorous testing to meet the standards for packages to be included in RHEL. Updates that are posted to CentOS Stream are the same as to those updates that are posted to the unreleased minor version of RHEL in development.

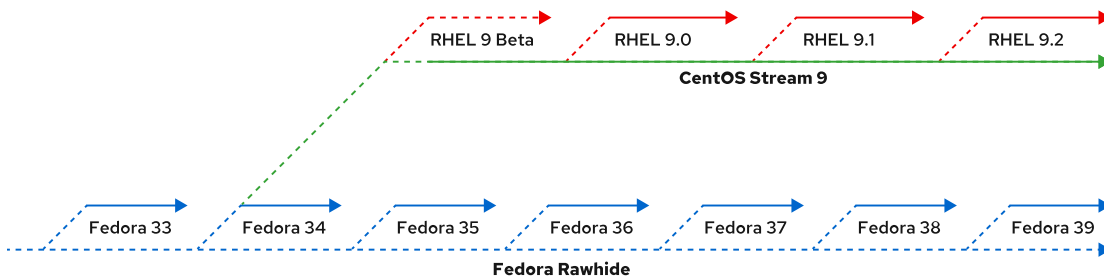


Figure 1.2: Red Hat Enterprise Linux continuous development

As shown in the figure, Fedora 34 is the original code base for RHEL 9 and for CentOS Stream 9. As packages are updated, they are then pushed into CentOS Stream and the nightly build of RHEL. The solid lines indicate distributions or builds that are available for public use.

Similar to the relationship between Fedora Rawhide and Fedora, CentOS Stream is the continuous development environment for preparing the next minor-version RHEL release. Red Hat performs extensive hardware, integration, dependency, and performance testing before releasing the next public RHEL distribution.

Obtaining Red Hat Enterprise Linux

Red Hat Enterprise Linux is typically obtained with a paid support subscription, and Red Hat provides multiple ways to obtain RHEL and other RHEL ecosystem products, many without cost.

- *Fedora Linux* and derivatives are freely available from the Fedora project at <https://getfedora.org/>, including an emerging version of Fedora CoreOS.
- *EPEL* and *EPEL Next* packages are freely available from the EPEL project repositories. Learn how to use EPEL at <https://docs.fedoraproject.org/en-US/epel/>.
- *CentOS Stream* is freely available at <https://www.centos.org/centos-stream/>.

RHEL Evaluation Download

An evaluation copy of RHEL is available at <https://access.redhat.com/products/red-hat-enterprise-linux/evaluation>. You must have a (free) Customer Portal account for <https://access.redhat.com> to access and download evaluation products. Product evaluations entitle you to receive updates and support for a limited period. Support ends when the evaluation period ends, but the evaluation software continues to operate. Additional information for many product evaluations is found on the Customer Portal evaluation pages.

Red Hat Developer Subscription

Red Hat provides a freely available subscription for many products through the Red Hat Developer Program at <https://developer.redhat.com>. With a Developer subscription, developers can quickly create, prototype, test, and demonstrate their applications on the same Red Hat software as on production systems. Create a personal account at <https://access.redhat.com>, and then register for the Developer program. You can use an existing personal account, but do not use an account that is already associated with any organization's support subscription. The Developer subscription is self-supported, but provides ongoing product updates. Red Hat recommends that individuals who want to gain experience with RHEL and developer products should join the Developer Program.

Public Cloud Platforms

The major hyperscale public cloud providers, such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure, offer official images for deploying Red Hat Enterprise Linux instances, with subscription management from the Red Hat Cloud Access service. Fully entitled subscriptions for RHEL and Red Hat products are available through the cloud provider, and are portable in hybrid and multi-vendor clouds.

Containers

You can use Red Hat Universal Base Images and associated content for development and deployment without a Red Hat subscription. For operational support and access to non-UBI tools, containers that are built on UBI must be deployed on a Red Hat-supported platform such as OpenShift or Red Hat Enterprise Linux. Access to non-UBI content requires a Red Hat subscription.



References

Get Started with Red Hat Enterprise Linux

<https://access.redhat.com/products/red-hat-enterprise-linux#getstarted>

No-cost Red Hat Enterprise Linux Individual Developer Subscription: FAQs

<https://developers.redhat.com/articles/faqs-no-cost-red-hat-enterprise-linux#>

The Open Source Way

<https://opensource.com/open-source-way>

Fedora

<https://getfedora.org/>

Red Hat Universal Base Images

<https://developers.redhat.com/products/rhel/ubi>

Red Hat Cloud Access

<https://www.redhat.com/en/technologies/cloud-computing/cloud-access>

► Quiz

Get Started with Red Hat Enterprise Linux

Choose the correct answers to the following questions:

- 1. **Which two statements are benefits of open source software for the user? (Choose two.)**
 - a. Code can survive the loss of the original developer or distributor.
 - b. Sensitive portions of code are protected and available only to the original developer.
 - c. You can learn from real-world code and develop more effective applications.
 - d. Code remains open provided that it is in a public repository, but the license might change when included with closed source software.
- 2. **Which two statements are ways in which Red Hat develops products for the future and interacts with the community? (Choose two.)**
 - a. Sponsor and integrate open source projects into the community-driven Fedora project.
 - b. Develop specific integration tools that are available only in Red Hat distributions.
 - c. Participate in upstream projects.
 - d. Repackage and relicense community products.
- 3. **Which two statements describe the benefits of Linux? (Choose two.)**
 - a. Linux is developed entirely by volunteers, which makes it a low-cost operating system.
 - b. Linux is modular and can be configured for a full graphical desktop or a small appliance.
 - c. Linux is locked in a known state for a minimum of one year for each release, so it is easier to develop custom software.
 - d. Linux includes a powerful and scriptable command-line interface, which enables easier automation and provisioning.

► Solution

Get Started with Red Hat Enterprise Linux

Choose the correct answers to the following questions:

- 1. Which two statements are benefits of open source software for the user? (Choose two.)
 - a. Code can survive the loss of the original developer or distributor.
 - b. Sensitive portions of code are protected and available only to the original developer.
 - c. You can learn from real-world code and develop more effective applications.
 - d. Code remains open provided that it is in a public repository, but the license might change when included with closed source software.
- 2. Which two statements are ways in which Red Hat develops products for the future and interacts with the community? (Choose two.)
 - a. Sponsor and integrate open source projects into the community-driven Fedora project.
 - b. Develop specific integration tools that are available only in Red Hat distributions.
 - c. Participate in upstream projects.
 - d. Repackage and relicense community products.
- 3. Which two statements describe the benefits of Linux? (Choose two.)
 - a. Linux is developed entirely by volunteers, which makes it a low-cost operating system.
 - b. Linux is modular and can be configured for a full graphical desktop or a small appliance.
 - c. Linux is locked in a known state for a minimum of one year for each release, so it is easier to develop custom software.
 - d. Linux includes a powerful and scriptable command-line interface, which enables easier automation and provisioning.

Summary

- Open source software has source code that anyone can freely use, study, modify, and share.
- A Linux distribution is an installable operating system that is constructed from a Linux kernel and that supports user programs and libraries.
- Red Hat participates in supporting and contributing code to open source projects; sponsors and integrates project software into community-driven distributions; and stabilizes the software to offer it as supported enterprise-ready products.
- Red Hat Enterprise Linux is the open source, enterprise-ready, commercially supported Linux distribution that Red Hat provides.
- A free Red Hat Developer Subscription is a useful method for obtaining learning resources and information, including developer subscriptions to Red Hat Enterprise Linux and other Red Hat products.

Chapter 2

Access the Command Line

Goal

Log in to a Linux system and run simple commands from the shell.

Objectives

- Log in to a Linux system and run simple commands from the shell.
- Log in to the Linux system with the GNOME desktop environment to run commands from a shell prompt in a terminal program.
- Save time when running commands from a shell prompt with Bash shortcuts.

Sections

- Access the Command Line (and Quiz)
- Access the Command Line with the Desktop (and Guided Exercise)
- Execute Commands with the Bash Shell (and Quiz)

Lab

- Access the Command Line

Access the Command Line

Objectives

- Log in to a Linux system and run simple commands with the shell.

Introduction to the Bash Shell

A *command line* is a text-based interface that is used to input instructions to a computer system. The Linux command line is provided by a program called the *shell*. Many shell program variants have been developed over the years. Every user can use a different shell, but Red Hat recommends using the default shell for system administration.

The default user shell in Red Hat Enterprise Linux (RHEL) is the GNU Bourne-Again Shell (**bash**). The **bash** shell is an improved version of the original Bourne Shell (**sh**) on UNIX systems.

The shell displays a string when it is waiting for user input, called the *shell prompt*. When a regular user starts a shell, the prompt includes an ending dollar (\$) character:

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

A hash (#) character replaces the dollar (\$) character when the shell is running as the superuser, **root**. This character indicates that it is a superuser shell, which helps to avoid mistakes that can affect the whole system.

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

Using **bash** to execute commands can be powerful. The **bash** shell provides a scripting language that can support task automation. The shell has capabilities that can enable or simplify operations that are hard to accomplish at scale with graphical tools.



Note

The **bash** shell is conceptually similar to the Microsoft Windows `cmd.exe` command-line interpreter. However, **bash** has a sophisticated scripting language, and is more similar to Windows PowerShell.

On macOS, the **bash** shell was the default shell before macOS 10.15 Catalina. Starting from macOS 10.15 Catalina, Apple changed the default shell to the **zsh** shell, an alternative shell that is also available in RHEL.

Shell Basics

Commands that are entered at the shell prompt have three basic parts:

- *Command* to run.
- *Options* to adjust the behavior of the command.
- *Arguments*, which are typically targets of the command.

The command is the name of the program to run. It might be followed by one or more options, which adjust the behavior of the command or what it does. Options normally start with one or two dashes (`-a` or `--all`, for example) to distinguish them from arguments. Commands might also be followed by one or more arguments, which often indicate a target that the command should operate on.

For example, in the `usermod -L user01` string, `usermod` is the command, `-L` is the option, and `user01` is the argument. This command locks the password of the `user01` user account.

Log in to a Local System

A *terminal* is a text-based interface to enter commands into and print output from a computer system. To run the shell, you must log in to the computer on a terminal.

A hardware keyboard and display for input and output might be directly connected to the computer. This is the *physical console* from the Linux machine. The physical console supports multiple *virtual consoles*, which can run on separate terminals. Each virtual console supports an independent login session. You can switch between the virtual consoles by pressing `Ctrl+Alt` and a function key (`F1` through `F6`) at the same time. Most of these virtual consoles run a terminal that provides a text login prompt. If you enter your username and password correctly, then you log in and get a shell prompt.

The computer might provide a graphical login prompt on one of the virtual consoles. You can use the graphical login prompt to log in to a *graphical environment*. The graphical environment also runs on a virtual console. To get a shell prompt, you must start a terminal program in the graphical environment. The shell prompt is provided in an application window of your graphical terminal program.



Note

Many system administrators choose not to run a graphical environment on their servers, because users do not log in to servers as a desktop workspace. A server's workload can more effectively use the significant resources that a graphical environment uses.

In Red Hat Enterprise Linux 9, if the graphical environment is available, then the login screen runs on the first virtual console, which is called `tty1`. Five additional text login prompts are available on virtual consoles two (`tty2`) through six (`tty6`).

The graphical environment starts on the first virtual console that a login session is not currently using. Normally, your graphical session replaces the login prompt on the second virtual console (`tty2`). However, if an active text login session (not just a login prompt) is using that console, then the next free virtual console is used instead.

The graphical login screen continues to run on the first virtual console (`tty1`). If you are already logged in to a graphical session, and switch to another user in the graphical environment without logging out, then another graphical environment is started for that user on the next available virtual console.

When you log out of a graphical environment, it exits the virtual console, and the physical console automatically switches back to the graphical login screen on the first virtual console.

**Note**

In Red Hat Enterprise Linux 6 and 7, the graphical login screen runs on the first virtual console, but when you log in, your initial graphical environment *replaces* the login screen on the first virtual console instead of starting on a new virtual console. In Red Hat Enterprise Linux 8, the behavior is the same as in Red Hat Enterprise Linux 9.

A *headless server* does not have a keyboard and display that are permanently connected to it. A data center might be filled with many racks of headless servers, and not providing each with a keyboard and display saves space and expense. For administrators to log in, a login prompt for a headless server might be provided by its *serial console*, which runs on a serial port that is connected to a networked console server for remote access.

The serial console is normally used to access the server if the server network card becomes misconfigured and logging to the server over the conventional network connection becomes impossible. Most of the time, however, headless servers are accessed by other means over the network, for example by using Virtual Network Computing (VNC) for running a graphical interface on the target machine.

Log in to a Remote System

Linux users and administrators often need to get shell access to a remote system by connecting to it over the network. In a modern computing environment, many headless servers are virtual machines or are running as public or private cloud instances. These systems are not physical and do not have real hardware consoles. They might not even provide access to their (simulated) physical console or serial console.

In Linux, the most common way to get a shell prompt on a remote system is to use Secure Shell (SSH). Most Linux systems (including Red Hat Enterprise Linux) and macOS provide the **OpenSSH** command-line program `ssh` for this purpose.

In this example, a user with a shell prompt on the `host` machine uses `ssh` to log in to the remote Linux system `remotehost` as the user `remoteuser`:

```
[user@host ~]$ ssh remoteuser@remotehost
remoteuser@remotehost's password: password
[remoteuser@remotehost ~]$
```

The `ssh` command encrypts the connection to secure the communication against eavesdropping or hijacking of the passwords and content.

Some systems, such as new cloud instances, for tighter security do not allow users to use a password to log in with `ssh`. An alternative way to authenticate to a remote machine without entering a password is through *public key authentication*.

With this authentication method, users have a special identity file with a *private key*, which is equivalent to a password, and which they keep secret. Their account on the server is configured with a matching *public key*, which does not have to be secret. When logging in, users can configure `ssh` to provide the private key. If their matching public key is installed in that account on that remote server, then it logs in the user without asking for a password.

In the next example, a user with a shell prompt on the `host` machine logs in to `remotehost` as `remoteuser` with `ssh`, by using the public key authentication method. The `ssh` command `-i`

option is used to specify the user's private key file, which is `mylab.pem`. The matching public key is already set up as an authorized key in the `remoteuser` account.

```
[user@host ~]$ ssh -i mylab.pem remoteuser@remotehost
[remoteuser@remotehost ~]$
```

For the connection to work, only the user who owns the file can have access to read the private key file. In the preceding example, where the private key is in the `mylab.pem` file, you can use the `chmod 600 mylab.pem` command to ensure that only the owner can read the file. How to set file permissions is discussed in more detail in a later chapter.

Users might also have configured private keys that are tried automatically, but that discussion is beyond the scope of this section. The References at the end of this section contain links to more information about this topic.

**Note**

When you first log in to a new machine, you are prompted with a warning from `ssh` that it cannot establish the authenticity of the host:

```
[user@host ~]$ ssh -i mylab.pem remoteuser@remotehost
The authenticity of host 'remotehost (192.0.2.42)' can't be established.
ECDSA key fingerprint is 47:bf:82:cd:fa:68:06:ee:d8:83:03:1a:bb:29:14:a3.
Are you sure you want to continue connecting (yes/no)? yes
[remoteuser@remotehost ~]$
```

Each time that you connect to a remote host with `ssh`, the remote host sends its *host key* to authenticate itself and to help to set up encrypted communication. The `ssh` command compares the host key against a list of saved host keys to ensure that it is not changed. If the host key changed, then it might indicate that someone is trying to pretend to be that host to hijack the connection, which is also known as an interceptor attack. In SSH, host keys protect against interceptor attacks; these host keys are unique for each server; and they need to be changed periodically and whenever a compromise is suspected.

You get this warning when your local machine does not have a saved host key for the remote host. If you enter `yes`, then the host key that the remote host sent is accepted and saved for future reference. The login process continues, and you should not see this message again when connecting to this host. If you enter `no`, then the host key is rejected and the connection is closed.

If the local machine does have a saved host key and it does not match the one that the remote host sent, then the connection is closed automatically with a warning.

Log Out from a Remote System

When you are finished with the shell and want to quit, you can choose one of several ways to end the session. You can enter the `exit` command to terminate the current shell session. Alternatively, finish a session by pressing `Ctrl+D`.

The following example shows a user who logs out of an SSH session:


```
[remoteuser@remotehost ~]$ exit  
logout  
Connection to remotehost closed.  
[user@host ~]$
```



References

intro(1), bash(1), pts(4), ssh(1), and ssh-keygen(1) man pages

For more information about OpenSSH and public key authentication, refer to the *Using Secure Communications between Two Systems with OpenSSH* chapter in the *Red Hat Enterprise Linux 9 Securing Networks* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/securing_networks/index

Instructions about how to read man pages and other online help documentation are included in the upcoming chapter.

► Quiz

Access the Command Line

Choose the correct answer to the following questions:

- 1. Which term describes the interpreter that executes commands that are typed as strings?
 - a. Command
 - b. Console
 - c. Shell
 - d. Terminal
- 2. Which term describes the visual cue that indicates that an interactive shell is waiting for the user to type a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- 3. Which term describes the name of a program to run?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- 4. Which term describes the part of the command line that adjusts the behavior of a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- 5. Which term describes the part of the command line that specifies the target that the command should operate on?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt

- ▶ 6. Which term describes the hardware display and keyboard to interact with a system?
 - a. Physical Console
 - b. Virtual Console
 - c. Shell
 - d. Terminal

- ▶ 7. Which term describes one of multiple logical consoles that can each support an independent login session?
 - a. Physical Console
 - b. Virtual Console
 - c. Shell
 - d. Terminal

- ▶ 8. Which term describes an interface that provides a display for output and a keyboard for input to a shell session?
 - a. Console
 - b. Virtual Console
 - c. Shell
 - d. Terminal

► Solution

Access the Command Line

Choose the correct answer to the following questions:

- 1. Which term describes the interpreter that executes commands that are typed as strings?
 - a. Command
 - b. Console
 - c. Shell
 - d. Terminal
- 2. Which term describes the visual cue that indicates that an interactive shell is waiting for the user to type a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- 3. Which term describes the name of a program to run?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- 4. Which term describes the part of the command line that adjusts the behavior of a command?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt
- 5. Which term describes the part of the command line that specifies the target that the command should operate on?
 - a. Argument
 - b. Command
 - c. Option
 - d. Prompt

- ▶ **6. Which term describes the hardware display and keyboard to interact with a system?**
 - a. Physical Console
 - b. Virtual Console
 - c. Shell
 - d. Terminal

- ▶ **7. Which term describes one of multiple logical consoles that can each support an independent login session?**
 - a. Physical Console
 - b. Virtual Console
 - c. Shell
 - d. Terminal

- ▶ **8. Which term describes an interface that provides a display for output and a keyboard for input to a shell session?**
 - a. Console
 - b. Virtual Console
 - c. Shell
 - d. Terminal

Access the Command Line with the Desktop

Objectives

- Log in to the Linux system with the GNOME desktop environment to run commands from a shell prompt in a terminal program.

Introduction to the GNOME Desktop Environment

The *desktop environment* is the graphical user interface on a Linux system. GNOME 40 is the default desktop environment in Red Hat Enterprise Linux 9. It provides an integrated desktop for users and a unified development platform on top of a graphical framework that either Wayland (by default) or the legacy X Window System provides.

GNOME Shell provides the core user interface functions for the GNOME desktop environment. The GNOME Shell application is highly customizable. Red Hat Enterprise Linux 9 defaults the GNOME Shell appearance to the "Standard" theme, which is used in this section. You can default to an alternative "Classic" theme, which is closer to the appearance of earlier versions of GNOME, and which is used on previous RHEL versions. You can select either theme persistently at login by clicking the gear icon next to the **Sign In** button. The gear icon is available after selecting your account but before entering your password.

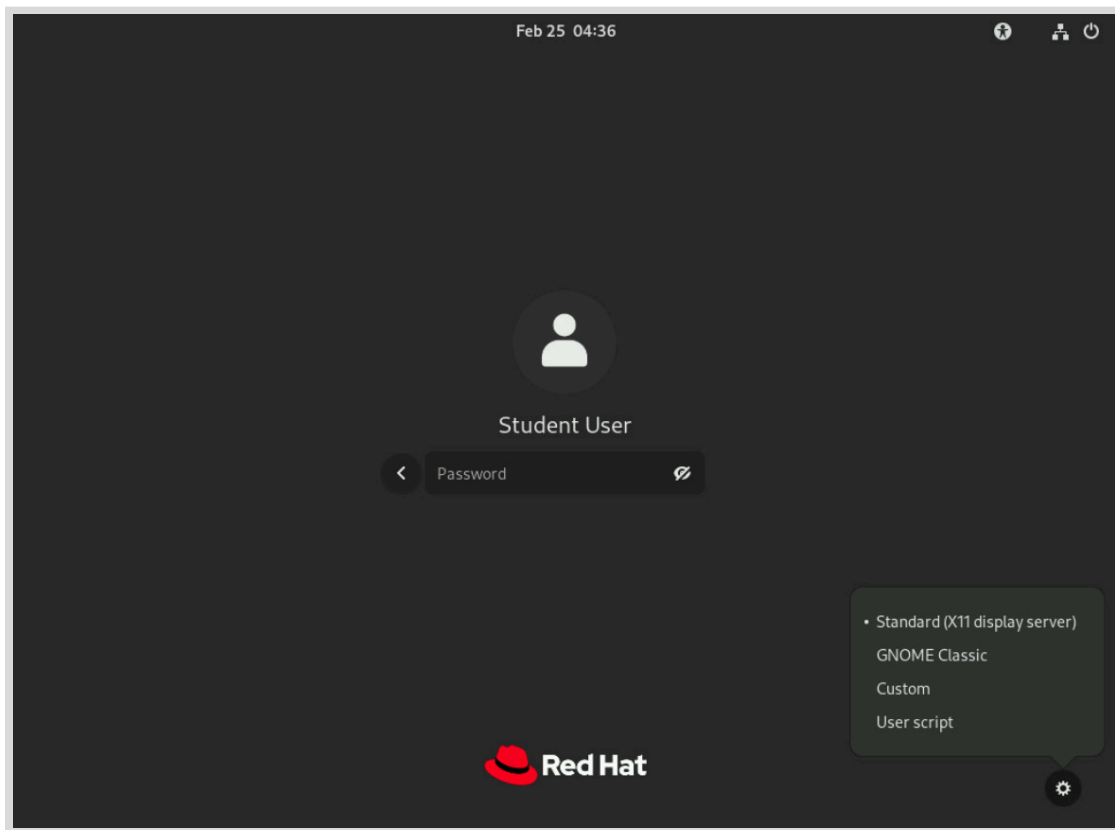


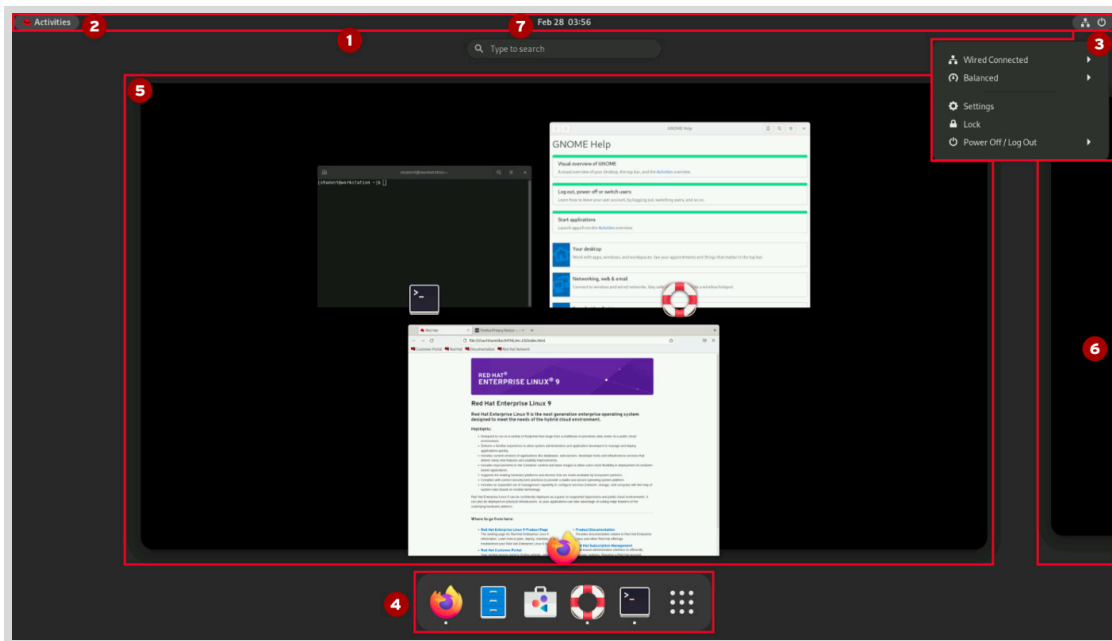
Figure 2.1: The RHEL 9 login screen

The first time that you log in as a new user, you can take an optional initial "Take Tour" program to learn about the new RHEL 9 features. After you either complete it or skip it, the main GNOME screen appears.

To review the documentation in **GNOME Help**, click the **Activities** button on the left side of the top bar. In the dash at the bottom of the screen, click the ring icon to launch it or hover your mouse over the **Help** icon.

Parts of the GNOME Shell

The elements of the GNOME Shell include the following parts, as shown in this screen capture of the GNOME Shell in Activities overview mode:



- 1 **Top bar:** The bar that runs along the top of the screen. It is displayed in the Activities overview and in workspaces. The top bar provides the **Activities** button and controls for volume, networking, calendar access, and switching between keyboard input methods (if more than one method is configured).
- 2 **Activities overview:** This mode helps to organize windows and to start applications. Enter the Activities overview by clicking the **Activities** button at the upper-left corner of the top bar, or by pressing the Super key. Find the Super key (sometimes called the **Windows** key or **Command** key) near the lower-left corner of most common keyboards. The three main areas are the **dash** at the bottom of the screen, the **windows overview** in the center, and the **workspace selector** on the right side.
- 3 **System menu:** The menu in the upper-right corner on the top bar provides control to adjust the brightness of the screen, and to switch on or off the network connections. Under the submenu for the user's name are options to adjust account settings and to log out of the system. The system menu also offers buttons to open the **Settings** window, lock the screen, or shut down the system.
- 4 **Dash:** This configurable list of icons shows your favorite applications, running applications, and a **Show Applications** button to select arbitrary applications. Start applications by clicking an icon or by using the **Show Applications** button to find less commonly used applications. The dash is also called the **dock**.

- 5 **Windows overview:** The area in the center of the Activities overview that displays thumbnails of active windows in the current workspace, for bringing windows to the foreground on a cluttered workspace, or moving them to another workspace.
- 6 **Workspace selector:** An area to the right, which displays thumbnails of active workspaces, and allows selecting workspaces and moving windows from one workspace to another.
- 7 **Message tray:** With the message tray, you can review notifications from applications or system components. If a notification occurs, the notification typically first appears briefly as a single line at the top of the screen, and a persistent indicator appears in the top bar next to the clock to inform you of recently received notifications. Open the message tray to review these notifications by clicking the clock on the top bar or by pressing **Super+M**. Close the message tray by clicking the clock on the top bar, or by pressing **Esc** or **Super+M** again. The message tray also shows the calendar and information about the events in the calendar.

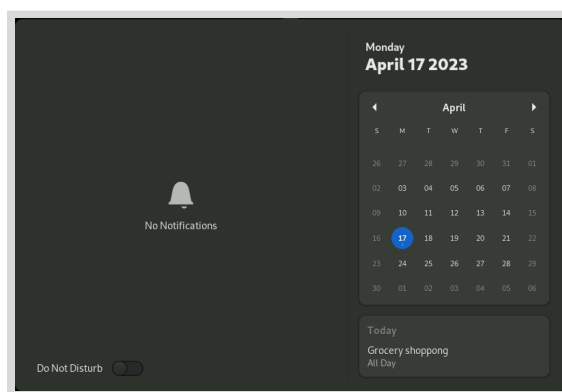


Figure 2.2: Closeup of an open message tray

View and edit the GNOME keyboard shortcuts that your account uses. Open the system menu on the right side of the top bar. Click the **Settings** button on the bottom of the menu on the left. In the application window that opens, select **Keyboard** from the left pane. The right pane displays your current shortcut settings under the **Keyboard Shortcuts > Customize Shortcuts** section.



Note

Some keyboard shortcuts, such as function keys or the **Super** key, might be difficult to send to a virtual machine. Special keystrokes that those shortcuts use might be captured by your local operating system, or by the application that you are using to access the graphical desktop of your virtual machine.

**Important**

In the current virtual training and self-paced training environments provided by Red Hat, use of the **Super** key can be tricky, because your web browser might not pass it to the virtual machine in the classroom environment.

At the top of your browser window that displays the interface for your virtual machine, click the keyboard icon on the right side. An on-screen keyboard opens. Click the icon again to close the on-screen keyboard.

The on-screen keyboard treats **Super** as a modifier key that is often held down while pressing another key. If you click it once, then it turns yellow, to indicate that the key is being held down. So for example, to enter **Super + M** in the on-screen keyboard, you can click **Super** and then click **M**.

To press and release **Super** in the on-screen keyboard, then click it twice. The first click holds down the **Super** key, and the second click releases it.

The other keys that the on-screen keyboard treats as modifier keys (like **Super**) are **Shift**, **Ctrl**, **Alt**, and **Caps**. The **Esc** and **Menu** keys are treated like normal keys and *not* modifier keys.

Access Workspaces

Workspaces are separate desktop screens that have different application windows. You can use workspaces to organize the working environment by grouping open application windows by task. For example, you can group windows for a particular system maintenance activity (such as setting up a new remote server) in one workspace, and you can group email and other communication applications in another workspace.

Choose between two methods to switch between workspaces. The first method is to press **Ctrl+Alt+LeftArrow** or **Ctrl+Alt+RightArrow** to switch between workspaces sequentially. The second is to switch to the **Activities** overview and click the chosen workspace.

An advantage of using the **Activities** overview is that you can click and drag windows between workspaces by using the **workspace selector** on the right side of the screen and the **windows overview** in the center of the screen.

**Important**

Like **Super**, in the current virtual training and self-paced training environments provided by Red Hat, your web browser does not usually pass **Ctrl+Alt** key combinations to the virtual machine in the classroom environment.

You can enter these key combinations to switch workspaces by using the on-screen keyboard. At least two workspaces must be in use. Open the on-screen keyboard and click **Ctrl**, **Alt**, and then either **LeftArrow** or **RightArrow**.

However, in those training environments, it is generally simpler to avoid the keyboard shortcuts and the on-screen keyboard. Switch workspaces by clicking the **Activities** button and then, in the workspace selector to the right of the **Activities** overview, clicking the workspace to switch to.

Start a Terminal

To get a shell prompt in GNOME, start a graphical terminal application such as **GNOME Terminal**. Use one of the following methods to start a terminal:

- From the **Activities** overview, select **Terminal** from the **dash**, either in Favorites or with the **Show Applications** button.
- Search for **terminal** in the search field at the top of the **windows overview**.
- Press the **Alt+F2** key combination to open the **Enter a Command** and enter **gnome-terminal**.

When you open a terminal window, a shell prompt is displayed for the user who started the graphical terminal program. The shell prompt and the terminal window's title bar indicate the current username, hostname, and working directory.

Lock the Screen and Log Out

Lock the screen, or log out entirely, from the system menu on the far right of the top bar.

To lock the screen, from the system menu in the upper-right corner, click the lock button at the bottom of the menu or press **Super+L** (which might be easier to remember as **Windows+L**). The screen also locks if the graphical session is idle for a few minutes.

A **lock screen curtain** appears that shows the system time and the name of the logged-in user. To unlock the screen, you can press **Enter**, **Space**, or click the left mouse button. Then, enter that user's password on the **lock screen**.

To log out and end the current graphical login session, select the system menu in the upper-right corner on the top bar and select **Power Off/Log Out > Log Out**. A window is displayed that offers the option to **Cancel** or confirm the **Log Out** action.

Power Off or Reboot the System

To shut down the system, from the system menu in the upper-right corner, select **Power Off/Log out > Power Off** or press **Ctrl+Alt+Del**. A window is displayed that offers the option to **Cancel** or confirm the **Power Off** action. If you do not make a choice, then the system automatically shuts down after 60 seconds.

To reboot the system, from the system menu in the upper-right corner, select **Power Off/Log out > Restart**. A window is displayed that offers the option to **Cancel** or confirm the **Restart** action. If you do not make a choice, then the system automatically restarts after 60 seconds.



References

GNOME Help

- `yelp`

GNOME Help: *Visual Overview of GNOME*

- `yelp help:gnome-help/shell-introduction`

GNOME 40 Web Page

<https://forty.gnome.org/>

► Guided Exercise

Access the Command Line with the Desktop

Log in through the graphical display manager as a regular user to become familiar with the GNOME Standard desktop environment that GNOME 40 provides.

Outcomes

- Log in to a Linux system by using the GNOME 40 desktop environment.
- Run commands from a shell prompt in a terminal program.

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 cli-desktop
```

Instructions

- 1. Log in to workstation as student with student as the password.
 - 1.1. On workstation, at the GNOME login screen, click the student user account. Enter student when prompted for the password.
 - 1.2. Press **Enter**.
- 2. Change the password for student from student to 55TurnK3y.



Important

The `finish` script resets the password for the student user to student. You must execute the script at the end of the exercise.

- 2.1. Open a **Terminal** window and use the `passwd` command at the shell prompt.
In the virtual learning environment with a visual keyboard, press the **Super** key twice to enter the **Activities** overview. Type `terminal` and then press **Enter** to start **Terminal**.
- 2.2. In the **Terminal** window that opens, type `passwd` at the shell prompt. Change the student password from student to 55TurnK3y.

```
[student@workstation ~]$ passwd
Changing password for user student.
Current password: student
New password: 55TurnK3y
Retype new password: 55TurnK3y
passwd: all authentication tokens updated successfully.
```

- ▶ **3.** Log out and log back in as **student** with **55TurnK3y** as the password to verify the changed password.
 - 3.1. Click the system menu in the upper-right corner.
 - 3.2. Select **Power Off/Log Out > Log Out**.
 - 3.3. Click **Log Out** in the confirmation dialog box that is displayed.
 - 3.4. At the GNOME login screen, click the **student** user account. Enter **55TurnK3y** when prompted for the password.
 - 3.5. Press **Enter**.
- ▶ **4.** Lock the screen.
 - 4.1. From the system menu in the upper-right corner, press the **Lock** button.
- ▶ **5.** Unlock the screen.
 - 5.1. Press **Enter** to unlock the screen.
 - 5.2. In the **Password** field, enter **55TurnK3y** as the password.
 - 5.3. Press **Enter**.
- ▶ **6.** Determine how to shut down **workstation** from the graphical interface, but **Cancel** the operation without shutting down the system.
 - 6.1. From the system menu in the upper-right corner, select **Power Off/Log Out > Power Off**. A dialog box is displayed with the options to either **Cancel** or **Power Off** the machine.
 - 6.2. Click **Cancel** in the dialog box that is displayed.

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 cli-desktop
```

Execute Commands with the Bash Shell

Objectives

- Save time when running commands from a shell prompt with Bash shortcuts.

Basic Command Syntax

The GNU Bourne-Again Shell (bash) is a program that interprets commands that the user types. Each string that is typed into the shell can have up to three parts: the command, options (which usually begin with a hyphen - or double hyphen -- characters), and arguments. Each word that is typed into the shell is separated from other words with spaces. Commands are the names of programs that are installed on the system. Each command has its options and arguments.

When you are ready to execute a command, press the **Enter** key. Type each command on a separate line. The command output is displayed before the following shell prompt appears.

```
[user@host ~]$ whoami
user
[user@host ~]$
```

To type more than one command on a single line, use the semicolon (;) as a command separator. A semicolon is a member of a class of characters called *metacharacters* that have a special interpretation for bash. In this case, the output of both commands is displayed before the following shell prompt appears.

The following example shows how to combine two commands (command1 and command2) on the command line.

```
[user@host ~]$ command1 ; command2
command1 output
command2 output
[user@host ~]$
```

Write Simple Commands

The **date** command displays the current date and time. The superuser or a privileged user can also use the **date** command to set the system clock. Use the plus sign (+) as an argument to specify a format string for the date command.

```
[user@host ~]$ date
Sun Feb 27 08:32:42 PM EST 2022
[user@host ~]$ date +%R
20:33
[user@host ~]$ date +%x
02/27/2022
```

The **passwd** command with no options changes the current user's password. To change the password, first specify the original password for the account. By default, the **passwd** command

is configured to require a strong password, to consist of lowercase letters, uppercase letters, numbers, and symbols, and not to be based on a dictionary word. A superuser or privileged user can use the `passwd` command to change another user's password.

```
[user@host ~]$ passwd
Changing password for user user.
Current password: old_password
New password: new_password
Retype new password: new_password
passwd: all authentication tokens updated successfully.
```

Linux does not require file name extensions to classify files by type. The `file` command scans the compiled header of a file for a 2-digit magic number and displays its type. Text files are recognized because they are not compiled.

```
[user@host ~]$ file /etc/passwd
/etc/passwd: ASCII text
[user@host ~]$ file /bin/passwd
/bin/passwd: setuid ELF 64-bit LSB pie executable, x86-64, version 1
(SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=a467cb9c8fa7306d41b96a820b0178f3a9c66055, for GNU/Linux 3.2.0,
stripped
[user@host ~]$ file /home
/home: directory
```

View the Contents of Files

The `cat` command is often used in Linux. Use this command to create single or multiple files, view the contents of files, concatenate the contents from various files, and redirect contents of the file to a terminal or to files.

The following example shows how to view the contents of the `/etc/passwd` file:

```
[user@host ~]$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
...output omitted...
```

To display the contents of multiple files, add the file names to the `cat` command as arguments:

```
[user@host ~]$ cat file1 file2
Hello World!!
Introduction to Linux commands.
```

Some files are long and might need more space to be displayed than the terminal provides. The `cat` command does not display the contents of a file as pages. The `less` command displays one page of a file at a time and you can scroll at your leisure.

Use the `less` command to page forward and backward through longer files than can fit on one terminal window. Use the UpArrow key and the DownArrow key to scroll up and down. Press `q` to exit the command.

The `head` and `tail` commands display the beginning and the end of a file, respectively. By default, these commands display 10 lines of the file, but they both have a `-n` option to specify a different number of lines.

```
[user@host ~]$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
[user@host ~]$ tail -n 3 /etc/passwd
gdm:x:42:42::/var/lib/gdm:/sbin/nologin
gnome-initial-setup:x:980:978::/run/gnome-initial-setup:/sbin/nologin
dnsmasq:x:979:977:Dnsmasq DHCP and DNS server:/var/lib/dnsmasq:/sbin/nologin
```

The `wc` command counts lines, words, and characters in a file. Use the `-l`, `-w`, or `-c` options to display only the given number of lines, words, or characters, respectively.

```
[user@host ~]$ wc /etc/passwd
41  98 2338 /etc/passwd
[user@host ~]$ wc -l /etc/passwd ; wc -l /etc/group
41 /etc/passwd
63 /etc/group
[user@host ~]$ wc -c /etc/group /etc/hosts
883 /etc/group
114 /etc/hosts
997 total
```

Understand Tab Completion

With tab completion, users can quickly complete commands or file names after typing enough at the prompt to make it unique. If the typed characters are not unique, then pressing the Tab key twice displays all commands that begin with the typed characters.

```
[user@host ~]$ pasTab+Tab ❶
passwd      paste       pasuspender
[user@host ~]$ passTab ❷
[user@host ~]$ passwd
Changing password for user user.
Current password:
```

- ❶ Press Tab twice.
- ❷ Press Tab once.

Tab completion helps to complete file names when typing them as arguments to commands. Press Tab to complete as much of the file name as possible. Pressing Tab a second time causes the

shell to list all files that the current pattern matches. Type additional characters until the name is unique, and then use tab completion to complete the command.

```
[user@host ~]$ ls /etc/pasTab ❶
[user@host ~]$ ls /etc/passwdTab ❷
passwd passwd-
```

- ❶ Press Tab once.
- ❷ Press Tab once.

Use the `useradd` command to create users on the system. The `useradd` command has many options that might be hard to remember. By using tab completion, you can complete the option name with minimal typing.

```
[root@host ~]# useradd --Tab+Tab ❶
--badnames          --gid                --no-log-init        --shell
--base-dir           --groups             --non-unique         --skel
--btrfs-subvolume-home --help              --no-user-group      --system
--comment            --home-dir           --password           --uid
--create-home        --inactive           --prefix             --user-
group
--defaults           --key                --root
--expiredate         --no-create-home     --selinux-user
```

- ❶ Press Tab twice.

Write a Long Command on Multiple Lines

Commands with many options and arguments can quickly become long and are automatically wrapped by the command window when the cursor reaches the right margin. Instead, type a long command by using more than one line for easier reading.

To write one command in more than one line, use a backslash character (`\`), which is referred to as the *escape character*. The backslash character ignores the meaning of the following character.

Previously, you learned that to complete a command entry, you press the **Enter** key, the newline character. By escaping the newline character, the shell moves to a new command line without executing the command. This way, the shell acknowledges the request by displaying a continuation prompt on an empty new line, which is known as the secondary prompt, and uses the greater-than character (`>`) by default. Commands can continue over many lines.

One issue with the secondary prompt's use of the greater-than character (`>`) is that new learners might mistakenly insert it as part of the typed command. Then, the shell interprets a typed greater-than character as *output redirection*, which the user did not intend. Output redirection is discussed in an upcoming chapter. This course book does not show secondary prompts in screen outputs, to avoid confusion. A user still sees the secondary prompt in their shell window, but the course material intentionally displays only the characters to be typed, as demonstrated in the following example.

```
[user@host ~]$ head -n 3 \
/usr/share/dict/words \
/usr/share/dict/linux.words
==> /usr/share/dict/words <==
```

```
1080
10-point
10th

==> /usr/share/dict/linux.words <==
1080
10-point
10th
```

Display the Command History

The `history` command displays a list of previously executed commands that are prefixed with a command number.

The exclamation point character (!) is a metacharacter to expand previous commands without retyping them. The `!number` command expands to the command that matches the specified number. The `!string` command expands to the most recent command that begins with the specified string.

```
[user@host ~]$ history
...output omitted...
 23 clear
 24 who
 25 pwd
 26 ls /etc
 27 uptime
 28 ls -l
 29 date
 30 history
[user@host ~]$ !ls
ls -l
total 0
drwxr-xr-x. 2 student student 6 Feb 27 19:24 Desktop
...output omitted...
[user@host ~]$ !26
ls /etc
abrt                hosts                pulse
adjtime              hosts.allow          purple
aliases              hosts.deny            qemu-ga
...output omitted...
```

The arrow keys help to navigate through previous commands in the shell's history. The `UpArrow` edits the previous command in the history list. The `DownArrow` edits the next command in the history list. The `LeftArrow` and `RightArrow` move the cursor left and right in the current command from the history list so that you can edit the command before running it.

Use either the `Esc+.` or `Alt+.` key combination simultaneously to insert the last word of the previous command at the cursor's current location. The repeated use of the key combination replaces that text with the last word of earlier commands in history. The `Alt+.` key combination is particularly convenient, because you can hold down `Alt` and press `.` repeatedly to quickly cycle earlier commands.

Edit the Command Line

When used interactively, `bash` has a command-line editing feature. Use the text editor commands to move around and modify the currently typed command. Using the arrow keys to move within the current command and to step through the command history was introduced earlier in this section. The following table shows further powerful editing commands.

Useful Command-line Editing Shortcuts

Shortcut	Description
Ctrl+A	Jump to the beginning of the command line.
Ctrl+E	Jump to the end of the command line.
Ctrl+U	Clear from the cursor to the beginning of the command line.
Ctrl+K	Clear from the cursor to the end of the command line.
Ctrl+LeftArrow	Jump to the beginning of the previous word on the command line.
Ctrl+RightArrow	Jump to the end of the next word on the command line.
Ctrl+R	Search the history list of commands for a pattern.

These command-line editing commands are the most helpful for new users. For other commands, refer to the `bash(1)` man page.



References

`bash(1)`, `date(1)`, `file(1)`, `magic(5)`, `cat(1)`, `more(1)`, `less(1)`, `head(1)`, `passwd(1)`, `tail(1)`, and `wc(1)` man pages

► Quiz

Execute Commands with the Bash Shell

Choose the correct answers to the following questions:

- 1. Which Bash command displays the last five lines of the `/var/log/messages` file?
 - a. `head -n 10 /var/log/messages`
 - b. `tail 10 /var/log/messages`
 - c. `tail -n 5 /var/log/messages`
 - d. `tail -l 10 /var/log/messages`
 - e. `less /var/log/messages`
- 2. Which Bash shortcut or command separates commands on the same line?
 - a. Pressing Tab
 - b. `history`
 - c. `;`
 - d. `!string`
 - e. Pressing Esc+.
- 3. Which Bash command is used to change a user's password?
 - a. `password`
 - b. `pass`
 - c. `passwd`
 - d. `usermod`
 - e. `userpassword`
- 4. Which Bash command is used to display the file type?
 - a. `file`
 - b. `less`
 - c. `cat`
 - d. `history`
 - e. `view`
- 5. Which Bash shortcut or command is used for completing commands, file names, and options?
 - a. `;`
 - b. `!number`
 - c. `history`
 - d. Pressing Tab
 - e. Pressing Esc+.

- ▶ 6. Which Bash shortcut or command re-executes a specific command in the history list?
 - a. Pressing Tab
 - b. *!number*
 - c. *!string*
 - d. history
 - e. Pressing Esc+.

- ▶ 7. Which Bash shortcut or command jumps to the beginning of the command line?
 - a. *!number*
 - b. *!string*
 - c. Pressing Ctrl+LeftArrow
 - d. Pressing Ctrl+K
 - e. Pressing Ctrl+A

- ▶ 8. Which Bash shortcut or command displays the list of previously executed commands?
 - a. Pressing Tab
 - b. *!string*
 - c. *!number*
 - d. history
 - e. Pressing Esc+.

- ▶ 9. Which Bash shortcut or command copies the last argument of previous commands?
 - a. Pressing Ctrl+K
 - b. Pressing Ctrl+A
 - c. *!number*
 - d. Pressing Esc+.

► Solution

Execute Commands with the Bash Shell

Choose the correct answers to the following questions:

- 1. Which Bash command displays the last five lines of the `/var/log/messages` file?
 - a. `head -n 10 /var/log/messages`
 - b. `tail 10 /var/log/messages`
 - c. `tail -n 5 /var/log/messages`
 - d. `tail -l 10 /var/log/messages`
 - e. `less /var/log/messages`
- 2. Which Bash shortcut or command separates commands on the same line?
 - a. Pressing Tab
 - b. `history`
 - c. `;`
 - d. `!string`
 - e. Pressing Esc+.
- 3. Which Bash command is used to change a user's password?
 - a. `password`
 - b. `pass`
 - c. `passwd`
 - d. `usermod`
 - e. `userpassword`
- 4. Which Bash command is used to display the file type?
 - a. `file`
 - b. `less`
 - c. `cat`
 - d. `history`
 - e. `view`
- 5. Which Bash shortcut or command is used for completing commands, file names, and options?
 - a. `;`
 - b. `!number`
 - c. `history`
 - d. Pressing Tab
 - e. Pressing Esc+.

- ▶ 6. Which Bash shortcut or command re-executes a specific command in the history list?
 - a. Pressing Tab
 - b. `!number`
 - c. `!string`
 - d. `history`
 - e. Pressing Esc+.

- ▶ 7. Which Bash shortcut or command jumps to the beginning of the command line?
 - a. `!number`
 - b. `!string`
 - c. Pressing Ctrl+LeftArrow
 - d. Pressing Ctrl+K
 - e. Pressing Ctrl+A

- ▶ 8. Which Bash shortcut or command displays the list of previously executed commands?
 - a. Pressing Tab
 - b. `!string`
 - c. `!number`
 - d. `history`
 - e. Pressing Esc+.

- ▶ 9. Which Bash shortcut or command copies the last argument of previous commands?
 - a. Pressing Ctrl+K
 - b. Pressing Ctrl+A
 - c. `!number`
 - d. Pressing Esc+.

► Lab

Access the Command Line

Use the Bash shell to execute commands.

Outcomes

- Successfully run simple programs from the Bash shell command line.
- Execute commands to identify file types and to display parts of text files.
- Practice using some Bash command history shortcuts to more efficiently repeat commands or parts of commands.

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 cli-review
```

Instructions

1. Use the `date` command to display the current time and date.
2. Display the current time in 24-hour clock time (for example, 13:57). Hint: The format string that displays that output is `%R`.
3. What kind of file is `/home/student/zcat`? Is it readable by humans?
4. Use the `wc` command and Bash shortcuts to display the size of the `zcat` file.
5. Display the first 10 lines of the `zcat` file.
6. Display the last 10 lines of the `zcat` file.
7. Repeat the previous command exactly with four or fewer keystrokes.
8. Use the `tail` command `-n 20` option to display the last 20 lines in the file. Use command-line editing to accomplish this task with a minimal number of keystrokes.
9. Use the shell history to run the `date +%R` command again.

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 cli-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 cli-review
```

► Solution

Access the Command Line

Use the Bash shell to execute commands.

Outcomes

- Successfully run simple programs from the Bash shell command line.
- Execute commands to identify file types and to display parts of text files.
- Practice using some Bash command history shortcuts to more efficiently repeat commands or parts of commands.

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 cli-review
```

Instructions

1. Use the `date` command to display the current time and date.

```
[student@workstation ~]$ date
Mon Feb 28 01:57:25 PM PDT 2022
```

2. Display the current time in 24-hour clock time (for example, 13:57). Hint: The format string that displays that output is `%R`.
 - 2.1. Use the `+%R` argument with the `date` command to display the current time in 24-hour clock time.

```
[student@workstation ~]$ date +%R
13:58
```

3. What kind of file is `/home/student/zcat`? Is it readable by humans?
 - 3.1. Use the `file` command to determine its file type.

```
[student@workstation ~]$ file zcat
zcat: a /usr/bin/sh script, ASCII text executable
```

4. Use the `wc` command and Bash shortcuts to display the size of the `zcat` file.
 - 4.1. You can use the `wc` command to display the number of lines, words, and bytes in the `zcat` script. Instead of retyping the file name, use the Bash history shortcut `Esc+`.

(the keys `Esc` and `.` pressed at the same time) to reuse the argument from the previous command.

```
[student@workstation ~]$ wc Esc+.
[student@workstation ~]$ wc zcat
  51  299 1988 zcat
```

5. Display the first 10 lines of the `zcat` file.

5.1. The `head` command displays the beginning of the file. Try the `Esc+.` shortcut again.

```
[student@workstation ~]$ head Esc+.
[student@workstation ~]$ head zcat
#!/bin/sh
# Uncompress files to standard output.

# Copyright (C) 2007, 2010-2018 Free Software Foundation, Inc.

# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3 of the License, or
# (at your option) any later version.
```

6. Display the last 10 lines of the `zcat` file.

6.1. Use the `tail` command to display the last 10 lines of the `zcat` file.

```
[student@workstation ~]$ tail Esc+.
[student@workstation ~]$ tail zcat
With no FILE, or when FILE is -, read standard input.

Report bugs to "<bug-gzip@gnu.org>."

case $1 in
--help)  printf '%s\n' "$usage"  || exit 1;;
--version) printf '%s\n' "$version" || exit 1;;
esac

exec gzip -cd "$@"
```

7. Repeat the previous command exactly with four or fewer keystrokes.

7.1. Repeat the previous command exactly with four or fewer keystrokes. Press the `UpArrow` key once to scroll back one command through the command history, and then press `Enter` (uses two keystrokes). An alternative would be to enter the shortcut command `!!` and then press `Enter` (uses four keystrokes) to run the most recent command in the command history. Try both methods.

```
[student@workstation]$ !!
tail zcat
With no FILE, or when FILE is -, read standard input.

Report bugs to "<bug-gzip@gnu.org>."
```

```
case $1 in
--help)  printf '%s\n' "$usage"  || exit 1;;
--version) printf '%s\n' "$version" || exit 1;;
esac

exec gzip -cd "$@"
```

8. Use the `tail` command `-n 20` option to display the last 20 lines in the file. Use command-line editing to accomplish this task with a minimal number of keystrokes.
 - 8.1. Use `UpArrow` key to display the previous command. Next, use the `Ctrl+A` key combination to move the cursor to the beginning of the line. Next, use the `Ctrl+RightArrow` key combination to jump to the next word. Next, add the `-n 20` option and press `Enter` to execute the command.

```
[student@workstation ~]$ tail -n 20 zcat
-l, --list          list compressed file contents
-q, --quiet         suppress all warnings
-r, --recursive    operate recursively on directories
-S, --suffix=SUF   use suffix SUF on compressed files
--synchronous      synchronous output (safer if system crashes, but slower)
-t, --test         test compressed file integrity
-v, --verbose      verbose mode
--help            display this help and exit
--version         display version information and exit
```

With no FILE, or when FILE is -, read standard input.

Report bugs to "<bug-gzip@gnu.org>."

```
case $1 in
--help)  printf '%s\n' "$usage"  || exit 1; exit;;
--version) printf '%s\n' "$version" || exit 1; exit;;
esac

exec gzip -cd "$@"
```

9. Use the shell history to run the `date +%R` command again.
 - 9.1. Use the `history` command to display the list of previous commands and to identify the specific `date` command to execute. Use `!number` to run the command, where *number* is the command number to use from the output of the `history` command.
Your shell history might differ from the following example. Determine the command number to use based on the output of your own `history` command.

```
[student@workstation ~]$ history
1  date
2  date +%R
3  file zcat
4  wc zcat
5  head zcat
6  tail zcat
7  tail -n 20 zcat
```

```
8 history
[student@workstation ~]$ !2
date +%R
14:02
```

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 cli-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 cli-review
```

Summary

- The Bash shell is a command interpreter that prompts interactive users to specify Linux commands.
- Many commands have a `--help` option that displays a usage message or screen.
- You can use workspaces to organize multiple application windows.
- The **Activities** button at the upper-left corner of the top bar provides an overview mode that helps to organize windows and to start applications.
- The `file` command scans the beginning of a file and displays what type it is.
- The `head` and `tail` commands display the beginning and end of a file, respectively.
- You can use tab completion to complete file names when typing them as arguments to commands.
- You can use the graphical interface for many administrative tasks. You can disable the interface to preserve resources for running applications.
- You can write many commands in the same line by using the semicolon `;` character, and can run a single command in multiple lines by using the backslash `\` character.

Chapter 3

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.
- Specify the absolute location and relative location of files to the current working directory, determine and change the working directory, and list the contents of directories.
- Create, copy, move, and remove files and directories.
- Create multiple file name references to 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)
- Specify Files by Name (and Quiz)
- Manage Files with Command-line Tools (and Guided Exercise)
- 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.

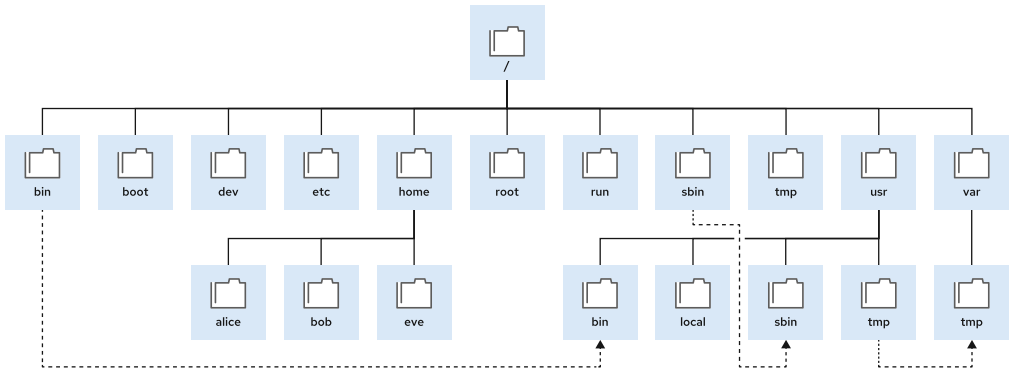


Figure 3.1: Significant file-system directories in Red Hat Enterprise Linux 9

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 that is 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 /boot subdirectory 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 in the /usr directory include the following commands: • /usr/bin: User commands • /usr/sbin: System administration commands • /usr/local: Locally customized software
/var	System-specific variable data should persist between boots. Files that dynamically change, such as databases, cache directories, log files, printer-spoiled 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

Specify Files by Name

Objectives

- Specify the absolute location and relative location of files to the current working directory, determine and change the working directory, and list the contents of directories.

Absolute Paths and Relative Paths

The *path* of a file or directory specifies its unique file-system location. Following a file path traverses one or more named subdirectories, which are delimited by a forward slash (/), until the destination is reached. Directories, also called *folders*, can contain other files and other subdirectories. Directories are referenced in the same manner as files.



Important

A space character is acceptable in a Linux file name. The shell also uses spaces to distinguish options and arguments on the command line. If a command includes a file with a space in its name, then the shell can misinterpret the command and assume that the file name is multiple arguments. To avoid this mistake, surround such file names in quotation marks so that the shell interprets the name as a single argument. Red Hat recommends avoiding spaces at all in file names.

Absolute Paths

An *absolute path* is a *fully qualified name* that specifies the exact location of the file in the file-system hierarchy. The absolute path begins at the root (/) directory and includes each subdirectory that must be traversed to reach the specific file. Every file in a file system has a unique absolute path name, which is recognized with a simple rule: a path name with a forward slash (/) as the first character is an absolute path name.

For example, the absolute path name for the system message log file is `/var/log/messages`. Absolute path names can be long to type, so files can also be located relative to the current working directory of your shell prompt.

The Current Working Directory and Relative Paths

When a user logs in and opens a command window, the initial location is typically the user's home directory. System processes also have an initial directory. Users and processes change to other directories as needed. The *working directory* and *current working directory* terms refer to their current location.

Similar to an absolute path, a *relative* path identifies a unique location, and specifies only the necessary path to reach the location from the working directory. Relative path names follow this rule: a path name with *anything other than* a forward slash as the first character is a relative path name. For example, relative to the `/var` directory, the message log file is `log/messages`.

Linux file systems, including `ext4`, `XFS`, `GFS2`, and `GlusterFS`, are case-sensitive. Creating the `FileCase.txt` and `filecase.txt` files in the same directory results in two unique files.

Non-Linux file systems might work differently. For example, VFAT, Microsoft NTFS, and Apple HFS+ have *case-preserving behavior*. Although these file systems are *not* case-sensitive, they do display file names with the file's original capitalization. By creating the files in the preceding example on a VFAT file system, both names would point to the same file instead of to two different files.

Navigate Paths in the File System

The `pwd` command displays the full path name of the current working directory for that shell. This command helps you to determine the syntax to reach files by using relative path names. The `ls` command lists directory contents for the specified directory or, if no directory is given, for the current working directory.

```
[user@host ~]$ pwd
/home/user
[user@host ~]$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
[user@host ~]$
```

Use the `cd` command to change your shell's current working directory. If you do not specify any arguments to the command, then it changes to your home directory.

In the following example, a mixture of absolute and relative paths are used with the `cd` command to change the current working directory for the shell.

```
[user@host ~]$ pwd
/home/user
[user@host ~]$ cd Videos
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd /home/user/Documents
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ cd
[user@host ~]$ pwd
/home/user
[user@host ~]$
```

In the preceding example, the default shell prompt also displays the last component of the absolute path to the current working directory. For example, for the `/home/user/Videos` directory, only the `Videos` directory is displayed. The prompt displays the tilde character (`~`) when your current working directory is your home directory.

The `touch` command updates the time stamp of a file to the current date and time without otherwise modifying it. This command is useful for creating empty files, and can be used for practice, because when you use the `touch` command with a file name that does not exist, the file is created. In the following example, the `touch` command creates practice files in the `Documents` and `Videos` subdirectories.

```
[user@host ~]$ touch Videos/blockbuster1.ogg
[user@host ~]$ touch Videos/blockbuster2.ogg
[user@host ~]$ touch Documents/thesis_chapter1.txt
[user@host ~]$ touch Documents/thesis_chapter2.txt
[user@host ~]$
```

The `ls` command has multiple options for displaying attributes on files. The most common options are `-l` (long listing format), `-a` (all files, including *hidden* files), and `-R` (recursive, to include the contents of all subdirectories).

```
[user@host ~]$ ls -l
total 0
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Desktop
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Documents
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Downloads
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Music
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Pictures
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Public
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Templates
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Videos
[user@host ~]$ ls -la
total 40
drwx-----. 17 user user 4096 Mar  2 03:07 .
drwxr-xr-x.  4 root root  35 Feb 10 10:48 ..
drwxr-xr-x.  4 user user   27 Mar  2 03:01 .ansible
-rw-----.  1 user user  444 Mar  2 04:32 .bash_history
-rw-r--r--.  1 user user   18 Aug  9 2021 .bash_logout
-rw-r--r--.  1 user user  141 Aug  9 2021 .bash_profile
-rw-r--r--.  1 user user  492 Aug  9 2021 .bashrc
drwxr-xr-x.  9 user user 4096 Mar  2 02:45 .cache
drwxr-xr-x.  9 user user 4096 Mar  2 04:32 .config
drwxr-xr-x.  2 user user    6 Mar  2 02:45 Desktop
drwxr-xr-x.  2 user user    6 Mar  2 02:45 Documents
...output omitted...
```

At the top of the listing are two special directories. One dot (.) refers to the current directory, and two dots (..) refer to the parent directory. These special directories exist in every directory on the system, and they are useful when using file management commands.



Important

File names that begin with a dot (.) indicate *hidden* files; you cannot see them in the normal view with `ls` and other commands. This behavior is *not* a security feature. Hidden files keep necessary user configuration files from cluttering home directories. Many commands process hidden files only with specific command-line options, and prevent one user's configuration from being accidentally copied to other directories or users.

To protect file *contents* from improper viewing requires the use of *file permissions*.

You can also use the tilde (~) special character in combination with other commands for better interaction with the home directory.

```
[user@host ~]$ cd /var/log/
[user@host log]$ ls -l ~
total 0
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Desktop
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Documents
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Downloads
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Music
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Pictures
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Public
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Templates
drwxr-xr-x. 2 user user 6 Mar  2 02:45 Videos
[user@host ~]$
```

The `cd` command has many options. Some options are useful to practice early and to use often. The `cd -` command changes to the previous directory, where the user was *previously* to the current directory. The following example illustrates this behavior, and alternates between two directories, which is useful when processing a series of similar tasks.

```
[user@host ~]$ cd Videos
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd /home/user/Documents
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ cd -
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd -
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ cd -
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd
[user@host ~]$
```

The `cd ..` command uses the `(..)` hidden directory to move up one level to the parent directory, without needing to know the exact parent name. The other hidden directory `(.)` specifies the *current directory* on commands where the current location is either the source or destination argument, and avoids the need to type the directory's absolute path name.

```
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd .
[user@host Videos]$ pwd
/home/user/Videos
[user@host Videos]$ cd ..
[user@host ~]$ pwd
/home/user
[user@host ~]$ cd ..
[user@host home]$ pwd
/home
[user@host home]$ cd ..
```

```
[user@host ~]$ pwd
/
[user@host ~]$ cd
[user@host ~]$ pwd
/home/user
[user@host ~]$
```



References

info libc 'file name resolution' (*GNU C Library Reference Manual*)

- Section 11.2.2 File Name Resolution

https://www.gnu.org/software/libc/manual/html_node/File-Name-Resolution.html

bash(1), cd(1), ls(1), pwd(1), unicode(7), and utf-8(7) man pages

UTF-8 and Unicode

<http://www.utf-8.com/>

► Quiz

Specify Files by Name

Choose the correct answers to the following questions:

- 1. Which command returns to your current home directory, assuming that the current working directory is /tmp and your home directory is /home/user?
 - a. cd
 - b. cd ..
 - c. cd .
 - d. cd *
 - e. cd /home
- 2. Which command displays the absolute path name of the current location?
 - a. cd
 - b. pwd
 - c. ls ~
 - d. ls -d
- 3. Which command returns you to the working directory before the current working directory?
 - a. cd -
 - b. cd -p
 - c. cd ~
 - d. cd ..
- 4. Which command changes the working directory up two levels from the current location?
 - a. cd ~/..
 - b. cd ../..
 - c. cd ../../
 - d. cd ~/
- 5. Which command lists files in the current location, with a long format, and including hidden files?
 - a. llong ~
 - b. ls -a
 - c. ls -l
 - d. ls -al

- ▶ 6. Which command creates an empty file called `hello world.py` in the user home directory, assuming that your current directory is `/home`?
 - a. `touch ./helloworld.py`
 - b. `touch ~/helloworld.py`
 - c. `touch helloworld.py`
 - d. `touch ../helloworld.py`

- ▶ 7. Which command changes the working directory to the parent of the current location?
 - a. `cd ~`
 - b. `cd ..`
 - c. `cd ../../`
 - d. `cd -u1`

- ▶ 8. Which command changes the working directory to `/tmp` if the current working directory is `/home/student`?
 - a. `cd tmp`
 - b. `cd ..`
 - c. `cd ../../tmp`
 - d. `cd ~tmp`

► Solution

Specify Files by Name

Choose the correct answers to the following questions:

- 1. Which command returns to your current home directory, assuming that the current working directory is /tmp and your home directory is /home/user?
 - a. `cd`
 - b. `cd ..`
 - c. `cd .`
 - d. `cd *`
 - e. `cd /home`
- 2. Which command displays the absolute path name of the current location?
 - a. `cd`
 - b. `pwd`
 - c. `ls ~`
 - d. `ls -d`
- 3. Which command returns you to the working directory before the current working directory?
 - a. `cd -`
 - b. `cd -p`
 - c. `cd ~`
 - d. `cd ..`
- 4. Which command changes the working directory up two levels from the current location?
 - a. `cd ~/..`
 - b. `cd ../..`
 - c. `cd ../../..`
 - d. `cd ~/`
- 5. Which command lists files in the current location, with a long format, and including hidden files?
 - a. `llong ~`
 - b. `ls -a`
 - c. `ls -l`
 - d. `ls -al`

- ▶ 6. Which command creates an empty file called `hello world.py` in the user home directory, assuming that your current directory is `/home`?
 - a. `touch ./helloworld.py`
 - b. `touch ~/helloworld.py`
 - c. `touch helloworld.py`
 - d. `touch ../helloworld.py`

- ▶ 7. Which command changes the working directory to the parent of the current location?
 - a. `cd ~`
 - b. `cd ..`
 - c. `cd ../../`
 - d. `cd -u1`

- ▶ 8. Which command changes the working directory to `/tmp` if the current working directory is `/home/student`?
 - a. `cd tmp`
 - b. `cd ..`
 - c. `cd ../../tmp`
 - d. `cd ~tmp`

Manage Files with Command-line Tools

Objectives

- Create, copy, move, and remove files and directories.

Command-line File Management

Creating, copying, moving, and, removing files and directories are common operations for a system administrator. Without options, some commands are used to interact with files, or they can manipulate directories with the appropriate set of options.

Be aware of the options that are used when running a command. The meaning of some options might differ between commands.

Create Directories

The `mkdir` command creates one or more directories or subdirectories. It takes as an argument a list of paths to the directories that you want to create.

In the following example, files and directories are organized beneath the `/home/user/Documents` directory. Use the `mkdir` command and a space-delimited list of the directory names to create multiple directories.

```
[user@host ~]$ cd Documents
[user@host Documents]$ mkdir ProjectX ProjectY ProjectZ
[user@host Documents]$ ls
ProjectX ProjectY ProjectZ
```

If the directory exists, or a parent directory of the directory that you are trying to create does not exist, then the `mkdir` command fails and it displays an error.

The `mkdir` command `-p` (*parent*) option creates any missing parent directories for the requested destination. In the following example, the `mkdir` command creates three `ChapterN` subdirectories with one command. The `-p` option creates the missing `Thesis` parent directory.

```
[user@host Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapter2 Thesis/Chapter3
[user@host Documents]$ ls -R Thesis/
Thesis/:
Chapter1 Chapter2 Chapter3

Thesis/Chapter1:

Thesis/Chapter2:

Thesis/Chapter3:
```

Use the `mkdir` command `-p` option with caution, because spelling mistakes can create unintended directories without generating error messages. In the following example, imagine that

you are trying to create a **Watched** subdirectory in the **Videos** directory, but you accidentally omitted the letter "s" in **Videos** in your **mkdir** command.

```
[user@host ~]$ mkdir Video/Watched
mkdir: cannot create directory Video/Watched: No such file or directory
```

The **mkdir** command fails because the **Video** directory does not exist. If you had used the **mkdir** command with the **-p** option, then the **Video** directory would be created unintentionally. The **Watched** subdirectory would be created in that incorrect directory.

Copy Files and Directories

The **cp** command copies a file, and creates a file either in the current directory or in a different specified directory.

```
[user@host ~]$ cd Videos
[user@host Videos]$ cp blockbuster1.ogg blockbuster3.ogg
[user@host Videos]$ ls -l
total 0
-rw-rw-r--. 1 user user  0 Feb  8 16:23 blockbuster1.ogg
-rw-rw-r--. 1 user user  0 Feb  8 16:24 blockbuster2.ogg
-rw-rw-r--. 1 user user  0 Feb  8 16:34 blockbuster3.ogg
```

You can also use the **cp** command to copy multiple files to a directory. In this scenario, the last argument must be a directory. The copied files retain their original names in the new directory. If a file with the same name exists in the target directory, then the existing file is overwritten.



Note

By default, the **cp** command does not copy directories; it ignores them.

In the following example, two directories are listed as arguments, the **Thesis** and **ProjectX** directories. The last argument, the **ProjectX** directory, is the target and is valid as a destination. The **Thesis** argument is ignored by the **cp** command, because it is intended to be copied and it is a directory.

```
[user@host Documents]$ cp thesis_chapter1.txt thesis_chapter2.txt Thesis ProjectX
cp: -r not specified; omitting directory 'Thesis'
[user@host Documents]$ ls Thesis ProjectX
ProjectX:
thesis_chapter1.txt  thesis_chapter2.txt

Thesis:
Chapter1  Chapter2  Chapter3
```

The **Thesis** directory failed to copy, but the copying of the **thesis_chapter1.txt** and **thesis_chapter2.txt** files succeeded.

You can copy directories and their contents by using the **cp** command **-r** option. Keep in mind that you can use the **.** and **..** special directories in command combinations. In the following example, the **Thesis** directory and its contents are copied to the **ProjectY** directory.

```
[user@host Documents]$ cd ProjectY
[user@host ProjectY]$ cp -r ../Thesis/ .
[user@host ProjectY]$ ls -lR
.:
total 0
drwxr-xr-x. 5 user user 54 Mar  7 15:08 Thesis

./Thesis:
total 0
drwxr-xr-x. 2 user user 6 Mar  7 15:08 Chapter1
drwxr-xr-x. 2 user user 6 Mar  7 15:08 Chapter2
drwxr-xr-x. 2 user user 6 Mar  7 15:08 Chapter3

./Thesis/Chapter1:
total 0

./Thesis/Chapter2:
total 0

./Thesis/Chapter3:
total 0
```

Move Files and Directories

The `mv` command moves files from one location to another. If you think of the absolute path to a file as its full name, then moving a file is effectively the same as renaming a file. The contents of the files that are moved remain unchanged.

Use the `mv` command to *rename* a file. In the following example, the `mv thesis_chapter2.txt` command renames the `thesis_chapter2.txt` file to `thesis_chapter2_reviewed.txt` in the same directory.

```
[user@host Documents]$ ls -l
-rw-r--r--. 1 user user 7100 Mar  7 14:37 thesis_chapter1.txt
-rw-r--r--. 1 user user 11431 Mar  7 14:39 thesis_chapter2.txt
...output omitted...
[user@host Documents]$ mv thesis_chapter2.txt thesis_chapter2_reviewed.txt
[user@host Documents]$ ls -l
-rw-r--r--. 1 user user 7100 Mar  7 14:37 thesis_chapter1.txt
-rw-r--r--. 1 user user 11431 Mar  7 14:39 thesis_chapter2_reviewed.txt
...output omitted...
```

Use the `mv` command to *move* a file to a different directory. In the next example, the `thesis_chapter1.txt` file is moved from the `~/Documents` directory to the `~/Documents/Thesis/Chapter1` directory. You can use the `mv` command `-v` option to display a detailed output of the command operations.

```
[user@host Documents]$ ls Thesis/Chapter1
[user@host Documents]$
[user@host Documents]$ mv -v thesis_chapter1.txt Thesis/Chapter1
renamed 'thesis_chapter1.txt' -> 'Thesis/Chapter1/thesis_chapter1.txt'
[user@host Documents]$ ls Thesis/Chapter1
thesis_chapter1.txt
```

```
[user@host Documents]$ ls -l
-rw-r--r--. 1 user user 11431 Mar  7 14:39 thesis_chapter2_reviewed.txt
...output omitted...
```

Remove Files and Directories

The `rm` command removes files. By default, `rm` does not remove directories. You can use the `rm` command `-r` or the `--recursive` option to enable the `rm` command to remove directories and their contents. The `rm -r` command traverses each subdirectory first, and individually removes their files before removing each directory.

In the following example, the `rm` command removes the `thesis_chapter1.txt` file without options, but to remove the `Thesis/Chapter1` directory, you must add the `-r` option.

```
[user@host Documents]$ ls -l Thesis/Chapter1
-rw-r--r--. 1 user user 7100 Mar  7 14:37 thesis_chapter1.txt
[user@host Documents]$ rm Thesis/Chapter1/thesis_chapter1.txt
[user@host Documents]$ rm Thesis/Chapter1
rm: cannot remove 'Thesis/Chapter1': Is a directory
[user@host Documents]$ rm -r Thesis/Chapter1
[user@host Documents]$ ls -l Thesis
drwxr-xr-x. 2 user user 6 Mar  7 12:37 Chapter2
drwxr-xr-x. 2 user user 6 Mar  7 12:37 Chapter3
```



Important

Red Hat Enterprise Linux does not provide a command-line undelete feature, nor a "trash bin" from which you can restore files held for deletion. A trash bin is a component of a desktop environment such as GNOME, but is not used by commands that are run from a shell.

It is a good idea to verify your current working directory before you remove a file or directory by using relative paths.

```
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ ls -l thesis*
-rw-r--r--. 1 user user 11431 Mar  7 14:39 thesis_chapter2_reviewed.txt
[user@host Documents]$ rm thesis_chapter2_reviewed.txt
[user@host Documents]$ ls -l thesis*
ls: cannot access 'thesis*': No such file or directory
```

You can use the `rm` command `-i` option to interactively prompt for confirmation before deleting. This option is essentially the opposite of using the `rm` command `-f` option, which forces the removal without prompting the user for confirmation.

```
[user@host Documents]$ rm -ri Thesis
rm: descend into directory 'Thesis'? y
rm: descend into directory 'Thesis/Chapter2'? y
rm: remove regular empty file 'Thesis/Chapter2/thesis_chapter2.txt'? y
rm: remove directory 'Thesis/Chapter2'? y
rm: remove directory 'Thesis/Chapter3'? y
rm: remove directory 'Thesis'? y
```



Warning

If you specify both the `-i` and `-f` options, then the `-f` option takes priority and you are not prompted for confirmation before `rm` removes files.

You can also use the `rmdir` command to remove empty directories. Use the `rm` command `-r` option to remove non-empty directories.

```
[user@host Documents]$ pwd
/home/user/Documents
[user@host Documents]$ rmdir ProjectZ
[user@host Documents]$ rmdir ProjectX
rmdir: failed to remove 'ProjectX': Directory not empty
[user@host Documents]$ rm -r ProjectX
[user@host Documents]$
```



References

`cp(1)`, `mkdir(1)`, `mv(1)`, `rm(1)`, and `rmdir(1)` man pages

► Guided Exercise

Manage Files with Command-line Tools

Create, organize, copy, and remove files and directories.

Outcomes

- Create, organize, copy, and remove files and directories.

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-manage
```

Instructions

- 1. Log in to the **servera** machine as the **student** user. In the **student** user's home directory, create three subdirectories: **Music**, **Pictures**, and **Videos**.
 - 1.1. Use the **ssh** command to log in to the **servera** machine as the **student** user. The systems are configured to use SSH keys for authentication; therefore, a password is not required.

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

- 1.2. In the **student** user's home directory, use the **mkdir** command to create three subdirectories: **Music**, **Pictures**, and **Videos**.

```
[student@servera ~]$ mkdir Music Pictures Videos
```

- 2. Use the **touch** command to create sets of empty practice files to use during this lab. In each set, replace **X** with the numbers 1 through 6.
 - Create six files with names of the form **songX.mp3**.
 - Create six files with names of the form **snapX.jpg**.
 - Create six files with names of the form **filmX.avi**.

```
[student@servera ~]$ touch song1.mp3 song2.mp3 song3.mp3 \
song4.mp3 song5.mp3 song6.mp3
[student@servera ~]$ touch snap1.jpg snap2.jpg snap3.jpg \
snap4.jpg snap5.jpg snap6.jpg
[student@servera ~]$ touch film1.avi film2.avi film3.avi \
```

```

film4.avi film5.avi film6.avi
[student@servera ~]$ ls -l
total 0
-rw-r--r--. 1 student student 0 Mar  7 20:58 film1.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film2.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film3.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film4.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film5.avi
-rw-r--r--. 1 student student 0 Mar  7 20:58 film6.avi
drwxr-xr-x. 2 student student 6 Mar  7 20:58 Music
drwxr-xr-x. 2 student student 6 Mar  7 20:58 Pictures
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap1.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap2.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap3.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap4.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap5.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap6.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 song1.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song2.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song3.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song4.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song5.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song6.mp3
drwxr-xr-x. 2 student student 6 Mar  7 20:58 Videos

```

- ▶ 3. Move the song files (.mp3 extension) to the Music directory, the snapshot files (.jpg extension) to the Pictures directory, and the movie files (.avi extension) to the Videos directory.

```

[student@servera ~]$ mv song1.mp3 song2.mp3 song3.mp3 \
song4.mp3 song5.mp3 song6.mp3 Music
[student@servera ~]$ mv snap1.jpg snap2.jpg snap3.jpg \
snap4.jpg snap5.jpg snap6.jpg Pictures
[student@servera ~]$ mv film1.avi film2.avi film3.avi \
film4.avi film5.avi film6.avi Videos
[student@servera ~]$ ls -l Music Pictures Videos
Music:
total 0
-rw-r--r--. 1 student student 0 Mar  7 20:58 song1.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song2.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song3.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song4.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song5.mp3
-rw-r--r--. 1 student student 0 Mar  7 20:58 song6.mp3

Pictures:
total 0
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap1.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap2.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap3.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap4.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap5.jpg
-rw-r--r--. 1 student student 0 Mar  7 20:58 snap6.jpg

```

```
Videos:
total 0
-rw-r--r--. 1 student student 0 Mar 7 20:58 film1.avi
-rw-r--r--. 1 student student 0 Mar 7 20:58 film2.avi
-rw-r--r--. 1 student student 0 Mar 7 20:58 film3.avi
-rw-r--r--. 1 student student 0 Mar 7 20:58 film4.avi
-rw-r--r--. 1 student student 0 Mar 7 20:58 film5.avi
-rw-r--r--. 1 student student 0 Mar 7 20:58 film6.avi
```

- 4. Create three subdirectories for organizing your files, and name the subdirectories **friends**, **family**, and **work**. Use a single command to create all three subdirectories at the same time.

```
[student@servera ~]$ mkdir friends family work
[student@servera ~]$ ls -l
total 0
drwxr-xr-x. 2 student student 6 Mar 7 21:01 family
drwxr-xr-x. 2 student student 6 Mar 7 21:01 friends
drwxr-xr-x. 2 student student 108 Mar 7 21:00 Music
drwxr-xr-x. 2 student student 108 Mar 7 21:00 Pictures
drwxr-xr-x. 2 student student 108 Mar 7 21:00 Videos
drwxr-xr-x. 2 student student 6 Mar 7 21:01 work
```

- 5. Copy files that contain numbers 1 and 2 to the **friends** directory, and files that contain numbers 3 and 4 to the **family** directory. Keep in mind that you are making copies; therefore, the original files must remain in their original locations after you complete the step.

When you copy files from multiple locations into a single location, Red Hat recommends that you change to the destination directory before you copy the files. Use the simplest path syntax, whether absolute or relative, to reach the source for each file management task.

- 5.1. Copy files that contain numbers 1 and 2 to the **friends** directory.

```
[student@servera ~]$ cd friends
[student@servera friends]$ cp ~/Music/song1.mp3 ~/Music/song2.mp3 \
~/Pictures/snap1.jpg ~/Pictures/snap2.jpg ~/Videos/film1.avi \
~/Videos/film2.avi .
[student@servera friends]$ ls -l
total 0
-rw-r--r--. 1 student student 0 Mar 7 21:02 film1.avi
-rw-r--r--. 1 student student 0 Mar 7 21:02 film2.avi
-rw-r--r--. 1 student student 0 Mar 7 21:02 snap1.jpg
-rw-r--r--. 1 student student 0 Mar 7 21:02 snap2.jpg
-rw-r--r--. 1 student student 0 Mar 7 21:02 song1.mp3
-rw-r--r--. 1 student student 0 Mar 7 21:02 song2.mp3
```

- 5.2. Copy files that contain numbers 3 and 4 to the **family** directory.

```
[student@servera friends]$ cd ../family
[student@servera family]$ cp ~/Music/song3.mp3 ~/Music/song4.mp3 \
~/Pictures/snap3.jpg ~/Pictures/snap4.jpg ~/Videos/film3.avi \
```



```
~/Videos/film4.avi .
[student@servera family]$ ls -l
total 0
total 0
-rw-r--r--. 1 student student 0 Mar  7 21:04 film3.avi
-rw-r--r--. 1 student student 0 Mar  7 21:04 film4.avi
-rw-r--r--. 1 student student 0 Mar  7 21:04 snap3.jpg
-rw-r--r--. 1 student student 0 Mar  7 21:04 snap4.jpg
-rw-r--r--. 1 student student 0 Mar  7 21:04 song3.mp3
-rw-r--r--. 1 student student 0 Mar  7 21:04 song4.mp3
```

- 6. Copy the **family** and **friends** directories and their contents to the **work** directory.

```
[student@servera family]$ cd ../work
[student@servera work]$ cp -r ~/family ~/friends .
[student@servera work]$ ls -l
total 0
drwxr-xr-x. 2 student student 108 Mar  7 21:05 family
drwxr-xr-x. 2 student student 108 Mar  7 21:05 friends
```

- 7. Your project tasks are now complete, and it is time to clean up the directories. Use the **rm -r** command to recursively delete the **family**, **friends**, and **work** directories and their contents.

```
[student@servera work]$ cd ..
[student@servera ~]$ rm -r family friends work
[student@servera ~]$ ls -l
total 0
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Music
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Pictures
drwxr-xr-x. 2 student student 108 Mar  7 21:00 Videos
```

- 8. 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-manage
```

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 that 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 inode structure 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 also show the new information. This behavior 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, which makes 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. 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. 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

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 file `/home/student/files/target.file`. Verify the link count for the original file and the new linked file.

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

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

- 2.3. Verify the link count for the original file `/home/student/files/target.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
```

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

When you type a command at the Bash shell prompt, the shell processes that command line through multiple *expansions* before running it. You can use these shell expansions to perform complex tasks that would otherwise be difficult or impossible.

Following are the main expansions that Bash shell performs:

- *Brace expansion*, which can generate multiple strings of characters
- *Tilde expansion*, which expand to a path to a user home directory
- *Variable expansion*, which replaces text with the value that is stored in a shell variable
- *Command substitution*, which replaces text with the output of a command
- *Pathname expansion*, which helps to select one or more files by pattern matching

Pathname expansion, historically called *globbing*, is one of the most useful features of Bash. With this feature, it is easier to manage many files. By using *metacharacters* that "expand" to match the file and path names that you are looking for, commands can act on a focused set of files at once.

Pathname Expansion and Pattern Matching

Pathname expansion expands a pattern of special characters that represent wild cards or classes of characters into a list of file names that match the pattern. Before the shell runs your command, it replaces the pattern with the list of file names that matched. If the pattern does not match anything, then the shell tries to use the pattern as a literal argument for the command that it runs. The following table lists common metacharacters and pattern classes that are used for pattern matching.

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

Pattern	Matches
<code>[:alpha:]</code>	Any alphabetic character
<code>[:lower:]</code>	Any lowercase character
<code>[:upper:]</code>	Any uppercase character
<code>[:alnum:]</code>	Any alphabetic character or digit
<code>[:punct:]</code>	Any printable character that is not a space or alphanumeric
<code>[:digit:]</code>	Any single digit from 0 to 9
<code>[:space:]</code>	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 alfa bravo charlie delta echo able baker cast dog easy
[user@host glob]$ ls
able alfa 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 file names that start with "a" and all the file names that contain an "a", respectively. The third command uses the asterisk and square brackets to match all the file names that start with "a" or "c".

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

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

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

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 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 create multiple files or directories.

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

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 username, if one matches, and replaces the string with the absolute path to that user's home directory. If no username matches, then the shell uses an actual tilde followed by the string of characters.

In the following example, the echo command displays the value of the tilde (~).

```
[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
```

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



Note

An earlier 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@host glob]$ echo The value of $HOME is your home directory.
The value of /home/user is your home directory.
[user@host glob]$ 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 special characters other than the dollar sign (\$), backslash (\), backtick (`), and exclamation point (!) from operating inside the quoted text. Double quotation marks block pathname expansion, but still allow command substitution and variable expansion to occur.

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

Use single quotation marks to interpret *all* text between the quotes literally.

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

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 is 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. Create 12 files with `tv_seasonX_episodeY.ogg` names 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 `mystery_chapterX.odf` names. 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 create 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 only 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 the `mystery_chapter5.odf` file 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 when 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
```

► Solution

Manage Files from the Command Line

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 is 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. Create 12 files with `tv_seasonX_episodeY.ogg` names 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 `mystery_chapterX.odf` names. 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_best_seller` subdirectory under the `Documents` directory, and

create 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:

Documents/project_plans:
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 only 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
```

- 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
```

- Change your current directory to the `changes` directory and use the `date +%F` command with command substitution to copy the `mystery_chapter5.odf` file 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
```

- 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 when 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
```

Summary

- Files on a Linux system are organized into a single inverted tree of directories, a file-system hierarchy.
- Absolute paths start with a forward slash character (/) and specify the location of a file in the file-system hierarchy.
- Relative paths do not start with a forward slash character.
- Relative paths specify a file location in relation to the current working directory.
- You can use commands in combination with the dot (.), double dot (..), and tilde (~) special characters to refer to a file location in the file system.
- The `mkdir`, `rmdir`, `cp`, `mv`, and `rm` commands are key commands to manage files in Linux.
- 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 4

Get Help in Red Hat Enterprise Linux

Goal

Resolve problems by using local help systems.

Objectives

- Find information in local Linux system manual pages.

Sections

- Read Manual Pages (and Guided Exercise)

Lab

- Get Help in Red Hat Enterprise Linux

Read Manual Pages

Objectives

- Find information in local Linux system manual pages.

Introduction to the Linux Manual Pages

One source of documentation that is generally available on the local system is system manual pages or *man pages*. Software packages ship these pages to provide documentation, and you can access them from the command line by using the `man` command. The pages are stored in subdirectories of the `/usr/share/man` directory.

Man pages originated from the historical Linux Programmer's Manual, which because of its size is split into multiple sections. Each section contains information about a particular topic.

Common Sections of the Linux Manual

Section	Content type	Description
1	User commands	Both executable and shell programs
2	System calls	Kernel routines that are invoked from user space
3	Library functions	Provided by program libraries
4	Special files	Such as device files
5	File formats	For many configuration files and structures
6	Games and screensavers	Historical section for amusing programs
7	Conventions, standards, and miscellaneous	Protocols and file systems
8	System administration and privileged commands	Maintenance tasks
9	Linux kernel API	Internal kernel calls

To distinguish identical topic names in different sections, man page references include the section number in parentheses after the topic. For example, `passwd(1)` describes the command to change passwords, whereas `passwd(5)` explains the `/etc/passwd` file format for storing local user accounts.

To read specific man pages, use the `man topic` command. The man pages display contents one screen at a time. The `man` command searches manual sections in alphanumeric order. For example, `man passwd` displays `passwd(1)` by default. To display the man page topic from a specific section, you can use the `man section topic` command. For example, `man 5 passwd` displays `passwd(5)`.

Popular system administration topics are in sections 1 (user commands), 5 (file formats), and 8 (administrative commands). Administrators who use certain troubleshooting tools also use section 2 (system calls). The remaining sections are generally for programmer reference or advanced administration.

Navigate and Search man Pages

It is a critical administration skill to search efficiently for topics and to navigate man pages. Although you can use GUI tools to configure common system resources, using the command-line interface is more efficient. To navigate the command line effectively, you must be able to find the information that you need in the man pages.

The following table lists some navigation commands when viewing man pages:

Navigate man Pages

Command	Result
Spacebar	Scroll forward (down) one screen.
PageDown	Scroll forward one screen.
PageUp	Scroll backward (up) one screen.
DownArrow	Scroll forward one line.
UpArrow	Scroll backward one line.
D	Scroll forward one half-screen.
U	Scroll backward one half-screen.
/string	Search forward for <i>string</i> in the man page.
N	Repeat previous search forward in the man page.
Shift+N	Repeat previous search backward in the man page.
G	Go to the start of the man page.
Shift+G	Go to the end of the man page.
Q	Exit man and return to the command shell prompt.



Important

You can use regular expressions to search in man pages. Although simple text search (such as `passwd`) works as expected, regular expressions use metacharacters (such as `$`, `*`, `.`, and `^`) for more sophisticated pattern matching. Therefore, searching with strings that include program expression metacharacters, such as `make $$$`, might yield unexpected results.

You can find more information about regular expressions and syntax in the `regex(7)` man topic.

Read man Pages

Man pages separate each topic into several parts. Most topics use the same headings and follow the same order. Typically, a topic does not feature all headings, because not all headings apply to all topics.

Common headings are as follows:

Headings

Heading	Description
NAME	Subject name, usually a command or file name, a brief description
SYNOPSIS	Summary of the command syntax
DESCRIPTION	Description to provide a basic understanding of the topic
OPTIONS	Explanation of the command execution options
EXAMPLES	Examples of how to use the command, function, or file
FILES	A list of files and directories that are related to the man page
SEE ALSO	Related information, normally other man page topics
BUGS	Known bugs in the software
AUTHOR	Information about who contributed to the development of the topic

Search for man Pages by Keyword

Use the `man` command `-k` option (equivalent to the `apropos` command) to search for a keyword in `man` page titles and descriptions. As a result, the keyword search displays a list of keyword-matching `man` page topics with section numbers. For example, the following command searches for `man` pages with the word `passwd`:

```
[user@host ~]$ man -k passwd
chgpaswd (8)      - update group passwords in batch mode
chpasswd (8)      - update passwords in batch mode
fgetpwent_r (3)   - get passwd file entry reentrantly
getpwent_r (3)    - get passwd file entry reentrantly
...
passwd (1)        - update user's authentication tokens
passwd (1openssl) - OpenSSL application commands
passwd (5)        - password file
passwd2des (3)    - RFS password encryption
...
```

The `man` command `-K` (uppercase) option searches for the keyword in the full-text page, not only in the titles and descriptions. A full-text search uses greater system resources and takes more time.

With the full-text page search, the `man` command displays the first page with a match. Press `Q` to exit this first page, and the `man` command displays the next page.

In this example, `man` displays each match, and you can view or skip each one.

```
[user@host ~]# man -K passwd
--Man-- next: cut(1p) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]
Ctrl-D
--Man-- next: logname(1p) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]
Ctrl-D
--Man-- next: sort(1p) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]
Ctrl-D
--Man-- next: xargs(1) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]
Ctrl-D
--Man-- next: chage(1) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]
Ctrl-C
```



Note

Keyword searches rely on an index that is generated by the `mandb(8)` command, which must be run as `root`.

The `man-db-cache-update` service automatically runs the `mandb` command when installing any package with man pages.



References

`man(1)`, `mandb(8)`, `man-pages(7)`, `less(1)`, `intro(1)`, `intro(2)`, `intro(5)`, `intro(7)`, and `intro(8)` man pages

► Guided Exercise

Read Manual Pages

Practice finding relevant information by using `man` options and arguments.

Outcomes

- Use the `man` Linux manual system and find useful information by searching and browsing.

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 help-manual
```

Instructions

- 1. On `workstation`, view the `gedit` man page. View the options for editing a specific file by using `gedit` from the command line.
Use one of the options from the `gedit` man page to open the `/home/student/manual` file by using `gedit` with the cursor at the end of the file.

- 1.1. View the `gedit` man page.

```
[student@workstation ~]$ man gedit
GEDIT(1)      General Commands Manual      GEDIT(1)
NAME
    gedit - text editor for the GNOME Desktop

SYNOPSIS
    gedit [OPTION...] [FILE...] [+LINE[:COLUMN]]
    gedit [OPTION...] -
...output omitted...
```

- 1.2. In the `gedit` man page, learn the options for editing a specific file from the command line.

OPTIONS

...output omitted...

FILE Specifies the file to open when gedit starts.

...output omitted...

`LINE` For the first file, go to the line specified by LINE (do not insert a space between the " sign and the number). If LINE is missing, go to the last line.

...output omitted...

Press q to quit the man page.

- 1.3. Use the `gedit`` command to open the ``manual`` file. The missing line number next to the ``` option opens a file that is passed as an argument with the cursor at the end of the last line.

```
[student@workstation ~]$ gedit + manual
this is the first line
the quick brown fox just came over to greet the lazy poodle!
```

Confirm that the file is opened with the cursor at the end of the last line in the file.
Press `Ctrl+q` to close the application.

- ▶ 2. Read the `su(1)` man page.

If you omit the `user` argument, then the `su` command assumes that the user is `root`. If the `su` command is followed by a single dash (`-`), then it starts a child login shell. Without the dash, the `su` command creates a non-login child shell that matches the user's current environment.

```
[student@workstation ~]$ man 1 su
SU(1)          User Commands          SU(1)
NAME
    su - run a command with substitute user and group ID

SYNOPSIS
    su [options] [-] [user [argument...]]

DESCRIPTION
    su allows to run commands with a substitute user and group ID.

    When called with no user specified, su defaults to running an interactive
    shell as root.
...output omitted...
OPTIONS
...output omitted...
    -, -l, --login
        Start the shell as a login shell with an environment similar to a real
    login.
...output omitted...
```

**Note**

Comma-separated options on a single line, such as `-`, `-l`, and `--login`, all result in the same behavior.

Press `q` to quit the man page.

- ▶ 3. The `man` command also has its own manual pages. Open the `man(1)` command manual page.

```
[student@workstation ~]$ man man
MAN(1)          Manual pager utils          MAN(1)

NAME
  man - an interface to the on-line reference manuals
...output omitted...
DESCRIPTION
  man is the system's manual pager. Each page argument given to man is
  normally the name of a program, utility or function. The manual page
  associated with each of these arguments is then found and displayed.
  A section, if provided, will direct man to look only in that section
  of the manual.
...output omitted...
```

Press `q` to quit the man page.

- ▶ 4. The `/usr/share/man` directory contains all man pages. Locate the binary, source, and manual pages for the `passwd` utility by using the `whereis` command. Verify that the `/usr/share/man` directory contains the man pages for the `passwd` utility.

```
[student@workstation ~]$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man1/passwd.1ossl.gz /usr/
share/man/man1/passwd.1.gz /usr/share/man/man5/passwd.5.gz
```

- ▶ 5. Use the `man -k zip` command to list the man page with detailed information about a ZIP archive.

```
[student@workstation ~]$ man -k zip
...output omitted...
zipinfo (1) - list detailed information about a ZIP archive
zipnote (1) - write the comments in zipfile to stdout, edit comments and
  rename files in zipfile
zipsplit (1) - split a zipfile into smaller zipfiles
```

- ▶ 6. Use the `man -k boot` command to list the man page with a list of parameters that can be passed to the kernel at boot time.

```
[student@workstation ~]$ man -k boot
binfmt.d (5)          - Configure additional binary formats for executables at boot
bootparam (7)        - introduction to boot time parameters of the Linux kernel
bootup (7)           - System bootup process
...output omitted...
```

- ▶ 7. Use the `man -k ext4` command to find the command to tune ext4 file-system parameters.

```
[student@workstation ~]$ man -k ext4
...output omitted...
resize2fs (8)        - ext2/ext3/ext4 file system resizer
tune2fs (8)          - adjust tunable filesystem parameters on ext2/ext3/ext4
                     filesystems
```

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 help-manual
```

► Lab

Get Help in Red Hat Enterprise Linux

Look up documentation to help you to complete tasks.

Outcomes

- Locate relevant information for commands by searching man pages.
- Learn new options for the most common documentation commands.
- Use appropriate tools to view and print documentation and other non-text formatted 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 help-review
```

Instructions

1. On the `workstation` machine, determine how to prepare a man page for printing. Specifically, find which is the default format or rendering language for printing.
2. Create a PostScript-formatted output file of the `passwd` man page. Call the file `passwd.ps` and place it in the `student` user's home directory. Determine the file format. Inspect the contents of the `passwd.ps` file by using the `less` command.



Note

Because you need to save the output of the `man` command to a file, you can use the `>` symbol, which redirects the standard output to a file.

As an example, the following command lists the home directory's regular file names into a file.

```
[student@workstation ~]$ ls > /tmp/my-file-names
```

This command is taught in more detail in a following chapter.

3. By using the man pages, find which commands you can use for viewing and printing PostScript files.
4. Learn how to use the `evince(1)` viewer in preview mode. Also, determine how to open a document to start on a specific page. Open your PostScript file by using `evince` three times: first by using the default mode, and then with the preview mode option, and finally to start at page 3. Close your document file when you finish.

5. By using the man pages, research `lp(1)` to determine how to print any document to start on a specific page. Without entering any commands (in the absence of printers), learn the syntax, in one command, to print only pages 2 and 3 of your PostScript file.
6. Use the Firefox browser to open the system's man page directory (`/usr/share/doc`) and browse to the `man-db` package subdirectory. View the provided manuals. After you finish reviewing the `man-db` manuals, locate and browse to the `kexec-tools` package subdirectory. View the `kexec-kdump-howto.txt` file, which describes important system configurations that are stored in the `/etc/sysconfig` 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 help-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 help-review
```

► Solution

Get Help in Red Hat Enterprise Linux

Look up documentation to help you to complete tasks.

Outcomes

- Locate relevant information for commands by searching man pages.
- Learn new options for the most common documentation commands.
- Use appropriate tools to view and print documentation and other non-text formatted 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 help-review
```

Instructions

1. On the `workstation` machine, determine how to prepare a man page for printing. Specifically, find which is the default format or rendering language for printing.
 - 1.1. Use the `man man` command to determine how to prepare a man page for printing.

```
[student@workstation ~]$ man man
...output omitted...
man -t bash | lpr -Pps
Format the manual page for bash into the default troff or groff format
and pipe it to the printer named ps. The default output for groff
is usually PostScript. man --help should advise as to which processor
is bound to the -t option.
...output omitted...
```

Press `q` to quit the man page.



Note

The `man` command `-t` option prepares a man page for printing, using by default PostScript.

2. Create a PostScript-formatted output file of the `passwd` man page. Call the file `passwd.ps` and place it in the `student` user's home directory. Determine the file format. Inspect the contents of the `passwd.ps` file by using the `less` command.

**Note**

Because you need to save the output of the `man` command to a file, you can use the `>` symbol, which redirects the standard output to a file.

As an example, the following command lists the home directory's regular file names into a file.

```
[student@workstation ~]$ ls > /tmp/my-file-names
```

This command is taught in more detail in a following chapter.

- 2.1. Use the `man -t` command to create a formatted file of the `passwd` man page.

```
[student@workstation ~]$ man -t passwd > passwd.ps
[student@workstation ~]$ ls -al
...output omitted...
-rw-r--r--. 1 student student 20168 Mar  8 09:02 passwd.ps
...output omitted...
```

- 2.2. Use the `file` command to determine the file format.

```
[student@workstation ~]$ file /home/student/passwd.ps
/home/student/passwd.ps: PostScript document text conforming DSC level 3.0
```

- 2.3. Use the `less` command to view the `/home/student/passwd.ps` file.

```
[student@workstation ~]$ less /home/student/passwd.ps
%!PS-Adobe-3.0
%%Creator: groff version 1.22.3
%%CreationDate: Tue Feb 26 11:14:40 2019
%%DocumentNeededResources: font Times-Roman
%%+ font Times-Bold
%%+ font Times-Italic
%%+ font Symbol
%%DocumentSuppliedResources: procset grops 1.22 3
...output omitted...
```

**Note**

The output of `file` states that the file is in PostScript format, and you can confirm it by viewing the file's contents. Notice the header lines of PostScript information. Use `q` to quit the `less` command.

3. By using the man pages, find which commands you can use for viewing and printing PostScript files.
 - 3.1. Search in the man pages for information about PostScript files. Use the `-k` option for this purpose.

```
[student@workstation ~]# man -k postscript viewer
enscript (1)          - convert text files to PostScript, HTML, RTF, ANSI, and
                        overstrikes
eps2eps (1)          - Ghostscript PostScript "distiller"
evince (1)           - GNOME document viewer
evince-previewer (1) - show a printing preview of PostScript and PDF documents
evince-thumbnailer (1) - create png thumbnails from PostScript and PDF documents
gcm-viewer (1)       - GNOME Color Manager Profile Viewer Tool
...output omitted...
```

**Note**

Using multiple words with the `-k` option finds man pages that match any word; such as "postscript" or "viewer" in their descriptions. Notice the `evince(1)` commands in the output.

4. Learn how to use the `evince(1)` viewer in preview mode. Also, determine how to open a document to start on a specific page. Open your PostScript file by using `evince` three times: first by using the default mode, and then with the preview mode option, and finally to start at page 3. Close your document file when you finish.

- 4.1. Use the `man evince` command to learn how to use the viewer in preview mode.

```
[student@workstation ~]$ man evince
...output omitted...
-i, --page-index=NUMBER
        Open the document on the page with the specified page index (this is
        the exact page number, not a page label).
...output omitted...
-w, --preview
        Run evince as a previewer.
...output omitted...
```

Press `q` to quit the man page.

**Note**

The `-w` (or `--preview`) option opens `evince` in preview mode. The `-i` option opens `evince` at the specified starting page.

- 4.2. Use the `evince` command to open the `/home/student/passwd.ps` file.

```
[student@workstation ~]$ evince /home/student/passwd.ps
```

- 4.3. Use the `evince -w /home/student/passwd.ps` command to open the file in preview mode.

```
[student@workstation ~]$ evince -w /home/student/passwd.ps
```

- 4.4. Use the `evince -i 3 /home/student/passwd.ps` command to open the file at page 3.

```
[student@workstation ~]$ evince -i 3 /home/student/passwd.ps
```

**Note**

Whereas the normal `evince` mode supports full-screen and presentation-style viewing, the `evince` preview mode is useful for quick browsing and printing. Notice the **print icon** at the top.

5. By using the man pages, research `lp(1)` to determine how to print any document to start on a specific page. Without entering any commands (in the absence of printers), learn the syntax, in one command, to print only pages 2 and 3 of your PostScript file.

5.1. Use the `man lp` command to determine how to print specific pages of a document.

```
[student@workstation ~]$ man lp
```

```
...output omitted...
```

```
-P page-list
```

Specifies which pages to print in the document. The list can contain a list of numbers and ranges (-) separated by commas, e.g., "1,3-5, 16". The page numbers refer to the output pages and not the document's original pages - options like "number-up" can affect the numbering of the pages.

```
...output omitted...
```

Press `q` to quit the man page.

**Note**

From `lp(1)`, you learn that the `-P` option specifies the page list to print in the document. The `lp` command spools to the *default* printer, and sends only the page range to start on 2 and to end on 3. Therefore, one valid answer is `lp passwd.ps -P 2-3`.

6. Use the Firefox browser to open the system's man page directory (`/usr/share/doc`) and browse to the `man-db` package subdirectory. View the provided manuals. After you finish reviewing the `man-db` manuals, locate and browse to the `kexec-tools` package subdirectory. View the `kexec-kdump-howto.txt` file, which describes important system configurations that are stored in the `/etc/sysconfig` directory.

6.1. Use `firefox /usr/share/doc` to view system documentation. Browse the `man-db` subdirectory. Click the manuals to view them.

```
[student@workstation ~]$ firefox /usr/share/doc
```

The `firefox` command might return output that is unrelated to the Satellite console.

**Note**

You can create bookmarks for any frequently used directory. After browsing the `man-db` directory, click to open and view the text version of the manual, and then close it. Click to open the PostScript version. As observed earlier, `evince` is the system's default viewer for PostScript and PDF documents. When finished, close the `evince` viewer.

Index of file:///usr/share/doc/man-db/

Up to higher level directory

Name	Size	Last Modified
File: ChangeLog	167 KB	6/22/20 16:22:20 EDT
File: NEWS	70 KB	6/22/20 16:06:52 EDT
File: README	12 KB	1/1/20 10:11:59 EST
man-db-manual.ps	129 KB	8/10/21 07:52:12 EDT
man-db-manual.txt	69 KB	8/10/21 07:52:12 EDT

- 6.2. In the Firefox browser, locate the `kexec-tools` package subdirectory and view the `kexec-kdump-howto.txt` file. This file describes important system configurations that are stored in the `/etc/sysconfig` directory.
- Close the document and Firefox when finished.

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 help-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 help-review
```

Summary

- Use the `man` command to view man pages and to display information about components of a Linux system, such as files, commands, and functions.
- By convention, to refer to a man page, the name of a page is followed by its section number in parentheses.
- You can use regular expressions to search content in man pages.

Chapter 5

Create, View, and Edit Text Files

Goal

Create, view, and edit text files from command output or in a text editor.

Objectives

- Save output or errors to a file with shell redirection, and process command output through multiple command-line programs with pipes.
- Create and edit text files from the command line with the `vim` editor.
- Set shell variables to run commands, and edit Bash startup scripts to set shell and environment variables to modify the behavior of the shell and programs that are run from the shell.

Sections

- Redirect Output to a File or Program (and Quiz)
- Edit Text Files from the Shell Prompt (and Guided Exercise)
- Change the Shell Environment (and Guided Exercise)

Lab

- Create, View, and Edit Text Files

Redirect Output to a File or Program

Objectives

- Save output or errors to a file with shell redirection, and process command output through multiple command-line programs with pipes.

Standard Input, Standard Output, and Standard Error

A running program, or *process*, reads input and writes output. When you run a command from the shell prompt, it normally reads its input from the keyboard and sends its output to the terminal window.

A process uses numbered channels called *file descriptors* to get input and send output. All processes start with at least three file descriptors. *Standard input* (channel 0) reads input from the keyboard. *Standard output* (channel 1) sends normal output to the terminal. *Standard error* (channel 2) sends error messages to the terminal.

If a program opens separate connections to other files, then it might use higher-numbered file descriptors.

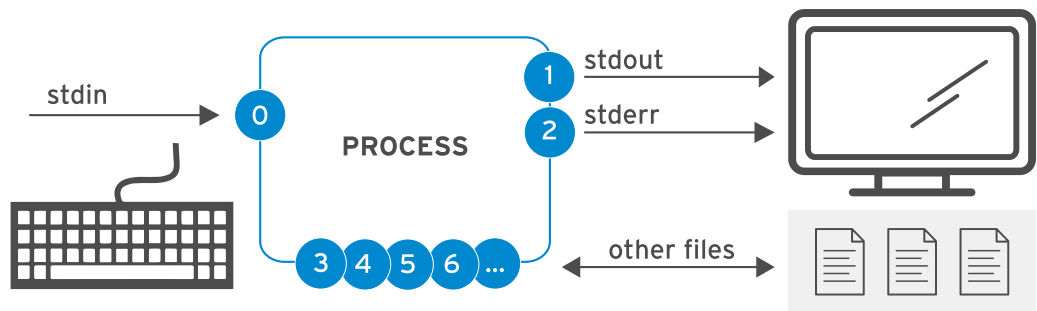


Figure 5.1: Process I/O channels (file descriptors)

The next table summarizes the information about the file descriptors:

Channels (File Descriptors)

Number	Channel name	Description	Default connection	Usage
0	stdin	Standard input	Keyboard	Read only
1	stdout	Standard output	Terminal	Write only
2	stderr	Standard error	Terminal	Write only
3+	<i>filename</i>	Other files	None	Read, write, or both

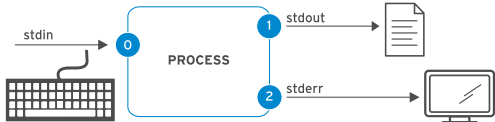
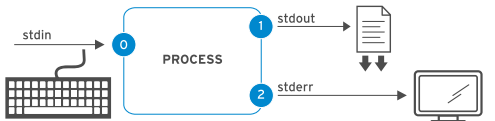
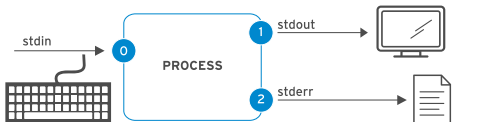
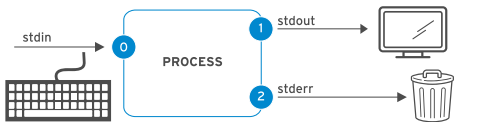
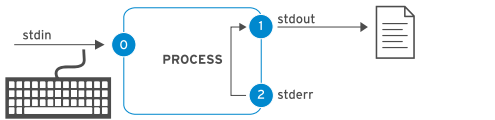
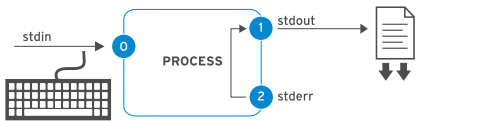
Redirect Output to a File

The Input/Output (I/O) redirection changes how the process gets its input or output. Instead of getting input from the keyboard, or sending output and errors to the terminal, the process can read from or write to files. With redirection, you can save the messages to a file instead of displaying the output on the terminal. Alternatively, you can use redirection to discard output or errors, so they are not displayed on the terminal or saved.

You can redirect a process `stdout` to suppress the process output from appearing on the terminal. If you redirect `stdout` to a file and the file does not exist, then the file is created. If the file does exist and the redirection does not append to the file, then the redirection overwrites the file's contents. To discard the output of a process, you can redirect to the empty `/dev/null` special file that discards channel output that is redirected to it.

As viewed in the following table, redirecting *only* `stdout` does not suppress displaying `stderr` error messages on the terminal.

Output Redirection Operators

Usage	Explanation	Visual aid
<code>> file</code>	Redirect <code>stdout</code> to overwrite a file.	
<code>>> file</code>	Redirect <code>stdout</code> to append to a file.	
<code>2> file</code>	Redirect <code>stderr</code> to overwrite a file.	
<code>2> /dev/null</code>	Discard <code>stderr</code> error messages by redirecting them to <code>/dev/null</code> .	
<code>> file 2>&1</code> <code>&> file</code>	Redirect <code>stdout</code> and <code>stderr</code> to overwrite the same file.	
<code>>> file 2>&1</code> <code>&>> file</code>	Redirect <code>stdout</code> and <code>stderr</code> to append to the same file.	

**Important**

The order of redirection operations is important. The following sequence redirects standard output to the `output.log` file and then redirects standard error messages to the same place as standard output (`output.log`).

```
> output.log 2>&1
```

The next sequence redirects in the opposite order. This sequence redirects standard error messages to the default place for standard output (the terminal window, so no change) and *then* redirects only standard output to the `output.log` file.

```
2>&1 > output.log
```

For this reason, some people prefer to use the merging redirection operators:

- `&> output.log` instead of `> output.log 2>&1`
- `&>> output.log` instead of `>> output.log 2>&1` (in Bash 4 or RHEL 6 and later)

However, system administrators and programmers prefer to avoid the newer merging redirection operators when using alternative shells to `bash` (known as Bourne-compatible shells) for scripting commands. These new redirection operators are not standardized or implemented in those shells, and have other limitations.

Examples for Output Redirection

Simplify many routine administration tasks by using redirection. Use the previous table to assist, and consider the following examples:

Save a time stamp in the `/tmp/saved-timestamp` file for later reference.

```
[user@host ~]$ date > /tmp/saved-timestamp
```

Copy the last 100 lines from the `/var/log/secure` file to the `/tmp/last-100-log-secure` file.

```
[user@host ~]$ tail -n 100 /var/log/secure > /tmp/last-100-log-secure
```

Concatenate all four `step` files into one file in the `tmp` directory.

```
[user@host ~]$ cat step1.sh step2.log step3 step4 > /tmp/all-four-steps-in-one
```

List the home directory's hidden and regular file names, and save the output to the `my-file-names` file.

```
[user@host ~]$ ls -a > my-file-names
```

Append a line to the existing `/tmp/many-lines-of-information` file.

```
[user@host ~]$ echo "new line of information" >> /tmp/many-lines-of-information
```

The next few commands generate error messages because some system directories are inaccessible to normal users. Observe the error message redirection.

Redirect errors from the `find` command to the `/tmp/errors` file when viewing normal command output on the terminal.

```
[user@host ~]$ find /etc -name passwd 2> /tmp/errors
```

Save process output to the `/tmp/output` file and error messages to the `/tmp/errors` file.

```
[user@host ~]$ find /etc -name passwd > /tmp/output 2> /tmp/errors
```

Save process output to the `/tmp/output` file and discard error messages.

```
[user@host ~]$ find /etc -name passwd > /tmp/output 2> /dev/null
```

Store output and generated errors together to the `/tmp/all-message-output` file.

```
[user@host ~]$ find /etc -name passwd &> /tmp/all-message-output
```

Append output and generated errors to the `/tmp/all-message-output` file.

```
[user@host ~]$ find /etc -name passwd >> /tmp/all-message-output 2>&1
```

Construct Pipelines

A *pipeline* is a sequence of one or more commands that are separated by the vertical bar character (`|`). A pipeline connects the standard output of the first command to the standard input of the next command.

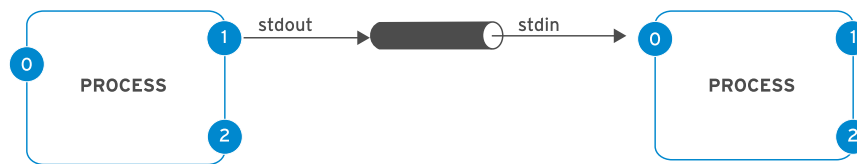


Figure 5.8: Process I/O piping

Use pipelines to manipulate and format the output of a process by other processes before it is output to the terminal. Imagine that data "flows" through the pipeline from one process to another, and is altered by each command in the pipeline that it flows through.



Note

Pipelines and I/O redirection both manipulate standard output and standard input. Pipelines send the standard output from one process to the standard input of another process. Redirection sends standard output to files, or gets standard input from files.

Pipeline Examples

The following list shows some pipeline examples:

Redirect the output of the `ls` command to the `less` command to display it on the terminal one screen at a time.

```
[user@host ~]$ ls -l /usr/bin | less
```

Redirect the output of the `ls` command to the `wc -l` command, which counts the number of received lines from `ls` and prints that value to the terminal.

```
[user@host ~]$ ls | wc -l
```

Redirect the output of the `ls -t` command to the `head` command to display the first 10 lines, with the final result redirected to the `/tmp/first-ten-changed-files` file.

```
[user@host ~]$ ls -t | head -n 10 > /tmp/first-ten-changed-files
```

Pipelines, Redirection, and Appending to a File

When you combine redirection with a pipeline, the shell sets up the entire pipeline first, and then it redirects the input/output. If you use output redirection in the *middle* of a pipeline, then the output goes to the file and not to the next command in the pipeline.

In the next example, the output of the `ls` command goes to the `/tmp/saved-output` file, and the `less` command displays nothing on the terminal.

```
[user@host ~]$ ls > /tmp/saved-output | less
```

The `tee` command overcomes this limitation. In a pipeline, `tee` copies its standard input to its standard output and also redirects its standard output to the files that are given as arguments to the command. If you imagine data as water that flows through a pipeline, then you can visualize `tee` as a "T" joint in the pipe that directs output in two directions.

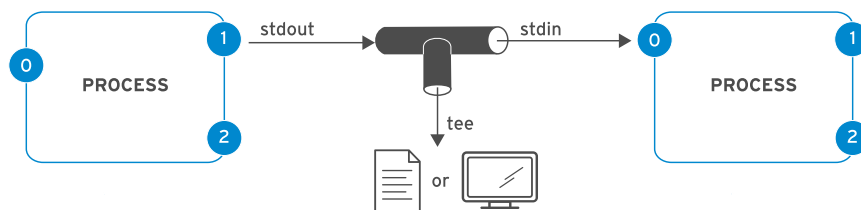


Figure 5.9: Process I/O piping with tee

Pipeline Examples with the tee Command

The next example redirects the output of the `ls` command to the `/tmp/saved-output` file and passes it to the `less` command, so it is displayed on the terminal one screen at a time.

```
[user@host ~]$ ls -l | tee /tmp/saved-output | less
```

If you use the `tee` command at the end of a pipeline, then the terminal shows the output of the commands in the pipeline and saves it to a file at the same time.

```
[user@host ~]$ ls -t | head -n 10 | tee /tmp/ten-last-changed-files
```

Use the `tee` command `-a` option to append the content to a file instead of overwriting it.

```
[user@host ~]$ ls -l | tee -a /tmp/append-files
```



Important

You can redirect standard error through a pipeline, but you cannot use the merging redirection operators (`&>` and `&>>`). The following example is the correct way to redirect both standard output and standard error through a pipeline:

```
[user@host ~]$ find / -name passwd 2>&1 | less
```



References

`info bash` (*The GNU Bash Reference Manual*)

- Section 3.2.3: Pipelines
- Section 3.6: Redirections

`info coreutils 'tee invocation'` (*The GNU coreutils Manual*)

- Section 17.1: Redirect output to multiple files or processes

`bash(1)`, `cat(1)`, `head(1)`, `less(1)`, `mail(1)`, `tee(1)`, `tty(1)`, `wc(1)` man pages

► Quiz

Redirect Output to a File or Program

Choose the correct answer to the following questions:

- 1. Which output redirection operator displays output to a terminal and discards all error messages?
 - a. `&> file`
 - b. `2> &> file`
 - c. `2> /dev/null`
 - d. `1> /dev/null`
- 2. Which output redirection operator sends output to a file and sends errors to a different file?
 - a. `> file 2> file2`
 - b. `> file 1> file2`
 - c. `> file &2> file2`
 - d. `| tee file`
- 3. Which output redirection operator sends both output and errors to a file, and creates it or overwrites its contents?
 - a. `| tee file`
 - b. `2 &> file`
 - c. `1 &> file`
 - d. `&> file`
- 4. Which output redirection operator sends output and errors to the same file and preserves the file content if it exists?
 - a. `> file 2> file2`
 - b. `&> file`
 - c. `>> file 2>&1`
 - d. `>> file 1>&1`
- 5. Which output redirection operator discards all messages that are normally sent to the terminal?
 - a. `> file 2> file2`
 - b. `&> /dev/null`
 - c. `&> /dev/null 2> file`
 - d. `&> file`

- **6. Which output redirection operator sends output to both the screen and a file at the same time?**
- a. `&> /dev/null`
 - b. `> file 2> file2`
 - c. `| tee file`
 - d. `| < file`
- **7. Which output redirection operator saves output to a file and discards all error messages?**
- a. `&> file`
 - b. `| tee file 2> /dev/null`
 - c. `> file 1> /dev/null`
 - d. `> file 2> /dev/null`

► Solution

Redirect Output to a File or Program

Choose the correct answer to the following questions:

- 1. Which output redirection operator displays output to a terminal and discards all error messages?
 - a. `&> file`
 - b. `2> &> file`
 - c. `2> /dev/null`
 - d. `1> /dev/null`
- 2. Which output redirection operator sends output to a file and sends errors to a different file?
 - a. `> file 2> file2`
 - b. `> file 1> file2`
 - c. `> file &2> file2`
 - d. `| tee file`
- 3. Which output redirection operator sends both output and errors to a file, and creates it or overwrites its contents?
 - a. `| tee file`
 - b. `2 &> file`
 - c. `1 &> file`
 - d. `&> file`
- 4. Which output redirection operator sends output and errors to the same file and preserves the file content if it exists?
 - a. `> file 2> file2`
 - b. `&> file`
 - c. `>> file 2>&1`
 - d. `>> file 1>&1`
- 5. Which output redirection operator discards all messages that are normally sent to the terminal?
 - a. `> file 2> file2`
 - b. `&> /dev/null`
 - c. `&> /dev/null 2> file`
 - d. `&> file`

- 6. Which output redirection operator sends output to both the screen and a file at the same time?
- a. `&> /dev/null`
 - b. `> file 2> file2`
 - c. `| tee file`
 - d. `| < file`
- 7. Which output redirection operator saves output to a file and discards all error messages?
- a. `&> file`
 - b. `| tee file 2> /dev/null`
 - c. `> file 1> /dev/null`
 - d. `> file 2> /dev/null`

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 edited with any text editor.

Vim is an improved version of the `vi` editor, which is distributed with Linux and UNIX systems. Vim is a highly configurable and efficient editor that provides 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 to 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*. As a Vim user, always verify the current mode, because the effect of keystrokes varies between modes.

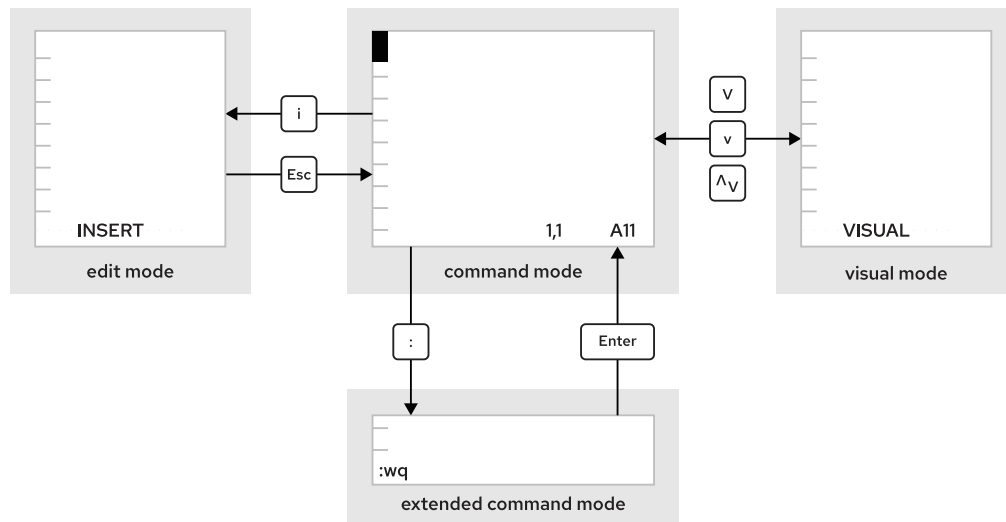


Figure 5.10: Moving between Vim modes

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 that you 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 various visual modes in 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

First, take the time to familiarize yourself with the basic Vim capabilities. Then, expand your Vim vocabulary by learning more Vim keystrokes.

The exercise for this section uses the **vimtutor** command. This tutorial, from the **vim-enhanced** package, 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 for 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

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 `i` (insert) key is pressed first.
- ▶ **6.** In the `vimtutor` window, perform *Lesson 1.5*.
The previous lecture taught only the `i` command to enter edit mode. This lesson demonstrates other available keystrokes to change the cursor placement in 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
```

Change the Shell Environment

Objectives

- Set shell variables to run commands, and edit Bash startup scripts to set shell and environment variables to modify the behavior of the shell and programs that are run from the shell.

Shell Variable Usage

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

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

Assign Values to Variables

Assign a value to a shell variable with the following syntax:

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

Variable names can contain uppercase or lowercase letters, digits, and the underscore character (`_`). For example, the following commands set shell variables:

```
[user@host ~]$ COUNT=40
[user@host ~]$ first_name=John
[user@host ~]$ file1=/tmp/abc
[user@host ~]$ _ID=RH123
```

Remember, this change affects only the shell that you run the command in, not any other shells that you might be running on that server.

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

```
[user@host ~]$ set | less
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:complete_fullquote:expand_aliases:extglob:extquote\
:force_ignores:globasciiranges:histappend:interactive_comments:login_shell:prog\
omp:promptvars:sourcepath
BASHRC_SOURCED=Y
...output omitted...
```


Retrieve Values with Variable Expansion

You can use *variable expansion* to refer to the value of a variable that you set. To use variable expansion, precede the name of the variable with a dollar sign (\$). In the following examples, the variable expansion occurs first and then the `echo` command prints the rest of the command line that is entered.

For example, the following command sets the variable `COUNT` to 40.

```
[user@host ~]$ COUNT=40
```

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

```
[user@host ~]$ echo COUNT
COUNT
```

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

```
[user@host ~]$ echo $COUNT
40
```

You can also use a variable to refer to a long file name for multiple commands.

```
[user@host ~]$ file1=/tmp/tmp.z9pXW0HqcC
[user@host ~]$ ls -l $file1
-rw-----. 1 student student 1452 Jan 22 14:39 /tmp/tmp.z9pXW0HqcC
[user@host ~]$ rm $file1
[user@host ~]$ ls -l $file1
total 0
```



Important

You can always use curly braces in variable expansion, although they are often unnecessary.

In the following example, the `echo` command tries to expand the nonexistent variable `COUNTx`, but returns nothing. The command does not report any errors either.

```
[user@host ~]$ echo Repeat $COUNTx
Repeat
```

If any trailing characters are next to a variable name, then delimit the variable name with curly braces. In the following example, the `echo` command now expands the `COUNT` variable.

```
[user@host ~]$ echo Repeat ${COUNT}x
Repeat 40x
```

Configure Bash with Shell Variables

Some shell variables are set when Bash starts. You can modify them to adjust the shell's behavior.

For example, the `HISTFILE`, `HISTFILESIZE`, and `HISTTIMEFORMAT` shell variables affect the shell history and the `history` command. The `HISTFILE` variable specifies which file to save the shell history to, and defaults to the `~/.bash_history` file. The `HISTFILESIZE` variable specifies how many commands to save in that file from the history. The `HISTTIMEFORMAT` variable defines the time stamp format for every command in the history. This variable does not exist by default.

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

Another example is the `PS1` variable, which controls the appearance of the shell prompt. If you change this value, then it changes the appearance of your shell prompt. Various special character expansions that the prompt supports are listed in the "PROMPTING" section of the `bash(1)` man page.

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

Because the value that the `PS1` variable sets is a prompt, Red Hat recommends ending the prompt with a trailing space. Also, whenever a variable value contains some form of space, including a space, a tab, or a return, the value must be enclosed in either single or double quotation marks. Unexpected results might occur if the quotation marks are omitted. The previous `PS1` variable conforms to both the trailing space recommendation and the quotation marks rule.

Configure Programs with Environment Variables

The shell provides an *environment* for the programs that you run from that shell. Among other items, this environment includes information about the current working directory on the file system, the command-line options that are passed to the program, and the values of *environment variables*. The programs might use these environment variables to change their behavior or their default settings.

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

**Note**

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

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

```
[user@host ~]$ EDITOR=vim
[user@host ~]$ export EDITOR
```

You can set and export a variable in one step:

```
[user@host ~]$ export EDITOR=vim
```

Applications and sessions use these variables to determine their behavior. For example, the shell automatically sets the `HOME` variable to the file name of the user's home directory when it starts. You can use this variable to help programs to determine where to save files.

Another example is the `LANG` variable, which sets the locale encoding. This variable adjusts the preferred language for program output; the character set; the formatting of dates, numbers, and currency; and the sort order for programs. If it is set to `en_US.UTF-8`, then the locale uses US English with UTF-8 Unicode character encoding. If it is set, for example, to `fr_FR.UTF-8`, then it uses French UTF-8 Unicode encoding.

```
[user@host ~]$ date
Tue Jan 22 16:37:45 CST 2019
[user@host ~]$ export LANG=fr_FR.UTF-8
[user@host ~]$ date
mar. janv. 22 16:38:14 CST 2019
```

Another important environment variable is `PATH`. The `PATH` variable contains a list of colon-separated directories that contain programs:

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

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

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

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

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

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

Set the Default Text Editor

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

```
[user@host ~]$ export EDITOR=nano
```



Important

By convention, environment variables and shell variables that are automatically set by the shell have names with all uppercase characters. If you are setting your own variables, then you might want to use names with lowercase characters to prevent naming collisions.

Set Variables Automatically

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

The exact scripts that run depend on whether the shell is interactive or non-interactive, and a login or non-login shell. A user directly enters commands into an interactive shell, whereas a non-interactive shell, such as a script, runs in the background without user intervention. A login shell is invoked when a user logs in locally via the terminal or remotely via the SSH protocol. A non-login shell is invoked from an existing session, such as to open a terminal from the GNOME GUI.

For interactive login shells, the `/etc/profile` and `~/.bash_profile` files configure the Bash environment. The `/etc/profile` and `~/.bash_profile` files also source the `/etc/bashrc` and `~/.bashrc` files respectively. For interactive non-login shells, only the `/etc/bashrc` and `~/.bashrc` files configure the Bash environment. Whereas the `/etc/profile` and `/etc/bashrc` files apply to the whole system, the `~/.bash_profile` and `~/.bashrc` files are user-specific. Non-interactive shells invoke any files that the `BASH_ENV` variable defines. This variable is not defined by default.

To create a variable that is available to all of your interactive shells, edit the `~/.bashrc` file. To apply a variable only once after the user logs in, define it in the `~/.bash_profile` file.

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

```
# .bash_profile

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

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

**Note**

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

Bash Aliases

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

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

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

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

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

Unset and Unexport Variables and Aliases

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

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

[user@host ~]$
```

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

```
[user@host ~]$ export -n PS1
```

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

```
[user@host ~]$ unalias hello
```



References

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

► Guided Exercise

Change the Shell Environment

Use shell variables and variable expansion to run commands, and set an environment variable to adjust the default editor for new shells.

Outcomes

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

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 edit-bashconfig
```

Instructions

- 1. Change the `student` user's `PS1` shell variable to `[\u@\h \t \w]$` (remember to put the value of `PS1` in quotation marks and to include a trailing space after the dollar sign). This change adds the time to the prompt.

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

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

- 1.2. Use Vim to edit the `~/ .bashrc` configuration file.

```
[student@servera ~]$ vim ~/.bashrc
```

- 1.3. Add the `PS1` shell variable and its value to the `~/ .bashrc` file. Set the value of the shell variable, including a trailing space at the end, inside quotation marks.

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

- 1.4. Exit from `servera` and log in again by using the `ssh` command to update the command prompt, or execute the `~/ .bashrc` file by using the `source ~/.bashrc` command.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera 14:45:05 ~]$
```

- ▶ 2. Assign a value to a local shell variable. Variable names can contain uppercase or lowercase letters, digits, and the underscore character. Retrieve the variable value.

- 2.1. Create a variable called `file` with a value of `tmp.zdkei083`. The `tmp.zdkei083` file exists in the student home directory.

```
[student@servera 14:47:05 ~]$ file=tmp.zdkei083
```

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

```
[student@servera 14:48:35 ~]$ echo $file
tmp.zdkei083
```

- 2.3. Use the `file` variable name and the `ls -l` command to list the `tmp.zdkei083` file. Use the `rm` command and the `file` variable name to delete the `tmp.zdkei083` file. Verify that the file is deleted.

```
[student@servera 14:59:07 ~]$ ls -l $file
-rw-r--r--. 1 student student 0 Jan 23 14:59 tmp.zdkei083
[student@servera 14:59:10 ~]$ rm $file
[student@servera 14:59:15 ~]$ ls -l $file
ls: cannot access 'tmp.zdkei083': No such file or directory
```

- ▶ 3. Assign a value to the `EDITOR` variable. Use one command to assign the variable as an environment variable.

```
[student@servera 14:46:40 ~]$ export EDITOR=vim
[student@servera 14:46:55 ~]$ echo $EDITOR
vim
```

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

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

Finish

On the `workstation` machine, 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-bashconfig
```

► Lab

Create, View, and Edit Text Files

Edit a text file with the `vim` editor.

Outcomes

- Use Vim to edit files.
- Use Vim visual mode to simplify editing large 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 edit-review
```

Instructions

1. On `workstation`, create the `lab_file` shell variable and assign `editing_final_lab.txt` as the value. List the contents of the `student` home directory, including hidden directories and files, as well as permissions, file sizes, and modification times, and redirect the output to the `editing_final_lab.txt` file by using the shell variable.
2. Use Vim to edit the `editing_final_lab.txt` file. Use the `lab_file` shell variable.
3. Enter the line-based visual mode of Vim. Your screen output might differ from these examples. Remove the first three lines of the `editing_final_lab.txt` file.
4. Enter the visual mode of Vim. Remove the last seven characters from the first column on the first line. Preserve only the first four characters of the first column.
Use the arrow keys to position the cursor at the last character of the first column on the first line. Delete the selection by typing `x`.
5. Enter the visual block mode of Vim. Repeat the operation of the previous step, but this time select from the second to the last line. Preserve only the first four characters of the first column.
6. Enter the visual block mode of Vim and remove the fourth column of the file.
7. Enter the visual block mode of Vim to remove the time column, to leave the month and day columns on all lines.
8. Enter the visual line mode of Vim and remove the rows that contain the `Desktop` and `Public` strings.
9. Save your changes and exit the file.
10. Back up the `editing_final_lab.txt` file and append the date (in seconds) at the end of the file name preceded with an underscore (`_`) character. Use the `lab_file` shell variable.

11. Append a dashed line to the `editing_final_lab.txt` file. The dashed line should contain 12 dash (-) characters for this lab to be graded correctly. Use the `lab_file` shell variable.
12. List the content of the `Documents` directory, and append the output to the `editing_final_lab.txt` file, and display the output in the terminal. Use the `tee` command and the `lab_file` shell variable.
13. Confirm that the directory listing is at the bottom of the lab file. Use the `lab_file` shell variable.

Evaluation

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

```
[student@workstation ~]$ lab grade edit-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 edit-review
```

► Solution

Create, View, and Edit Text Files

Edit a text file with the `vim` editor.

Outcomes

- Use Vim to edit files.
- Use Vim visual mode to simplify editing large 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 edit-review
```

Instructions

1. On `workstation`, create the `lab_file` shell variable and assign `editing_final_lab.txt` as the value. List the contents of the `student` home directory, including hidden directories and files, as well as permissions, file sizes, and modification times, and redirect the output to the `editing_final_lab.txt` file by using the shell variable.

On the `workstation` machine, create the `lab_file` shell variable and assign a value of `editing_final_lab.txt`. Use the `ls -al` command in the `student` home directory and redirect the output to the `editing_final_lab.txt` file.

```
[student@workstation ~]$ lab_file=editing_final_lab.txt
[student@workstation ~]$ ls -al > $lab_file
```

2. Use Vim to edit the `editing_final_lab.txt` file. Use the `lab_file` shell variable.

```
[student@workstation ~]$ vim $lab_file
```

3. Enter the line-based visual mode of Vim. Your screen output might differ from these examples. Remove the first three lines of the `editing_final_lab.txt` file.
Use the arrow keys to position the cursor at the first character in the first line. Enter the line-based visual mode with `Shift+V`. Move down by using the down arrow key twice to select the first three lines. Delete the lines by typing `x`.

```

student@workstation:~ -- /usr/bin/vim editing_final_lab.txt
total 36
drwx----- 17 student student 4096 Mar 9 01:25 .
drwxr-xr-x 5 root root 33 Mar 8 22:32 ..
drwxr-xr-x 3 student student 17 Mar 4 03:36 ansible
drwx----- 1 student student 378 Mar 9 01:21 .bash_history
drwxr-xr-x 1 student student 18 Nov 5 07:46 .bash_logout
drwxr-xr-x 1 student student 141 Nov 5 07:46 .bash_profile
drwxr-xr-x 1 student student 492 Nov 5 07:46 .bashrc
drwxr-xr-x 9 student student 4096 Mar 8 22:37 .cache
drwxr-xr-x 8 student student 4096 Mar 8 22:37 .config
drwxr-xr-x 2 student student 6 Mar 8 22:37 Desktop
drwxr-xr-x 2 student student 6 Mar 8 22:37 Documents
drwxr-xr-x 2 student student 6 Mar 8 22:37 Downloads
drwxr-xr-x 1 student student 0 Mar 8 22:39 editing_final_lab.txt
drwxr-xr-x 2 student student 25 Mar 4 03:37 .grading
drwxr-xr-x 4 student student 32 Mar 8 22:37 .local
drwxr-xr-x 2 student student 6 Mar 8 22:37 Music
drwxr-xr-x 2 student student 6 Mar 8 22:37 Pictures
drwxr-xr-x 2 student student 6 Mar 8 22:37 Public
drwx----- 2 student student 77 Mar 4 03:31 .ssh
drwxr-xr-x 2 student student 6 Mar 8 22:37 Templates
drwxr-xr-x 3 student student 18 Mar 4 03:36 .venv
drwxr-xr-x 2 student student 6 Mar 8 22:37 Videos
drwx----- 1 student student 5653 Mar 9 01:25 .viminfo
-- VISUAL LINE --
3 3,1 All

```

4. Enter the visual mode of Vim. Remove the last seven characters from the first column on the first line. Preserve only the first four characters of the first column.

Use the arrow keys to position the cursor at the fifth character of the first column on the first line. Enter the visual mode by typing `v`.

Use the arrow keys to position the cursor at the last character of the first column on the first line. Delete the selection by typing `x`.

```

student@workstation:~ -- /usr/bin/vim editing_final_lab.txt
drwxr-xr-x 3 student student 17 Mar 4 03:36 ansible
drwxr-xr-x 1 student student 18 Nov 5 07:46 .bash_logout
drwxr-xr-x 1 student student 141 Nov 5 07:46 .bash_profile
drwxr-xr-x 1 student student 492 Nov 5 07:46 .bashrc
drwxr-xr-x 9 student student 4096 Mar 8 22:37 .cache
drwxr-xr-x 8 student student 4096 Mar 8 22:37 .config
drwxr-xr-x 2 student student 6 Mar 8 22:37 Desktop
drwxr-xr-x 2 student student 6 Mar 8 22:37 Documents
drwxr-xr-x 2 student student 6 Mar 8 22:37 Downloads
drwxr-xr-x 1 student student 0 Mar 8 22:39 editing_final_lab.txt
drwxr-xr-x 2 student student 25 Mar 4 03:37 .grading
drwxr-xr-x 4 student student 32 Mar 8 22:37 .local
drwxr-xr-x 2 student student 6 Mar 8 22:37 Music
drwxr-xr-x 2 student student 6 Mar 8 22:37 Pictures
drwxr-xr-x 2 student student 6 Mar 8 22:37 Public
drwx----- 2 student student 77 Mar 4 03:31 .ssh
drwxr-xr-x 2 student student 6 Mar 8 22:37 Templates
drwxr-xr-x 3 student student 18 Mar 4 03:36 .venv
drwxr-xr-x 2 student student 6 Mar 8 22:37 Videos
-- VISUAL --
7 1,11 All

```

- Enter the visual block mode of Vim. Repeat the operation of the previous step, but this time select from the second to the last line. Preserve only the first four characters of the first column.

Use the arrow keys to position the cursor at the fifth character of the second line. Enter the visual mode by using the `Ctrl+V` control sequence. Use the arrow keys to position the cursor at the last character of the first column on the last line. Delete the selection by typing `x`.

```

student@workstation:~ -- /usr/bin/vim editing_final_lab.txt
drwx 3 student student 17 Mar 4 03:36 .ansible
-rw-r--r-- 1 student student 18 Nov 5 07:46 .bash_logout
-rw-r--r-- 1 student student 141 Nov 5 07:46 .bash_profile
-rw-r--r-- 1 student student 492 Nov 5 07:46 .bashrc
drwxr-xr-x 9 student student 4096 Mar 8 22:37 .cache
drwxr-xr-x 2 student student 4096 Mar 8 22:37 .config
drwxr-xr-x 2 student student 6 Mar 8 22:37 Desktop
drwxr-xr-x 2 student student 6 Mar 8 22:37 Documents
drwxr-xr-x 2 student student 6 Mar 8 22:37 Downloads
-rw-r--r-- 1 student student 0 Mar 8 22:39 editing_final_lab.txt
drwxr-xr-x 2 student student 25 Mar 4 03:37 .grading
drwxr-xr-x 4 student student 32 Mar 8 22:37 .local
drwxr-xr-x 2 student student 6 Mar 8 22:37 Music
drwxr-xr-x 2 student student 6 Mar 8 22:37 Pictures
drwxr-xr-x 2 student student 6 Mar 8 22:37 Public
drwx----- 2 student student 77 Mar 4 03:31 .ssh
drwxr-xr-x 2 student student 6 Mar 8 22:37 Templates
drwxr-xr-x 3 student student 18 Mar 4 03:36 .venv
drwxr-xr-x 2 student student 6 Mar 8 22:37 Videos
-- VISUAL BLOCK --
18x7 19,11 All

```

- Enter the visual block mode of Vim and remove the fourth column of the file. Use the arrow keys to position the cursor at the first character of the fourth column. Enter the visual block mode by using `Ctrl+V`. Use the arrow keys to position the cursor at the last character and row of the fourth column. Delete the selection by typing `x`.

```

student@workstation:~ -- /usr/bin/vim editing_final_lab.txt
drwx 3 student student 17 Mar 4 03:36 .ansible
-rw- 1 student student 18 Nov 5 07:46 .bash_logout
-rw- 1 student student 141 Nov 5 07:46 .bash_profile
-rw- 1 student student 492 Nov 5 07:46 .bashrc
drwx 9 student student 4096 Mar 8 22:37 .cache
drwx 8 student student 4096 Mar 8 22:37 .config
drwx 2 student student 6 Mar 8 22:37 Desktop
drwx 2 student student 6 Mar 8 22:37 Documents
drwx 2 student student 6 Mar 8 22:37 Downloads
-rw- 1 student student 0 Mar 8 22:39 editing_final_lab.txt
drwx 2 student student 25 Mar 4 03:37 .grading
drwx 4 student student 32 Mar 8 22:37 .local
drwx 2 student student 6 Mar 8 22:37 Music
drwx 2 student student 6 Mar 8 22:37 Pictures
drwx 2 student student 6 Mar 8 22:37 Public
drwx 2 student student 77 Mar 4 03:31 .ssh
drwx 2 student student 6 Mar 8 22:37 Templates
drwx 3 student student 18 Mar 4 03:36 .venv
drwx 2 student student 6 Mar 8 22:37 Videos
-- VISUAL BLOCK --
19x7 19,23 All

```

- Enter the visual block mode of Vim to remove the time column, to leave the month and day columns on all lines.

Use the arrow keys to position the cursor at the first character of the current seventh column. Enter the visual block mode by typing `Ctrl+V`. Use the arrow keys to position the cursor at the last character of the seventh column on the last row. Delete the selection by typing `x`.

```

student@workstation:~ -- /usr/bin/vim editing_final_lab.txt
drwx 3 student 17 Mar 4 01:36 .ansible
-rw- 1 student 18 Nov 5 07:46 .bash_logout
-rw- 1 student 141 Nov 5 07:46 .bash_profile
-rw- 1 student 492 Nov 5 07:46 .bashrc
drwx 9 student 4096 Mar 8 22:37 .cache
drwx 8 student 4096 Mar 8 22:37 .config
drwx 2 student 6 Mar 8 22:37 Desktop
drwx 2 student 6 Mar 8 22:37 Documents
drwx 2 student 6 Mar 8 22:37 Downloads
-rw- 1 student 0 Mar 8 22:39 editing_final_lab.txt
drwx 2 student 25 Mar 4 03:37 .grading
drwx 4 student 32 Mar 8 22:37 .local
drwx 2 student 6 Mar 8 22:37 Music
drwx 2 student 6 Mar 8 22:37 Pictures
drwx 2 student 6 Mar 8 22:37 Public
drwx 2 student 77 Mar 4 03:31 .ssh
drwx 2 student 6 Mar 8 22:37 Templates
drwx 3 student 18 Mar 4 03:36 .venv
drwx 2 student 6 Mar 8 22:37 Videos

-- VISUAL BLOCK --
19x5 19,34 All

```

- Enter the visual line mode of Vim and remove the rows that contain the `Desktop` and `Public` strings.

Use the arrow keys to position the cursor at any character on the `Desktop` row. Enter visual mode with uppercase `V`. The full line is selected. Delete the selection by typing `x`. Repeat the operation for the row with the `Public` string.

```

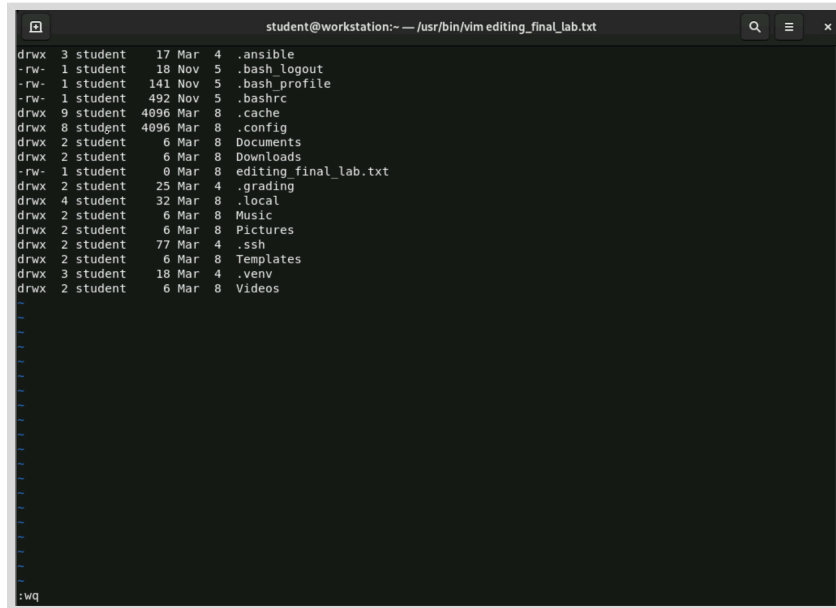
student@workstation:~ -- /usr/bin/vim editing_final_lab.txt
drwx 3 student 17 Mar 4 .ansible
-rw- 1 student 18 Nov 5 .bash_logout
-rw- 1 student 141 Nov 5 .bash_profile
-rw- 1 student 492 Nov 5 .bashrc
drwx 9 student 4096 Mar 8 .cache
drwx 8 student 4096 Mar 8 .config
drwx 2 student 6 Mar 8 Desktop
drwx 2 student 6 Mar 8 Documents
drwx 2 student 6 Mar 8 Downloads
-rw- 1 student 0 Mar 8 editing_final_lab.txt
drwx 2 student 25 Mar 4 .grading
drwx 4 student 32 Mar 8 .local
drwx 2 student 6 Mar 8 Music
drwx 2 student 6 Mar 8 Pictures
drwx 2 student 6 Mar 8 Public
drwx 2 student 77 Mar 4 .ssh
drwx 2 student 6 Mar 8 Templates
drwx 3 student 18 Mar 4 .venv
drwx 2 student 6 Mar 8 Videos

-- VISUAL LINE --
1 7,38 All

```

9. Save your changes and exit the file.

To save and exit the file, enter the last-line `:wq` command.



10. Back up the `editing_final_lab.txt` file and append the date (in seconds) at the end of the file name preceded with an underscore (`_`) character. Use the `lab_file` shell variable.

Use the `cp` command to back up the `editing_final_lab.txt` file. Use the `$(date +%s)` command at the end of the backup name preceded with an underscore character to make the name unique.

```
[student@workstation ~]$ cp $lab_file \
editing_final_lab_$(date +%s).txt
```

11. Append a dashed line to the `editing_final_lab.txt` file. The dashed line should contain 12 dash (`-`) characters for this lab to be graded correctly. Use the `lab_file` shell variable.

Use the `echo` command with 12 dashes and append the output to the `editing_final_lab.txt` file.

```
[student@workstation ~]$ echo "-----" >> $lab_file
```

12. List the content of the `Documents` directory, and append the output to the `editing_final_lab.txt` file, and display the output in the terminal. Use the `tee` command and the `lab_file` shell variable.

Use the `ls` command to list the `Documents` directory and pipe the output to the `tee -a` command to append the output to the `editing_final_lab.txt` file.

```
[student@workstation ~]$ ls Documents/ | tee -a $lab_file
lab_review.txt
```


13. Confirm that the directory listing is at the bottom of the lab file. Use the `lab_file` shell variable.

```
[student@workstation ~]$ cat $lab_file
drwx  3 student   17 Mar  4 .ansible
-rw-  1 student   18 Nov  5 .bash_logout
-rw-  1 student  141 Nov  5 .bash_profile
-rw-  1 student  492 Nov  5 .bashrc
drwx  9 student 4096 Mar  8 .cache
drwx  8 student 4096 Mar  8 .config
drwx  2 student   6 Mar  8 Documents
drwx  2 student   6 Mar  8 Downloads
-rw-  1 student   0 Mar  8 editing_final_lab.txt
drwx  2 student  25 Mar  4 .grading
drwx  4 student  32 Mar  8 .local
drwx  2 student   6 Mar  8 Music
drwx  2 student   6 Mar  8 Pictures
drwx  2 student  77 Mar  4 .ssh
drwx  2 student   6 Mar  8 Templates
drwx  3 student  18 Mar  4 .venv
drwx  2 student   6 Mar  8 Videos
-----
lab_review.txt
```

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 edit-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 edit-review
```

Summary

- Running programs, or processes, have three standard communication channels: standard input, standard output, and standard error.
- You can use I/O redirection to read standard input from a file or to write the output or errors from a process to a file.
- Pipelines can connect standard output from one process to the standard input of another process, and can format output or build complex commands.
- Know how to use at least one command-line text editor, and Vim is the recommended option because it is commonly installed by default in Linux distributions.
- Shell variables can help you to run commands, and are unique to a shell session.
- You can modify the behavior of the shell or the processes with environment variables.

Chapter 6

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, manage, 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 people and programs that can run commands.

Users have *usernames* 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 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 ttyS0    Ss+  22:43   0:00 /sbin/agetty -o -p --
\u --keep-baud 1
user01    1769  0.0  0.1 377700  6844 tty2     Ssl+ 22:45   0:00 /usr/libexec/gdm-x-
session --register-
user01    1773  1.3  1.3 528948 78356 tty2     Sl+  22:45   0:03 /usr/libexec/Xorg vt2
-displayfd 3 -au
user01    1800  0.0  0.3 521412 19824 tty2     Sl+  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; it 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 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; it 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 supplementary group.

Primary Groups and Supplementary 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 a regular user is created, 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 separated by default.

Users might also have *supplementary groups*. Membership in supplementary 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 supplementary. For example, if the `user01` user has a `user01` primary group and `wheel` and `webadmin` supplementary 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 supplementary groups.

```
[user01@host ~]$ id
uid=1001(user01) gid=1003(user01) groups=1003(user01),10(wheel),10000(group01)
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 can control most devices, but some exceptions apply. 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 to escalate privileges to `root` only when needed.

The `root` account on Linux is similar 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 username, then the `su` or `su -` command attempts to switch to `root` by default.

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

The `su` command starts a *non-login shell*, whereas the `su -` command (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, whereas `su` starts a shell as that user, but uses the original user's environment settings.

Usually, 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 the `root` user. However, you can use 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 that 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:

	su	su -	sudo
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	<code>su</code> command only	<code>su</code> 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 of 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 previously used this group for a non-standard purpose must update a previous configuration, to prevent unexpected and unauthorized users from 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. 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 `wheel` word 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 as 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

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
```

Although obvious security risks apply to granting this level of access to a user or group, system administrators often use this approach with cloud instances, virtual machines, and provisioning systems for configuring servers. You must 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 `ec2-user` 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

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 `su -` 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 the `operator1` user.

6.1. Switch to the `operator1` user.

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

6.2. Try 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. Try to view the last five lines of `/var/log/messages` by using `sudo`. It should succeed.

```
[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. Try 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. Try 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. Try 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. Try 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
```

Manage Local User Accounts

Objectives

- Create, modify, and delete local user accounts.

Manage Local Users

You can use command-line tools to manage local user accounts. This section reviews some important tools.

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. Usually, you can use the same options with the `usermod` command to modify an existing user.

The `/etc/login.defs` file sets some 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 a UID by using the `-u` option.

Modify Existing Users from the Command Line

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

usermod options:	Usage
<code>-a, --append</code>	Use it with the <code>-G</code> option to add the supplementary groups to the user's current set of group memberships instead of replacing the set of supplementary 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 supplementary groups for the user account.

usermod options:	Usage
-L, --lock	Lock the user account.
-m, --move-home	Move the user's home directory to a new location. You must use it with the -d option.
-s, --shell SHELL	Specify a particular login shell for the user account.
-U, --unlock	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, an unassigned UID now owns the user's files. If you create a user and that user is assigned the deleted user's UID, then the new account owns 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 scenario 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 the `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 that are statically assigned to system processes.
- **UID 201-999** : UIDs that are assigned to system processes that do not own files on this system. Software that requires an unprivileged UID is dynamically assigned a 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

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 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 **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 directory 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 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 users-user
```

Manage Local Group Accounts

Objectives

- Create, modify, and delete local group accounts.

Manage Local Groups

Several command-line tools enable group management. Although you can use the **Users** GUI utility to manage groups, Red Hat recommends that you 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 that the **GID_MIN** and **GID_MAX** variables specify in the **/etc/login.defs** file. By default, the command assigns a GID value that is greater than any other existing GIDs, even if a lower value becomes available.

The **groupadd** command **-g** option specifies a 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 supplementary 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, ensure first that you 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 supplementary 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 supplementary groups that are not included in the `-G` option's list.

Compare Primary and Supplementary 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 belong to only one primary group at a time.

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

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

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

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 a supplementary group that you already belong to. You might switch if you are about to create files, manually or scripted, and want to assign a different group as owner when they are being created.

Use the `newgrp` command to switch your primary group, in this shell session. You can switch between any primary or supplementary group that you belong to, but only one group at a time can be primary. Your primary group returns to the default if you log out and log in again. In this example, the `group01` group 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

Create groups, use them as supplementary 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 supplementary 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 the root user.

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

- 2. Create the `operator`'s supplementary group with a GID of 30000.

```
[root@servera ~]# groupadd -g 30000 operators
```

- 3. Create the `admin` supplementary group without specifying a GID.

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

- 4. Verify that both the `operators` and `admin` supplementary 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 supplementary 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
```

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. These passwords were considered adequate until dictionary attacks on encrypted passwords became common. The cryptographically hashed 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 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:$6$CSsXsd3rwghsdarf:17933:0:99999:7:2:18113:
```

Each field of this code block is separated by a colon:

- **user03** : Name of the user account.
- **\$6\$CSsXsd3rwghsdarf** : The cryptographically hashed 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 that 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 is reserved for future use.

Format of an Cryptographically Hashed Password

The cryptographically hashed password field stores three pieces of information: the hashing algorithm in use, the *salt*, and the cryptographical hash. Salt adds random data to the cryptographical hash, for creating a unique hash to strengthen the cryptographically hashed password. Each piece of information is delimited by the dollar (\$) character.

```
$6$CSsXcY61L/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 cryptographically hash the password; it is originally chosen at random.
- **2W6evvJahUfzfHpc9X.45Jc6H30E** : The cryptographical hash of the user's password; combining the salt and the plain text password and then cryptographically hashing 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, and combines the salt for the user with the plain text typed password. The system then cryptographically hashes the combination of the salt and plain text password with the specified hashing algorithm. If the result matches the cryptographical hash, then the user typed the right password. If the result does not match the cryptographical hash, then the user typed the wrong password and the login attempt fails. This method enables 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". Do not confuse the command with the word "change".

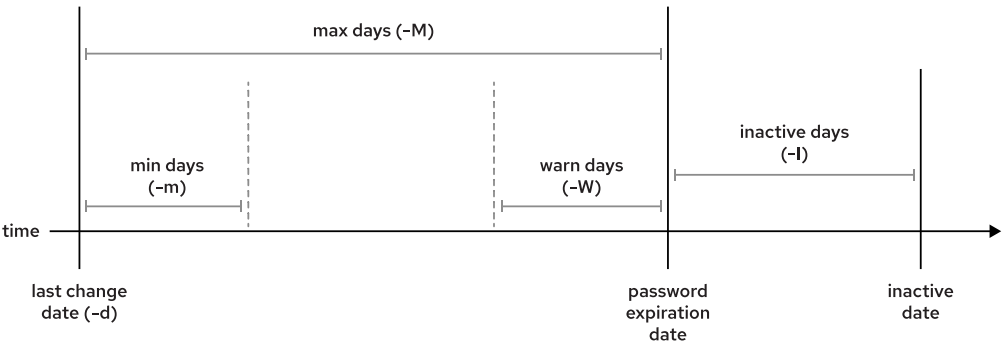


Figure 6.1: Password aging parameters

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 `cloudadmin10` 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 ❶
2022-03-10
[root@host ~]# date -d "+30 days" +%F ❷
2022-04-09
[root@host ~]# chage -E $(date -d "+30 days" +%F) cloudadmin10 ❸
[root@host ~]# chage -l cloudadmin10 | grep "Account expires" ❹
Account expires      : Apr 09, 2022
```

- ❶ Use the `date` command to get the current date.
- ❷ Use the `date` command to get the date 30 days from now.
- ❸ Use the `chage` command `-E` option to change the expiration date for the `cloudadmin10` user.
- ❹ 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 when 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 that 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 earlier password aging settings rather than the later 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 else 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 log in interactively 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 in directly 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

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 the **student** user to the **servera** machine.

```
[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: student
```

- 2.2. Try 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. Try 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: 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
```

► Lab

Manage Local Users and Groups

Set a default local password policy, create a supplementary 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 supplementary group for new users.
- Create three new users with the new supplementary group.
- Set an initial password for the created users.
- Configure the supplementary 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. Avoid using `visudo` to edit the `/etc/sudoers` file. Instead, place the configuration file in the `/etc/sudoers.d` directory.
5. Create the `consultant1`, `consultant2`, and `consultant3` users with the `consultants` group as their supplementary 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
```

► Solution

Manage Local Users and Groups

Set a default local password policy, create a supplementary 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 supplementary group for new users.
- Create three new users with the new supplementary group.
- Set an initial password for the created users.
- Configure the supplementary 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 when opening the file with the text editor. You can use the `vim /etc/login.defs` command for 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. Avoid using visudo to edit the /etc/sudoers file. Instead, place the configuration file in the /etc/sudoers.d directory.

- 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 for this step.

```
%consultants  ALL=(ALL) ALL
```

5. Create the consultant1, consultant2, and consultant3 users with the consultants group as their supplementary 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
```

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 supplementary 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 7

Control Access to Files

Goal

Set Linux file-system permissions on files and interpret the security effects of different permission settings.

Objectives

- List file-system permissions on files and directories, and interpret the effects of those permissions on access by users and groups.
- Change the permissions and ownership of files with command-line tools.
- Control the default permissions of user-created files, explain the effects of special permissions, and use special and default permissions to set the group owner of files that are created in a directory.

Sections

- Interpret Linux File System Permissions (and Quiz)
- 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

Interpret Linux File System Permissions

Objectives

- List file system permissions on files and directories, and interpret the effect of those permissions on access by users and groups.

Linux File-system Permissions

File permissions control access to files. Linux file permissions are flexible and can handle most normal permission cases.

Files have three user categories that permissions apply to. The file is owned by a user, normally the file creator. The file is also owned by a single group, which is usually the primary group of the user who created the file, but you can change it.

You can set different permissions for the owning user (*user permissions*), the owning group (*group permissions*), and for all other users on the system that are not the user or a member of the owning group (*other permissions*).

The most specific permissions take precedence. User permissions override group permissions, which override other permissions.

As an example of how group membership enables collaboration between users, imagine that your system has two users: `alice` and `bob`. `alice` is a member of the `alice` and `web` groups, and `bob` is a member of the `bob`, `wheel`, and `web` groups. When `alice` and `bob` work together, the files should be associated with the `web` group, and group permissions should allow both users to have access to the files.

Three permission categories apply: read, write, and execute. The following table explains how these permissions affect access to files and directories.

Effects of Permissions on Files and Directories

Permission	Effect on files	Effect on directories
r (read)	File contents can be read.	Contents of the directory (the file names) can be listed.
w (write)	File contents can be changed.	Any file in the directory can be created or deleted.
x (execute)	Files can be executed as commands.	The directory can become the current working directory. You can run the <code>cd</code> command to it, but it also requires read permission to list files there.

Users normally have both read and execute permissions on read-only directories so that they can list the directory and have read-only access to all of its contents. If a user has only read access on a directory, then they can list the names of the files in it. However, the user cannot access other information, such as permissions or time stamps. If a user has only execute access on a directory, then they cannot list file names in the directory. If they know the name of a file that they have

permission to read, then they can access the contents of that file from outside the directory by explicitly specifying the relative file name.

Anyone who owns or has both write and execute permissions on a directory can remove files from it, regardless of the ownership or permissions on the file itself. You can override this behavior by using the *sticky bit* permission, to be discussed later in this chapter.



Note

Linux file permissions work differently from the Microsoft NTFS file-system permissions. On Linux, permissions apply only to the file or directory on which they are set. The subdirectories within a directory do not automatically inherit the parent directory's permissions. However, directory permissions can block access to the directory contents, if they are set restrictively.

The **read** permission on a Linux directory is similar to **List folder contents** in Windows. The **write** permission on a Linux directory is similar to **Modify** in Windows; it implies the ability to delete files and subdirectories. In Linux, with **write** permissions and the **sticky bit** on a directory, then only the user or group owner can delete files, which is similar to the Windows **Write** behavior.

The Linux **root** user has the equivalent of the Windows **Full Control** permission on all files. However, SELinux policy can use process and file security contexts to restrict access even to the **root** user. SELinux is discussed in the *Red Hat System Administration II* (RH134) course.

View File and Directory Permissions and Ownership

The **ls** command -**l** option shows detailed information about permissions and ownership:

```
[user@host ~]$ ls -l test
-rw-rw-r--. 1 student student 0 Mar  8 17:36 test
```

Use the **ls** command -**d** option to show detailed information about a directory itself, and not its contents.

```
[user@host ~]$ ls -ld /home
drwxr-xr-x. 5 root root 4096 Feb 31 22:00 /home
```

The first character of the long listing is the file type, and is interpreted as follows:

- **-** is a regular file.
- **d** is a directory.
- **l** is a symbolic link.
- **c** is a character device file.
- **b** is a block device file.
- **p** is a named pipe file.
- **s** is a local socket file.

The next nine characters represent the file permissions. These characters are interpreted as three sets of three characters: The first set are permissions that apply to the file owner. The second set are for the file's group owner. The last set applies to all other (world) users. If a set is an **rwX** string, then that set has all three permissions: read, write, and execute. If a letter is replaced by a - dash character, then that set does not have that permission.

After the second column (the link count), the first name specifies the file owner, and the second name is the file's group owner.

In the first example, the permissions for the `student` user are the first set of three characters. The `student` user has read and write permissions on the `test` file, but not execute permission. The second set of three characters are the permissions for the `student` group: read and write permissions on `test`, but not execute permission. The third set of three characters are the permissions for all other users: only read permission on `test`.

The most specific set of permissions applies. So if the `student` user has different permissions from the `student` group, and the `student` user is also a member of that group, then only the user owner permissions apply. This permission allows setting a more restrictive set of permissions on a user than their group membership provides, when it might not be practical to remove the user from the group.

Examples of Permission Effects

The following examples illustrate how file permissions interact. For these examples, your system has four users with the following group memberships:

User	Group Memberships
operator1	operator1, consultant1
database1	database1, consultant1
database2	database2, operator2
contractor1	contractor1, operator2

Those users work with files in the `dir` directory. A long listing of the files in that directory is as follows:

```
[database1@host dir]$ ls -la
total 24
drwxrwxr-x. 2 database1 consultant1 4096 Mar  4 10:23 .
drwxr-xr-x. 10 root      root      4096 Mar  1 17:34 ..
-rw-rw-r--. 1 operator1 operator1 1024 Mar  4 11:02 app1.log
-rw-r--rw-. 1 operator1 consultant1 3144 Mar  4 11:02 app2.log
-rw-rw-r--. 1 database1 consultant1 10234 Mar  4 10:14 db1.conf
-rw-r-----. 1 database1 consultant1 2048 Mar  4 10:18 db2.conf
```

The `ls` command `-a` option shows the permissions of hidden files, including the special files to represent the directory and its parent. In this example, the `.` special directory reflects the permissions of `dir` itself, and the `..` special directory reflects the permissions of its parent directory.

For the `db1.conf` file, the user that owns the file (`database1`) has read and write permissions but not execute permission. The group that owns the file (`consultant1`) has read and write permissions but not execute permission. All other users have read permission but not write or execute permissions.

The following table explores some effects of this set of permissions for these users:

Effect	Why is this effect true?
The <code>operator1</code> user can change the contents of the <code>db1.conf</code> file.	The <code>operator1</code> user is a member of the <code>consultant1</code> group, and that group has both read and write permissions on the <code>db1.conf</code> file.
The <code>database1</code> user can view and modify the contents of the <code>db2.conf</code> file.	The <code>database1</code> user owns the <code>db2.conf</code> file and has both read and write access.
The <code>operator1</code> user can view but not modify the contents of the <code>db2.conf</code> file.	The <code>operator1</code> user is a member of the <code>consultant1</code> group, and that group has only read access to the <code>db2.conf</code> file.
The <code>database2</code> and <code>contractor1</code> users do not have any access to the contents of the <code>db2.conf</code> file.	The <code>other</code> permissions apply to the <code>database2</code> and <code>contractor1</code> users, and those permissions do not include the read or write permission.
The <code>operator1</code> user is the only user who can change the contents of the <code>app1.log</code> file.	The <code>operator1</code> user and members of the <code>operator1</code> group have write permission on the file, whereas other users do not. However, the only member of the <code>operator1</code> group is the <code>operator1</code> user.
The <code>database2</code> user can change the contents of the <code>app2.log</code> file.	The <code>database2</code> user is not the owner of the <code>app2.log</code> file and is not in the <code>consultant1</code> group, and so the <code>other</code> permissions apply. The <code>other</code> permissions grant write permission to the file.
The <code>database1</code> user can view the contents of the <code>app2.log</code> file, but cannot modify the contents of the <code>app2.log</code> file.	The <code>database1</code> user is a member of the <code>consultant1</code> group, and that group has only read permissions on the <code>app2.log</code> file. Even though the <code>other</code> permissions include write permission, the group permissions take precedence.
The <code>database1</code> user can delete the <code>app1.log</code> and <code>app2.log</code> files.	The <code>database1</code> user has write permissions on the <code>dir</code> directory, which the <code>.</code> special directory shows, and therefore can delete any file in that directory. This operation is possible even if the <code>database1</code> user does not have write permission on the files directly.



References

- ls(1) man page
- info coreutils (*GNU Coreutils*)
- Section 13: Changing File Attributes

► Quiz

Interpret Linux File System Permissions

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

The system has four users that are assigned to the following groups:

- The `consultant1` user is a member of the `consultant1` and `database1` groups.
- The `operator1` user is a member of the `operator1` and `database1` groups.
- The `contractor1` user is a member of the `contractor1` and `contractor3` groups.
- The `operator2` user is a member of the `operator2` and `contractor3` groups.

The `.` special directory contains four files with the following permissions:

<code>drwxrwxr-x.</code>	<code>operator1</code>	<code>database1</code>	<code>.</code>
<code>-rw-rw-r--.</code>	<code>consultant1</code>	<code>consultant1</code>	<code>app1.log</code>
<code>-rw-r--rw-.</code>	<code>consultant1</code>	<code>database1</code>	<code>app2.log</code>
<code>-rw-rw-r--.</code>	<code>operator1</code>	<code>database1</code>	<code>db1.conf</code>
<code>-rw-r-----.</code>	<code>operator1</code>	<code>database1</code>	<code>db2.conf</code>

Choose the correct answers to the following questions:

- 1. Which regular file does the `operator1` user own and all users can read?
 - a. `app1.log`
 - b. `app2.log`
 - c. `db1.conf`
 - d. `db2.conf`
- 2. Which file can the `contractor1` user modify?
 - a. `app1.log`
 - b. `app2.log`
 - c. `db1.conf`
 - d. `db2.conf`
- 3. Which file can the `operator2` user not read?
 - a. `app1.log`
 - b. `app2.log`
 - c. `db1.conf`
 - d. `db2.conf`

► 4. Which file does the consultant1 group own?

- a. app1.log
- b. app2.log
- c. db1.conf
- d. db2.conf

► 5. Which files can the operator1 user delete?

- a. Only db1.conf
- b. Only db2.conf
- c. All the files in the directory
- d. None of the files

► 6. Which files can the operator2 user delete?

- a. Only app1.log
- b. Only app2.log
- c. Both app1.log and app2.log
- d. None of the files

► Solution

Interpret Linux File System Permissions

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

The system has four users that are assigned to the following groups:

- The `consultant1` user is a member of the `consultant1` and `database1` groups.
- The `operator1` user is a member of the `operator1` and `database1` groups.
- The `contractor1` user is a member of the `contractor1` and `contractor3` groups.
- The `operator2` user is a member of the `operator2` and `contractor3` groups.

The `.` special directory contains four files with the following permissions:

<code>drwxrwxr-x.</code>	<code>operator1</code>	<code>database1</code>	<code>.</code>
<code>-rw-rw-r--.</code>	<code>consultant1</code>	<code>consultant1</code>	<code>app1.log</code>
<code>-rw-r--rw-.</code>	<code>consultant1</code>	<code>database1</code>	<code>app2.log</code>
<code>-rw-rw-r--.</code>	<code>operator1</code>	<code>database1</code>	<code>db1.conf</code>
<code>-rw-r-----.</code>	<code>operator1</code>	<code>database1</code>	<code>db2.conf</code>

Choose the correct answers to the following questions:

► 1. Which regular file does the `operator1` user own and all users can read?

- `app1.log`
- `app2.log`
- `db1.conf`
- `db2.conf`

► 2. Which file can the `contractor1` user modify?

- `app1.log`
- `app2.log`
- `db1.conf`
- `db2.conf`

► 3. Which file can the `operator2` user not read?

- `app1.log`
- `app2.log`
- `db1.conf`
- `db2.conf`

► **4. Which file does the consultant1 group own?**

- a. app1.log
- b. app2.log
- c. db1.conf
- d. db2.conf

► **5. Which files can the operator1 user delete?**

- a. Only db1.conf
- b. Only db2.conf
- c. All the files in the directory
- d. None of the files

► **6. Which files can the operator2 user delete?**

- a. Only app1.log
- b. Only app2.log
- c. Both app1.log and app2.log
- d. None of the 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 has the following characteristics: It changes file and directory permissions from the command line. It can be interpreted as "change mode", because the *mode* of a file is another name for file permissions. It 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.

```
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 the default.

Who	Set	Description
u	<i>user</i>	The file owner.
g	<i>group</i>	Member of the file's group.
o	<i>other</i>	Users who are not the file owner nor members of the file's group.
a	<i>all</i>	All the three previous groups.

What is the operator that modifies the *Which*, as in the following table.

What	Operation	Description
+	<i>add</i>	Adds the permissions to the file.
-	<i>remove</i>	Removes the permissions to the file.
=	<i>set exactly</i>	Sets exactly the provided permissions to the file.

Which is the mode, and specifies the permissions to the files or directories, as in the following table.

Which	Mode	Description
r	<i>read</i>	Read access to the file. Listing access to the directory.

Which	Mode	Description
w	write	Write permissions to the file or directory.
x	execute	Execute permissions to the file. Allows entering the directory, and accessing files and subdirectories inside the directory.
X	special execute	Execute permissions to a directory, or execute permissions to a file if at least one of the execute bits is set.

The *symbolic* method of changing file permissions uses letters to represent the permission groups: **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 group that own 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. With the `chmod` command `X` option, you can 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 any execute permission is set on a file, 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 applies group execute permissions only to directories and files where execute permission is already 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.
- To add read permissions for this access level, add 4.
- To add write permissions, add 2.
- To add execute permissions, add 1.

The following diagram illustrates how systems interpret the 644 octal permission value.

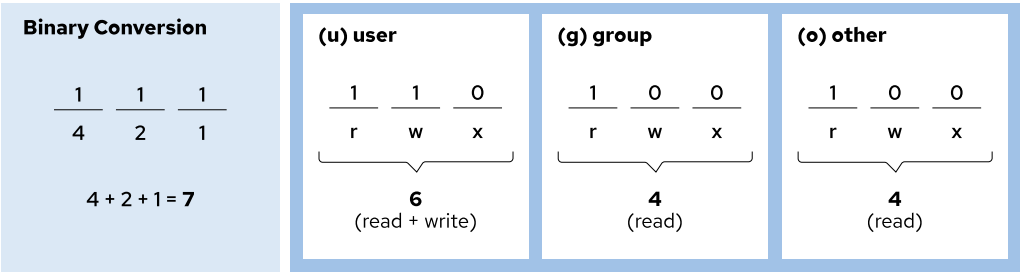


Figure 7.1: Visual representation of the octal method

Experienced administrators often use octal permissions to implement on single or matching files, and provide 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 `samplendir` directory:

```
[user@host ~]$ chmod 750 samplendir
```

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.

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 regular users can **ONLY** 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 not using this syntax, and always using a colon. Because a period is a valid character in a username, a `chown` command might misinterpret your intent. The command might 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

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 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
[sudo] password for student: 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. Modify the permissions of the `consultants` group to allow its group members to create files in, and delete files from, the `/home/consultants` directory.

The current permissions forbid others from accessing the files. You must set 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.

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. Use the symbolic method for setting the appropriate permissions.

```
[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. Use the octal method for setting the appropriate permissions.

```
[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 contents of the `/home/consultants` directory.

```
[consultant1@servera consultants]$ ls
consultant1.txt
```

- ▶ 8. 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 ~]$
```

- 9. Navigate to the `/home/consultants` directory. Ensure that the `consultant2` user can add content to the `consultant1.txt` file.

9.1. Change to the `/home/consultants` directory. Create an empty file called `consultant2.txt`.

```
[consultant2@servera ~]$ cd /home/consultants/
[consultant2@servera consultants]$ touch consultant2.txt
```

- 10. List the contents of the `/home/consultants` directory.

```
[consultant2@servera consultants]$ ls
consultant1.txt  consultant2.txt
```

- 11. 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
```


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 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 following table.

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 primary group of 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 factors 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.

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
0022
```

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. For example, *umask 077* is the same as *umask 0077*.

The system's default umask values for Bash shell users are defined in the */etc/login.defs* file, and in the */etc/bashrc* file. Users can override the system defaults in the *.bash_profile* or *.bashrc* files in their home directories.



Important

In Red Hat Enterprise Linux 8 and earlier, if a user account has a UID of 200 or greater and the username and the account's primary group name are the same, then its default umask is 0002. Otherwise, its default umask is 0022.

Red Hat Enterprise Linux 9 is in the process of changing the default umask so that all accounts have a umask of 0022. In RHEL 9.0 and 9.1, when you start a login shell, your umask is 0022. However, when you start an interactive non-login shell (such as when you start *gnome-terminal* in the graphical UI), your umask is 0002 if your account's UID is 200 or greater and your primary group has the same name as your user account. This default umask is expected to change in future versions of Red Hat Enterprise Linux 9 so that the umask defaults to 0022 in all circumstances.

Bugzilla issue https://bugzilla.redhat.com/show_bug.cgi?id=2062601 is tracking this change in the behavior of RHEL 9.

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.



Important

The following examples assume that the umask of the shell is set to 0022.

If you create a regular file, then its initial octal permissions are 0666 (000 110 110 110, in binary representation). Then, the 0022 umask (000 000 010 010) disables the write permission bit for group and others. Thus, the owner has both read and write permission on files, and both group and other are set to read (000 110 100 100).

	Symbolic	Numeric octal	Numeric binary
Initial file permissions	rw-rw-rw-	0666	000 110 110 110
umask	---w--w-	0022	000 000 010 010
Resulting file permissions	rw-r--r--	0644	000 110 100 100

Figure 7.2: Example of umask calculation on a file

```
[user@host ~]$ umask
0022
[user@host ~]$ touch default.txt
[user@host ~]$ ls -l default.txt
-rw-r--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 0022 umask (000 000 010 010) disables the write permission bit for group and other. Thus, the owner has read, write, and execute permissions on directories, and both group and other are set for read and execution (000 111 101 101).

	Symbolic	Numeric octal	Numeric binary
Initial directory permissions	rw-rwxrwx	0777	000 111 111 111
umask	---w--w-	0022	000 000 010 010
Resulting directory permissions	rw-r-xr-x	0755	000 111 101 101

Figure 7.3: Example of umask calculation on a directory

```
[user@host ~]$ umask
0022
[user@host ~]$ mkdir default
[user@host ~]$ ls -ld default
drwxr-xr-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 execute permissions 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
```

Changing Default Permissions

In Red Hat Enterprise Linux 9, the `/etc/login.defs` file sets the default umask for users. By default, its `UMASK` line specifies that the default umask is 0022.

In Red Hat Enterprise Linux 9.0 and 9.1, this file affects only login shells. If you start a new terminal window or start an interactive non-login shell in some other way, then settings in `/etc/bashrc` still apply. For these shells, 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 0002. Otherwise, the umask is 0022.

The `root` user can change the default umask for interactive non-login shells 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 0007 for users with a UID greater than 199 and with a username and primary group name that match, and to 0022 for everyone else. (Leading zeros can be omitted.) To set the umask to 0022 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, or until you change it manually with the `umask` command.



References

`bash(1)`, `ls(1)`, `chmod(1)`, and `umask(1)` man pages

► Guided Exercise

Manage Default Permissions and File Access

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 change.

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
0022
```

- 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
drwxr-xr-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-r--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
drwxr-xr-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-r--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-r--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-r--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
-rw-r-----. 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
0022
```

- ▶ 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. Create an `ops_prod2.txt` file in the `/tmp/shared/` directory. Verify that newly created files have read and write access for the `operators` group and no access for other users, due to the new `umask` of 007.

```
[operator1@servera ~]$ touch /tmp/shared/ops_prod2.txt
[operator1@servera ~]$ ls -l /tmp/shared/ops_prod2.txt
-rw-rw----. 1 operator1 operators 0 Feb  0 15:56 /tmp/shared/ops_prod2.txt
```

- ▶ 12. 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
```

► Lab

Control Access to Files

Configure permissions on files and set up a directory that users in a particular group can use to 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. Run the `sudo -i` command at the shell prompt to become the root user. Use `student` as the student user 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 correctly.
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 `/etc/login.defs` file to adjust the default `umask` for login shells. Normal users should have a `umask` setting that allows the user and group to create, write, and execute files and directories, and 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
```

► Solution

Control Access to Files

Configure permissions on files and set up a directory that users in a particular group can use to 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. Run the `sudo -i` command at the shell prompt to become the root user. Use `student` as the student user 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.

- 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 correctly.

```
[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. Add some text to the `/home/techdocs/techdoc1.txt` file. 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-r--r--. 1 tech1 techdocs 0 Feb  5 16:42 /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ echo "This is the first tech doc."
> /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 7.2. Switch to the `tech2` user. Display the content of the `/home/techdocs/techdoc1.txt` file. Create a `techdoc2.txt` file in the `/home/techdocs` directory. Exit from the `tech2` user shell.

```
[root@serverb ~]# su - tech2
[tech2@serverb ~]$ cd /home/techdocs
[tech2@serverb techdocs]$ cat techdoc1.txt
This is the first tech doc.
[tech2@serverb techdocs]$ touch /home/techdocs/techdoc2.txt
[tech2@serverb techdocs]$ ls -l
total 4
-rw-r--r--. 1 tech1 techdocs 28 Feb  5 17:43 techdoc1.txt
-rw-r--r--. 1 tech2 techdocs  0 Feb  5 17:45 techdoc2.txt
[tech2@serverb techdocs]$ exit
logout
[root@serverb ~]#
```

- 7.3. Switch to the `database1` user. Display the content of the `/home/techdocs/techdoc1.txt` file. You get a `Permission Denied` message. Verify that the `database1` user 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 ~]$ cat /home/techdocs/techdoc1.txt
cat: /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 `/etc/login.defs` file to adjust the default `umask` for login shells. Normal users should have a `umask` setting that allows the user and group to create, write, and execute files and directories, and 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
0022
[student@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 8.2. Edit the `/etc/login.defs` file and set a `umask` of `007`. The `/etc/login.defs` file already contains a `umask` definition. Search the file and update with the appropriate value.

```
[root@serverb ~]# cat /etc/login.defs
...output omitted...
UMASK          007
...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
```

Summary

- File ownership permissions have three categories. A file is owned by a user, a single group, and other users. The most specific permission applies. User permissions override group permissions, and group permissions override other permissions.
- The `ls` command - `l` option expands the file listing to include both the file permissions and ownership.
- 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/login.defs` file and might be affected by settings in the `/etc/profile` and `/etc/bashrc` files, files in `/etc/profile.d`, or your account's shell initialization files.
- The `suid`, `sgid`, and `sticky` special permissions provide additional access-related features to files.

Chapter 8

Monitor and Manage Linux Processes

Goal

Evaluate and control processes that run on a Red Hat Enterprise Linux system.

Objectives

- Determine status, resource use, and ownership of running programs on a system, to control them.
- Use Bash job control to manage multiple processes that were started from the same terminal session.
- 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.

Sections

- Process States and Lifecycle (and Quiz)
- Control Jobs (and Guided Exercise)
- Kill Processes (and Guided Exercise)
- Monitor Process Activity (and Guided Exercise)

Lab

- Monitor and Manage Linux Processes

Process States and Lifecycle

Objectives

- Determine status, resource use, and ownership of running programs on a system, to control them.

Definition of a Process

A *process* is a running instance of a launched, executable program. From the moment that a process is created, it consists of the following items:

- An address space of allocated memory
- Security properties, including ownership credentials and privileges
- One or more execution threads of program code
- A process state

The *environment* of a process is a list of information that includes the following items:

- Local and global variables
- A current scheduling context
- Allocated system resources, such as file descriptors and network ports

An existing *parent* process duplicates its own address space, which is known as a process *fork*, to create a *child* process structure. Every new process is assigned a unique *process ID* (PID) for tracking and security purposes. The PID and the parent's process ID (PPID) are elements of the new process environment. Any process can create a *child* process. All processes are descendants of the first system process, `systemd`, on a Red Hat system.

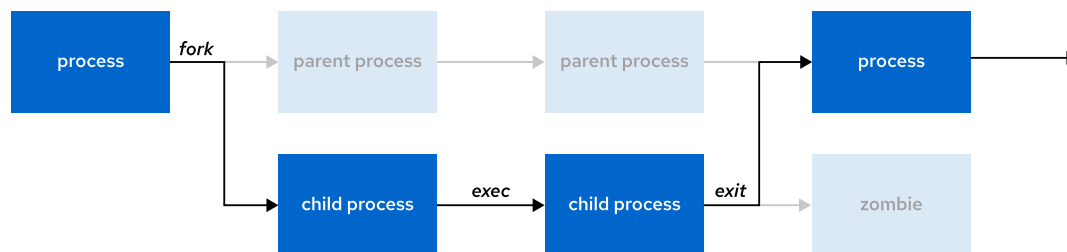


Figure 8.1: Process lifecycle

Through the *fork* routine, a child process inherits security identities, previous and current file descriptors, port and resource privileges, environment variables, and program code. A child process can then execute its own program code.

Normally, a parent process *sleeps* when the child process runs, and sets a *wait* request to be signaled when the child completes. After the child process exits, it closes or discards its resources and environment, and leaves a *zombie* resource, which is an entry in the process table. The parent, which is signaled to *wake* when the child exits, cleans the process table of the child's entry, and it frees the last resource of the child process. The parent process then continues with its own program code execution.

Describe Process States

In a multitasking operating system, each CPU (or CPU core) can be working on one process at a time. As a process runs, its immediate requirements for CPU time and resource allocation change. Processes are assigned a *state*, which changes as circumstances dictate.

The following diagram and table describe Linux process states in detail.

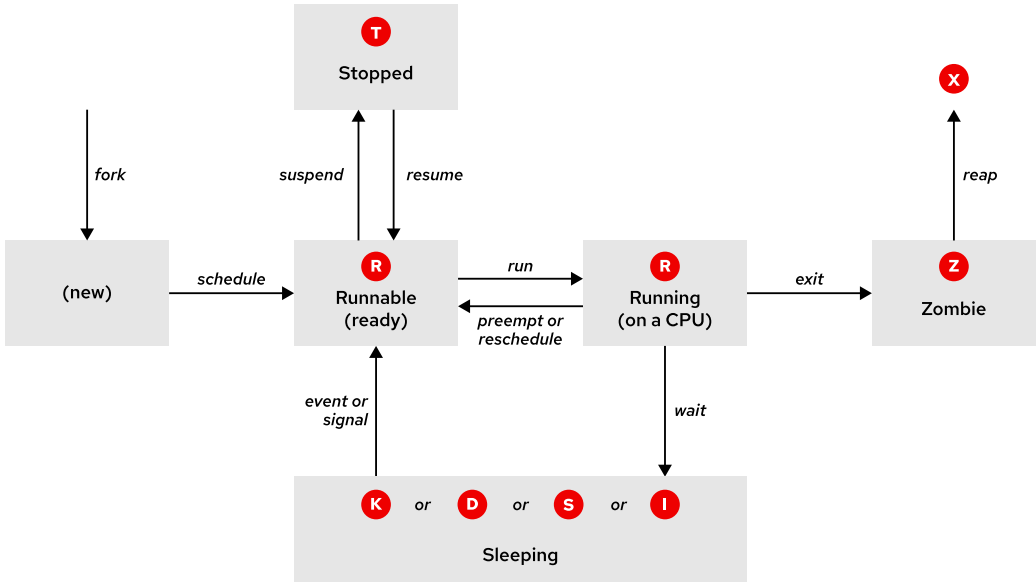


Figure 8.2: Linux process states

Linux Process States

Name	Flag	Kernel-defined state name and description
Running	R	TASK_RUNNING: The process is either executing on a CPU or waiting to run. The process can be executing user routines or kernel routines (system calls), or be queued and ready when in the <i>Running</i> (or <i>Runnable</i>) state.
Sleeping	S	TASK_INTERRUPTIBLE: The process is waiting for some condition: a hardware request, system resource access, or a signal. When an event or signal satisfies the condition, the process returns to <i>Running</i> .
	D	TASK_UNINTERRUPTIBLE: This process is also sleeping, but unlike the S state, does not respond to signals. It is used only when process interruption might cause an unpredictable device state.
	K	TASK_KILLABLE: Same as the uninterruptible D state, but modified to allow a waiting task to respond to the signal to kill it (exit completely). Utilities often display <i>Killable</i> processes as the D state.

Name	Flag	Kernel-defined state name and description
	I	TASK_REPORT_IDLE: A subset of state D. The kernel does not count these processes when calculating the load average. It is used for kernel threads. The TASK_UNINTERRUPTIBLE and TASK_NOLOAD flags are set. It is similar to TASK_KILLABLE, and is also a subset of state D. It accepts fatal signals.
Stopped	T	TASK_STOPPED: The process is stopped (suspended), usually by being signaled by a user or another process. The process can be continued (resumed) by another signal to return to running.
	T	TASK_TRACED: A process that is being debugged is also temporarily stopped and shares the T state flag.
Zombie	Z	EXIT_ZOMBIE: A child process signals to its parent as it exits. All resources except for the process identity (PID) are released.
	X	EXIT_DEAD: When the parent cleans up (reaps) the remaining child process structure, the process is now released completely. This state cannot be observed in process-listing utilities.

Importance of Process States

When troubleshooting a system, it is important to understand how the kernel communicates with processes, and how processes communicate with each other. The system assigns a state to every new process. The S column of the `top` command or the STAT column of the `ps` command shows the state of each process. On a single CPU system, only one process can run at a time. It is possible to see several processes with an R state. However, not all processes are running consecutively; some of them are in *waiting* status.

```
[user@host ~]$ top
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2259 root 20 0 270856 40316 8332 S 0.3 0.7 0:00.25 sssd_kcm
1 root 20 0 171820 16140 10356 S 0.0 0.3 0:01.62 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
...output omitted...

[user@host ~]$ ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
...output omitted...
root 2 0.0 0.0 0 0 ? S 11:57 0:00 [kthreadd]
student 3448 0.0 0.2 266904 3836 pts/0 R+ 18:07 0:00 ps aux
...output omitted...
```

Use signals to suspend, stop, resume, terminate, or interrupt processes. Processes can catch signals from the kernel, other processes, and other users on the same system. Signals are discussed later in this chapter.

Listing Processes

The `ps` command is used for listing detailed information for current processes.

- User identification (UID), which determines process privileges

- Unique process identification (PID)
- Amount of used CPU and elapsed real time
- Amount of allocated memory
- The process `stdout` location, which is known as the controlling terminal
- The current process state



Important

The Linux version of the `ps` command supports the following option formats:

- UNIX (POSIX) options, which can be grouped and must be preceded by a dash
- BSD options, which can be grouped and must not be used with a dash
- GNU long options, which are preceded by two dashes

For example, the `ps -aux` command is not the same as the `ps aux` command.

The common `ps` command `aux` option displays all processes including processes without a controlling terminal. A long listing (`lax` options) provides more detail, and gives faster results by avoiding username lookups. The similar UNIX syntax uses the `-ef` options to display all processes. In the following examples, scheduled kernel threads are displayed in brackets at the top of the list.

```
[user@host ~]$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.2 171820 16140 ?        Ss   16:47   0:01 /usr/lib/
systemd/systemd ...
root           2  0.0  0.0      0      0 ?        S    16:47   0:00 [kthreadd]
root           3  0.0  0.0      0      0 ?        I<   16:47   0:00 [rcu_gp]
root           4  0.0  0.0      0      0 ?        I<   16:47   0:00 [rcu_par_gp]
root           6  0.0  0.0      0      0 ?        I<   16:47   0:00 [kworker/0:0H-
events_highpri]
...output omitted...
[user@host ~]$ ps lax
F  UID      PID      PPID  PRI  NI     VSZ   RSS WCHAN    STAT TTY      TIME COMMAND
4   0         1         0   20   0 171820 16140 -          Ss   ?         0:01 /usr/lib/
systemd/systemd ...
1   0         2         0   20   0      0      0 -          S    ?         0:00 [kthreadd]
1   0         3         2    0 -20     0      0 -          I<   ?         0:00 [rcu_gp]
1   0         4         2    0 -20     0      0 -          I<   ?         0:00 [rcu_par_gp]
1   0         6         2    0 -20     0      0 -          I<   ?         0:00 [kworker/0:0H-
events_highpri]
...output omitted...
[user@host ~]$ ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1      0  0 16:47 ?          00:00:01 /usr/lib/systemd/systemd ...
root           2      0  0 16:47 ?          00:00:00 [kthreadd]
root           3      2  0 16:47 ?          00:00:00 [rcu_gp]
root           4      2  0 16:47 ?          00:00:00 [rcu_par_gp]
root           6      2  0 16:47 ?          00:00:00 [kworker/0:0H-events_highpri]
...output omitted...
```

By default, the `ps` command with no options selects all processes with the current user's *effective user ID* (EUID), and selects processes that are associated with the terminal that is running the command. Zombie processes are listed with the `exiting` or `defunct` label.

You can use the `ps` command - `-forest` option to display the processes in a tree format so you can view relationships between parent and child processes.

The default output of the `ps` command is sorted by process ID number. At first glance, the output might appear to use chronological order, but the kernel reuses process IDs, so the order is less structured than it appears. Use the `ps` command `-0` or `--sort` options to sort the output. The display order matches that of the system process table, which reuses table rows when processes die and spawn.



References

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

- Section 24: Signal Handling

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

- Section 26: Processes

`ps(1)` and `signal(7)` man pages

► Quiz

Process States and Lifecycle

Choose the correct answers to the following questions:

- 1. Which state represents a process that is stopped or suspended?
 - a. D
 - b. R
 - c. S
 - d. T
 - e. Z

- 2. Which state represents a process that released all of its resources except its PID?
 - a. D
 - b. R
 - c. S
 - d. T
 - e. Z

- 3. Which process does a parent use to duplicate its address space, and creates a child process?
 - a. exec
 - b. fork
 - c. zombie
 - d. syscall
 - e. reap

- 4. Which state represents a process that is sleeping until some condition is met?
 - a. D
 - b. R
 - c. S
 - d. T
 - e. Z

► Solution

Process States and Lifecycle

Choose the correct answers to the following questions:

- 1. Which state represents a process that is stopped or suspended?
 - a. D
 - b. R
 - c. S
 - d. T
 - e. Z

- 2. Which state represents a process that released all of its resources except its PID?
 - a. D
 - b. R
 - c. S
 - d. T
 - e. Z

- 3. Which process does a parent use to duplicate its address space, and creates a child process?
 - a. exec
 - b. fork
 - c. zombie
 - d. syscall
 - e. reap

- 4. Which state represents a process that is sleeping until some condition is met?
 - a. D
 - b. R
 - c. S
 - d. T
 - e. Z

Control Jobs

Objectives

- Use Bash job control to manage multiple processes that were started from the same terminal session.

Describe Jobs and Sessions

With the *job control* shell feature, a single shell instance can run and manage multiple commands.

A *job* is associated with each *pipeline* that is entered at a shell prompt. All processes in that pipeline are part of the job and are members of the same *process group*. You can consider a minimal pipeline to be only one command that is entered in the shell prompt to create a job with only one member.

Only one job at a time can read input and keyboard-generated signals from a particular terminal window. Processes that are part of that job are *foreground* processes of that *controlling terminal*.

A *background* process of that controlling terminal is any other job that is associated with that terminal. Background processes of a terminal cannot read input or receive keyboard-generated interrupts from the terminal, but can write to the terminal. A background job might be stopped (suspended) or it might be running. If a running background job tries to read from the terminal, then it is automatically suspended.

Each terminal runs in its own *session*, and can have a foreground process and any number of background processes. A job that runs in its own session belongs to its controlling terminal.

The `ps` command shows the device name of the controlling terminal in the TTY column. Some processes, such as *system daemons*, are started by the system and not from a controlling terminal. These processes are not members of a job, and cannot be brought to the foreground. The `ps` command displays a question mark (?) in the TTY column for these processes.

Run Jobs in the Background

Any command or pipeline can be started in the background by appending an ampersand (&) character to the command. The Bash shell displays a *job number* (unique to the session) and the PID of the new child process. The shell does not wait for the child process to terminate, but instead displays the shell prompt.

```
[user@host ~]$ sleep 10000 &
[1] 5947
[user@host ~]$
```

When a command line with a pipe (|) is sent to the background, the PID of the last command in the pipeline is displayed. All pipeline processes are members of that job.

```
[user@host ~]$ example_command | sort | mail -s "Sort output" &
[1] 5998
```

Use the `jobs` command to display the list of jobs for the shell's session.

```
[user@host ~]$ jobs
[1]+  Running      sleep 10000 &
[user@host ~]$
```

Use the `fg` command to bring a background job to the foreground. Use the `(%jobNumber)` format to specify the process to foreground.

```
[user@host ~]$ fg %1
sleep 10000
```

In the preceding example, the `sleep` command is running in the foreground on the controlling terminal. The shell itself is asleep and waiting for this child process to exit.

To send a foreground process to the background, press the keyboard-generated *suspend* request (`Ctrl+Z`) in the terminal. The job is placed in the background and suspended.

```
[user@host ~]$ sleep 10000
^Z
[1]+  Stopped              sleep 10000
[user@host ~]$
```

The `ps j` command displays information about jobs. Use the `ps j` command to find process and session information.

- The PID is the unique process ID of the process.
- The PPID is the PID of the *parent process* of this process, the process that started (forked) it.
- The PGID is the PID of the *process group leader*, normally the first process in the job's pipeline.
- The SID is the PID of the *session leader*, which (for a job) is normally the interactive shell that is running on its controlling terminal.

In the next example, the `sleep` command is currently suspended and the process state is T.

```
[user@host ~]$ ps j
  PPID   PID   PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
   2764   2768   2768   2768 pts/0    6377 Ss    1000    0:00 /bin/bash
   2768   5947   5947   2768 pts/0    6377 T    1000    0:00 sleep 10000
   2768   6377   6377   2768 pts/0    6377 R+    1000    0:00 ps j
```

Use the `bg` command with the job ID to start running the suspended process.

```
[user@host ~]$ bg %1
[1]+ sleep 10000 &
```

The shell warns a user who attempts to exit a terminal window (session) with suspended jobs. If the user tries again to exit immediately, then the suspended jobs are killed.

**Note**

In the previous examples, the + sign indicates that this job is the current default. If a job-control command is used without the `%jobNumber` argument, then the action is taken on the default job. The - sign indicates the previous job that will become the default job when the current default job finishes.

**References**

Bash info page (*The GNU Bash Reference Manual*)

<https://www.gnu.org/software/bash/manual>

- Section 7: Job Control

`bash(1)`, `builtins(1)`, `ps(1)`, and `sleep(1)` man pages

► Guided Exercise

Control Jobs

Use job control to start, suspend, and move multiple processes to the background and foreground.

Outcomes

- Use job control to suspend and restart user 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-control
```

Instructions

1. On the `workstation` machine, open two terminal windows side by side. In this section, these two 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
[student@servera ~]$
```

2. In the left terminal shell, create the `/home/student/bin` directory. Create a shell script called `control` in the `/home/student/bin` directory. Change the script permissions to make it executable.

2.1. Create the `/home/student/bin` directory.

```
[student@servera ~]$ mkdir /home/student/bin
```

2.2. Create a script called `control` in the `/home/student/bin` directory. To enter Vim interactive mode, press the `i` key. Use the `:wq` command to save the file and quit.

```
[student@servera ~]$ vim /home/student/bin/control
#!/bin/bash
while true; do
  echo -n "$@" >> ~/control_outfile
  sleep 1
done
```

**Note**

The *control* script runs until the process is terminated. The script appends command-line arguments to the `~/control_outfile` file once per second.

2.3. Make the `control` file executable.

```
[student@servera ~]$ chmod +x /home/student/bin/control
```

- ▶ **3.** Execute the `control` script. The script continuously appends the word "technical" and a space to the `~/control_outfile` file at one-second intervals.

```
[student@servera ~]$ control technical
```

- ▶ **4.** In the right terminal shell, verify that the new process is writing to the `/home/student/control_outfile` file.

```
[student@servera ~]$ tail -f ~/control_outfile
technical technical technical technical
...output omitted...
```

- ▶ **5.** In the left terminal shell, press `Ctrl+Z` to suspend the running process. The shell returns the job ID in square brackets. In the right terminal shell, confirm that the process output is stopped.

```
^Z
[1]+  Stopped                  control technical
[student@servera ~]$
```

```
technical technical technical technical
...no further output...
```

- ▶ **6.** In the left terminal shell, view the `jobs` command output. Remember that the `+` sign indicates the default job. Restart the job in the background. In the right terminal shell, verify that the process output is again active.

6.1. View the list of jobs.

```
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
```

6.2. Restart the `control` job in the background.

```
[student@servera ~]$ bg
[1]+ control technical &
```

6.3. Verify that the `control` job is running again.

```
[student@servera ~]$ jobs
[1]+  Running                  control technical &
```

6.4. In the right terminal shell, confirm that the `tail` command is producing output.

```
...output omitted...
technical technical technical technical technical technical technical technical
```

- 7. In the left terminal shell, start two more `control` processes to append to the `~/output` file. Use the ampersand (`&`) special command to start the processes in the background. Replace `technical` with `documents` and then with `database`. Replacing the arguments helps to differentiate between the three processes.

```
[student@servera ~]$ control documents &
[2] 6579
[student@servera ~]$ control database &
[3] 6654
```

- 8. In the left terminal shell, use the `jobs` command to view the three running processes. In the right terminal shell, verify that all three processes are appending to the file.

```
[student@servera ~]$ jobs
[1]  Running                  control technical &
[2]- Running                  control documents &
[3]+ Running                  control database &
```

```
...output omitted...
technical documents database technical documents database technical documents
database technical documents database
...output omitted...
```

- 9. Suspend the `control technical` process. Confirm that it is suspended. Terminate the `control documents` process and verify that it is terminated.

9.1. In the left terminal shell, foreground the `control technical` process. Press `Ctrl+Z` to suspend the process. Verify that the process is suspended.

```
[student@servera ~]$ fg %1
control technical
^Z
[1]+  Stopped                  control technical
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
[2]  Running                  control documents &
[3]-  Running                  control database &
```

9.2. In the right terminal shell, verify that the `control technical` process is no longer sending output.


```
database documents database documents database
...no further output...
```

9.3. In the left terminal shell, run the `control documents` process in the foreground. Press `Ctrl+C` to terminate the process. Verify that the process is terminated.

```
[student@servera ~]$ fg %2
control documents
^C
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
[3]-  Running                  control database &
```

9.4. In the right terminal shell, verify that the `control documents` process is no longer sending output.

```
...output omitted...
database database database database database database database database
...no further output...
```

► 10. In the left terminal shell, view the remaining jobs. The suspended jobs have a state of T. The other background jobs are sleeping and have a state of S.

```
[student@servera ~]$ ps jT
PPID  PID  PGID  SID TTY          TPGID STAT  UID   TIME COMMAND
27277 27278 27278 27278 pts/1      28702 Ss    1000   0:00 -bash
27278 28234 28234 27278 pts/1      28702 T    1000   0:00 /bin/bash /home/student/bin/control technical
27278 28251 28251 27278 pts/1      28702 S    1000   0:00 /bin/bash /home/student/bin/control database
28234 28316 28234 27278 pts/1      28702 T    1000   0:00 sleep 1
28251 28701 28251 27278 pts/1      28702 S    1000   0:00 sleep 1
27278 28702 28702 27278 pts/1      28702 R+   1000   0:00 ps jT
```

► 11. In the left terminal shell, view the current jobs. Terminate the `control database` process and verify that it is terminated.

```
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
[3]-  Running                  control database &
```

Use the `fg` command with the job ID to run the `control database` process in the foreground. Press `Ctrl+C` to terminate the process. Verify that the process is terminated.

```
[student@servera ~]$ fg %3
control database
^C
[student@servera ~]$ jobs
[1]+  Stopped                  control technical
```

- ▶ **12.** In the right terminal shell, use the `Ctrl+c` command to stop the `tail` command. Delete the `~/control_outfile` file.

```
...output omitted...  
^C  
[student@servera ~]$ rm ~/control_outfile
```

- ▶ **13.** Close the extra terminal. 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 processes-control
```

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 (Ctrl+c).
3	QUIT	Keyboard quit : Similar to SIGINT; adds a process dump at termination. Sent by pressing the QUIT key sequence (Ctrl+\).
9	KILL	Kill, unblockable : Causes abrupt program termination. It cannot be blocked, ignored, or handled; consistently fatal.
15 default	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-cleanup 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 (Ctrl+z).

**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) immediately.
- **Core** : Save a program's memory image (core dump), and 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, with the **-HUP** or **-SIGHUP** options) or by number (with 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

```

Control Specific Processes

Use the `pkill` command to signal one or more processes that match selection criteria. Selection criteria can be a command name, a process that a specific user owns, or all system-wide processes.

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. By default, the `pkill` command sends the `SIGTERM` signal to each process instead of listing them on `stdout`.

First, use the `pgrep` command to identify the PID numbers to kill. This command operates similarly to the `pkill` command, including most of the same options, except that the `pgrep` command lists processes rather than killing them.

Use the `pgrep` command with the `-l` option to list the process names and IDs. Use either command with the `-u` option to specify the ID of the user who owns the processes.

```

[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.

Use the `-t` option to match processes with a specific terminal ID.

```

[root@host ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@host ~]# w -u bob
USER      TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
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 ~]#
```



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 sending SIGTERM first, and then trying SIGINT; and only if both fail, trying again with SIGKILL.

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 it 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)
```

Signal Multiple Processes

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 ~]$
```

Terminate Background Jobs

To terminate background jobs, use the `kill` command and specify the job number.

Use the `jobs` command to find the job number of the process to terminate.

```
[user@host ~]$ jobs
[1]-  Running                  sleep 500 &
[2]+  Running                  sleep 1000 &
[user@host ~]$
```

Terminate a specific job by using the `kill` command. Prefix the job number with a percent sign (%).

```
[user@host ~]$ kill -SIGTERM %1
[user@host ~]$ jobs
[2]+  Running                  sleep 1000 &
```

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 a remote login session. If it is `ttyn`, then the user is on a system console, an alternative console, or another directly connected terminal device.

```
[user@host ~]$ w -f
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.



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

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 bin]$ 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
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 /home/student/
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
```

Monitor Process Activity

Objectives

- Define load average and determine resource-intensive server processes.

Describe Load Average

Load average is a measurement that the 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 usage or run queue length to indicate system load. Linux also includes disk or network usage, 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 load value per CPU to determine whether the system is experiencing significant waiting.

Use the `lscpu` command to determine the number of CPUs on a system.

In the following example, the system is a dual-core single-socket system with two hyper threads per core. 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...
```

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 use 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, and 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. Although not indicating CPU use, 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, because tasks are seldom found waiting in the queue. Load average increases only when resource saturation causes requests to remain queued, and the load calculation routine counts them. When resource use approaches 100%, each extra 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 username (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; the 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 username (effective, real).
Shift+m	Sort process listing by memory usage, in descending order.
Shift+p	Sort process listing by processor use, 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 <code>top</code> restart.
q	Quit.
f	Manage the columns by enabling or disabling fields. You can also set the sort field for <code>top</code> .



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

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 `l`, `t`, and `m` single keystrokes 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
 671 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 value is currently less than 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% to 25%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 671 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% to 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.

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
```

► Lab

Monitor and Manage Linux Processes

Locate and manage processes that use the most resources on a system.

Outcomes

- Manage processes with `top` as a process management tool.

Before You Begin

Log in to `workstation` as `student` with `student` as the password.

On `workstation`, run the `lab start processes-review` command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

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

Instructions

1. On `workstation`, open two terminal windows side by side. In this section, these terminals are referred to as *left* and *right*. On each terminal window, log in to `serverb` as the `student` user.

Create the `process101` script in the `/home/student/bin` directory. The `process101` script generates artificial CPU load.

```
#!/bin/bash
while true; do
  var=1
  while [[ var -lt 50000 ]]; do
    var=$((var+1))
  done
  sleep 1
done
```

2. In the right terminal shell, run the `top` utility.
3. In the left terminal shell, verify the number of logical CPUs on the virtual machine. Run the `process101` script in the background.
4. In the right terminal shell, observe the `top` display. Note the process ID (PID), and view the CPU percentage that the `process101` process uses. The CPU percentage that the process uses should hover around 10% to 15%. Toggle the `top` utility display between load, threads, and memory. Return to the CPU usage display of the `top` utility.
5. Turn off the use of bold in the display. Save this configuration for reuse when `top` is restarted. Confirm that the changes are saved.
6. Copy the `process101` script to a new `process102` file, and increase the artificial CPU load to one hundred thousand in the new script. Start the `process102` process in the background.

7. In the right terminal shell, verify that the process is running and uses the most CPU resources. The load should hover between 25% to 35%.
8. Notice that the load average is below 1. Copy the `process101` script to a new script called `process103`. Increase the addition count to eight hundred thousand. Start `process103` in the background. Confirm that the load average is above 1. It might take a few minutes for the load average to change.
9. In the left terminal shell, switch to the `root` user. Suspend the `process101` process. List the remaining jobs. Observe that the process state for `process101` is now in the T state.
10. Resume the `process101` process.
11. Terminate `process101`, `process102`, and `process103` from the command line. Verify that the processes are no longer displayed in `top`.
12. Stop the processes and return to the `workstation` machine.

Evaluation

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

```
[student@workstation ~]$ lab grade processes-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 processes-review
```

► Solution

Monitor and Manage Linux Processes

Locate and manage processes that use the most resources on a system.

Outcomes

- Manage processes with `top` as a process management tool.

Before You Begin

Log in to `workstation` as `student` with `student` as the password.

On `workstation`, run the `lab start processes-review` command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

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

Instructions

1. On `workstation`, open two terminal windows side by side. In this section, these terminals are referred to as *left* and *right*. On each terminal window, log in to `serverb` as the `student` user.

Create the `process101` script in the `/home/student/bin` directory. The `process101` script generates artificial CPU load.

```
#!/bin/bash
while true; do
  var=1
  while [[ var -lt 50000 ]]; do
    var=$((var+1))
  done
  sleep 1
done
```

- 1.1. On `workstation`, open two terminal windows side by side. In each terminal, use the `ssh` command to log in to the `serverb` machine as the `student` user.

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

- 1.2. In the left terminal shell, create the `/home/student/bin` directory.

```
[student@serverb ~]$ mkdir /home/student/bin
```

- 1.3. In the left terminal shell, create the `process101` script. Press the `i` key to enter the Vim interactive mode. Type `:wq` to save the file.

```
[student@serverb ~]$ vim /home/student/bin/process101
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 50000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

- 1.4. Make the `process101` script executable.

```
[student@serverb ~]$ chmod +x /home/student/bin/process101
```

2. In the right terminal shell, run the `top` utility.

- 2.1. Size the window to be as tall as possible.

```
[student@serverb ~]$ top
top - 17:02:43 up 42 min,  2 users,  load average: 0.00, 0.00, 0.00
Tasks: 120 total,  1 running, 119 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,   0.0 sy,   0.0 ni,100.0 id,   0.0 wa,   0.0 hi,   0.0 si,   0.0 st
MiB Mem : 1774.8 total, 1420.7 free,  206.3 used,  147.8 buff/cache
MiB Swap:   0.0 total,   0.0 free,   0.0 used. 1417.3 avail Mem

  PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root        20   0 105972   1752   10292 S   0.0   1.0   0:01.30 systemd
    2 root        20   0      0      0        0 S   0.0   0.0   0:00.00 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
    6 root         0 -20      0      0        0 I   0.0   0.0   0:00.00 kworker/0:0H-
event+
...output omitted...
```

3. In the left terminal shell, verify the number of logical CPUs on the virtual machine. Run the `process101` script in the background.

- 3.1. Verify the number of logical CPUs.

```
[student@serverb ~]$ grep "model name" /proc/cpuinfo | wc -l
2
```

- 3.2. Change to the `/home/student/bin` directory. Run the `process101` script in the background.

```
[student@serverb ~]$ cd /home/student/bin
[student@serverb bin]$ process101 &
[1] 1161
```

4. In the right terminal shell, observe the `top` display. Note the process ID (PID), and view the CPU percentage that the `process101` process uses. The CPU percentage that the process

uses should hover around 10% to 15%. Toggle the `top` utility display between load, threads, and memory. Return to the CPU usage display of the `top` utility.

4.1. Press `Shift+m`.

```
top - 17:11:24 up 51 min,  2 users,  load average: 0.16, 0.07, 0.02
Tasks: 118 total,   1 running, 117 sleeping,   0 stopped,   0 zombie
%Cpu(s):  7.8 us,   0.7 sy,   0.0 ni, 91.2 id,   0.0 wa,   0.2 hi,   0.2 si,   0.0 st
MiB Mem :  1774.8 total,   1419.5 free,    207.4 used,    147.9 buff/cache
MiB Swap:    0.0 total,    0.0 free,    0.0 used.  1416.2 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
  761 root        20   0  340412 41416 17888 S   0.0   2.3   0:00.44 firewallld
  780 root        20   0  474344 30704 13508 S   0.0   1.7   0:00.62 tuned
  736 polkitd    20   0 2577132 24592 18320 S   0.0   1.4   0:00.07 polkitd
  767 root        20   0  471864 18992 16416 S   0.0   1.0   0:00.15 NetworkManager
    1 root        20   0 105972 17592 10292 S   0.0   1.0   0:01.30 systemd
...output omitted...
1161 student   20   0  222652   3888   3432 S  12.3   0.2   0:54.81 process101
...output omitted...
```



Note When the `top` utility switches into *memory* mode, the `process101` process is no longer the first process. You can press `Shift+p` to return to CPU usage.

4.2. Press `m` to display more memory details.

```
top - 17:16:14 up 56 min,  2 users,  load average: 0.20, 0.12, 0.04
Tasks: 118 total,   1 running, 117 sleeping,   0 stopped,   0 zombie
%Cpu(s):  7.5 us,   0.8 sy,   0.0 ni, 91.5 id,   0.0 wa,   0.2 hi,   0.0 si,   0.0 st
MiB Mem : 19.9/1774.8  [|||||||]
MiB Swap:  0.0/0.0    [          ]

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
  761 root        20   0  340412 41416 17888 S   0.0   2.3   0:00.44 firewallld
  780 root        20   0  474344 30704 13508 S   0.0   1.7   0:00.66 tuned
  736 polkitd    20   0 2577132 24592 18320 S   0.0   1.4   0:00.07 polkitd
  767 root        20   0  471864 18992 16416 S   0.0   1.0   0:00.15 NetworkManager
    1 root        20   0 105972 17592 10292 S   0.0   1.0   0:01.30 systemd
1068 student   20   0   21652  13144 10128 S   0.0   0.7   0:00.08 systemd
1114 root        20   0   19332  11928   9648 S   0.0   0.7   0:00.02 sshd
...output omitted...
1161 student   20   0  222652   3888   3432 S  11.0   0.2   1:35.17 process101
...output omitted...
```

4.3. Press `t`.

```
top - 17:21:43 up  1:01,  2 users,  load average: 0.23, 0.18, 0.09
Tasks: 121 total,   1 running, 120 sleeping,   0 stopped,   0 zombie
%Cpu(s):  7.5/1.0      8[|||||]
MiB Mem : 20.1/1774.8  [|||||||]
MiB Swap:  0.0/0.0    [          ]
```

```
PID USER      PR  NI     VIRT     RES     SHR S  %CPU  %MEM    TIME+  COMMAND
761 root        20   0   340412   41416   17888 S   0.0   2.3    0:00.44 firewallld
780 root        20   0   474344   30704   13508 S   0.0   1.7    0:00.70 tuned
736 polkitd     20   0   2577132  24592   18320 S   0.0   1.4    0:00.07 polkitd
767 root        20   0   471864   18992   16416 S   0.0   1.0    0:00.17 NetworkManager
  1 root        20   0   105972   17592   10292 S   0.0   1.0    0:01.31 systemd
1068 student    20   0    21652   13144   10128 S   0.0   0.7    0:00.08 systemd
1114 root        20   0    19332   11928    9648 S   0.0   0.7    0:00.02 sshd
 668 root        20   0    33656   11892    8728 S   0.0   0.7    0:00.10 systemd-udevd
1064 root        20   0    19328   11780    9504 S   0.0   0.6    0:00.03 sshd
...output omitted...
1155 student    20   0   225976    4400    3656 R   0.0   0.2    0:01.31 top
...output omitted...
```

4.4. Press **Shift+p** to switch to CPU usage.

```
top - 17:23:33 up 1:03,  2 users,  load average: 0.17, 0.17, 0.09
Tasks: 121 total,  1 running, 120 sleeping,   0 stopped,   0 zombie
%Cpu(s):  7.3/0.8   8[|||||]
MiB Mem : 20.2/1774.8  [|||||||||]
MiB Swap:  0.0/0.0   [
PID USER      PR  NI     VIRT     RES     SHR S  %CPU  %MEM    TIME+  COMMAND
1161 student    20   0   222652    3888    3432 S  15.6   0.2    2:09.61 process101
  1 root        20   0   105972   17592   10292 S   0.0   1.0    0:01.31 systemd
...output omitted...
```

5. Turn off the use of bold in the display. Save this configuration for reuse when **top** is restarted. Confirm that the changes are saved.

5.1. Press **Shift+b** to switch off the use of bold.

```
top - 17:29:12 up 1:09,  2 users,  load average: 0.17, 0.15, 0.10
Tasks: 117 total,  2 running, 115 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.6/0.7   6[||||]
MiB Mem : 20.4/1774.8  [|||||||||]
MiB Swap:  0.0/0.0   [
PID USER      PR  NI     VIRT     RES     SHR S  %CPU  %MEM    TIME+  COMMAND
1161 student    20   0   222652    3888    3432 R  12.0   0.2    2:57.18 process101
  1 root        20   0   105972   17592   10292 S   0.0   1.0    0:01.31 systemd
...output omitted...
```

5.2. Press **Shift+w** to save this configuration. The default configuration is stored in the **toprc** file in the **/home/student/.config/procps** directory. In the left terminal shell, confirm that the **toprc** file exists.

```
[student@serverb bin]$ ls -l /home/student/.config/procps/toprc
-rw-rw-r--. 1 student student 966 Feb 18 19:45 /home/student/.config/procps/toprc
```

5.3. In the right terminal shell, exit **top**, and then restart it. Confirm that the new display uses the saved configuration.

```
top - 17:51:48 up 1:31, 2 users, load average: 0.09, 0.12, 0.09
Tasks: 119 total, 1 running, 118 sleeping, 0 stopped, 0 zombie
%Cpu(s):  5.0/0.5   5[|||||]
MiB Mem : 20.0/1774.8  [|||||]
MiB Swap:  0.0/0.0    [

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1161	student	20	0	222652	3888	3432	S	10.6	0.2	6:08.76	process101
1	root	20	0	105972	17592	10292	S	0.0	1.0	0:01.33	systemd

```
...output omitted...
```

- Copy the process101 script to a new process102 file, and increase the artificial CPU load to one hundred thousand in the new script. Start the process102 process in the background.

6.1. In the left terminal shell, copy process101 to process102.

```
[student@serverb bin]$ cp process101 process102
```

6.2. Edit the process102 script and increase the addition calculations from fifty thousand to one hundred thousand. Enter interactive mode by using `i`. Type `:wq` to save the file and quit.

```
[student@serverb bin]$ vim process102
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 100000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

6.3. Start the process102 process in the background.

```
[student@serverb bin]$ process102 &
[2] 4023
```

6.4. Verify that both processes are running in the background.

```
[student@serverb bin]$ jobs
[1]-  Running                  process101 &
[2]+  Running                  process102 &
```

- In the right terminal shell, verify that the process is running and uses the most CPU resources. The load should hover between 25% to 35%.

7.1. In the right terminal shell, verify that the process is running. The load should hover between 25% to 35%.

```
top - 18:04:54 up 1:44, 2 users, load average: 0.37, 0.24, 0.13
Tasks: 120 total, 1 running, 119 sleeping, 0 stopped, 0 zombie
%Cpu(s): 18.1 us, 2.0 sy, 0.0 ni, 79.7 id, 0.0 wa, 0.2 hi, 0.0 si, 0.0 st
MiB Mem : 1774.8 total, 1374.3 free, 210.1 used, 190.4 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1410.7 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
 4023 student    20   0  222652   3980   3524 S   22.3   0.2   0:32.94 process102
 1161 student    20   0  222652   3888   3432 S   17.7   0.2   7:59.52 process101
    1 root       20   0  105972  17592  10292 S    0.0   1.0   0:01.33 systemd
...output omitted...
```

**Note**

If you do not see the `process101` and `process102` processes at the top of the process list, then press `Shift+p` to ensure that the `top` utility sorts the output by CPU usage.

8. Notice that the load average is below 1. Copy the `process101` script to a new script called `process103`. Increase the addition count to eight hundred thousand. Start `process103` in the background. Confirm that the load average is above 1. It might take a few minutes for the load average to change.

8.1. In the right terminal shell, verify that the load average is below 1.

```
top - 18:12:49 up 1:52, 2 users, load average: 0.45, 0.38, 0.24
...output omitted...
```

8.2. In the left terminal shell, copy `process101` to a new `process103` script.

```
[student@serverb bin]$ cp process101 process103
```

8.3. In the left terminal shell, edit the `process103` script. Increase the addition count to eight hundred thousand. Enter interactive mode with the `i` key. Type `:wq` to save the file and quit.

```
[student@serverb bin]$ vim process103
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 800000 ]]; do
        var=$((var+1))
    done
    sleep 1
done
```

8.4. Start the `process103` process in the background. The CPU usage hovers between 60% to 85%.

```
[student@serverb bin]$ process103 &
[3] 5172
```

8.5. Verify that all three jobs are running in the background.

```
[student@serverb bin]$ jobs
[1]   Running                  process101 &
[2]-  Running                  process102 &
[3]+  Running                  process103 &
```

8.6. In the right terminal shell, verify that the load average is above 1. It might take a few minutes for the load to increase.

```
top - 18:16:07 up 1:56, 2 users, load average: 1.11, 0.77, 0.45
...output omitted...
```

9. In the left terminal shell, switch to the root user. Suspend the process101 process. List the remaining jobs. Observe that the process state for process101 is now in the T state.

9.1. Switch to the root user.

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

9.2. Suspend the process101 process.

```
[root@serverb ~]# pkill -SIGSTOP process101
```

9.3. In the right terminal shell, confirm that the process101 process is no longer running.

```
top - 18:19:17 up 1:59, 2 users, load average: 0.92, 0.83, 0.50
Tasks: 123 total, 3 running, 118 sleeping, 1 stopped, 1 zombie
%Cpu(s): 42.9 us, 4.0 sy, 0.0 ni, 52.8 id, 0.0 wa, 0.3 hi, 0.0 si, 0.0 st
MiB Mem : 1774.8 total, 1368.4 free, 215.5 used, 190.8 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1405.2 avail Mem

  PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 5172 student    20   0  222652   3900   3448 R   66.4   0.2   3:25.81 process103
 4023 student    20   0  222652   3980   3524 R   26.9   0.2   4:07.89 process102
     1 root        20   0  105972  17592  10292 S    0.0   1.0   0:01.34 systemd
     2 root        20   0      0      0      0 S    0.0   0.0   0:00.00 kthreadd
...output omitted...
```

9.4. In the left terminal shell, view the remaining jobs.

```
[root@serverb ~]# ps jT
...output omitted...
PPID PID PGID SID TTY TPGID STAT UID TIME COMMAND
1117 1118 1118 1118 pts/1 5778 Ss 1000 0:00 -bash
1118 1161 1161 1118 pts/1 5778 T 1000 10:00 /bin/bash /home/student/bin/
process101
1118 4023 4023 1118 pts/1 5778 S 1000 4:19 /bin/bash /home/student/bin/
process102
1118 5172 5172 1118 pts/1 5778 S 1000 3:59 /bin/bash /home/student/bin/
process103
...output omitted...
```

Note that `process101` has a status of `T`. It means that the process is currently suspended.

10. Resume the `process101` process.

10.1. In the left terminal shell, resume the `process101` process.

```
[root@serverb ~]# pkill -SIGCONT process101
```

10.2. In the right terminal shell, verify that the process is running again.

```
top - 18:24:18 up 2:04, 2 users, load average: 1.06, 0.96, 0.65
Tasks: 125 total, 2 running, 123 sleeping, 0 stopped, 0 zombie
%Cpu(s): 48.3 us, 4.3 sy, 0.0 ni, 47.2 id, 0.0 wa, 0.2 hi, 0.0 si, 0.0 st
MiB Mem : 1774.8 total, 1368.6 free, 215.2 used, 191.0 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1405.5 avail Mem

    PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM     TIME+ COMMAND
  5172 student    20   0  222652  3900  3448 R   72.0   0.2   7:02.30 process103
  4023 student    20   0  222652  3980  3524 S   22.0   0.2   5:23.52 process102
  1161 student    20   0  222652  3888  3432 S   11.0   0.2  10:00.92 process101
...output omitted...
```

11. Terminate `process101`, `process102`, and `process103` from the command line. Verify that the processes are no longer displayed in `top`.

11.1. In the left terminal shell, terminate `process101`, `process102`, and `process103`.

```
[root@serverb ~]# pkill process101
[root@serverb ~]# pkill process102
[root@serverb ~]# pkill process103
```

11.2. In the right terminal shell, verify that the processes no longer appear in `top`.

```
top - 18:25:12 up 2:05, 2 users, load average: 0.93, 0.95, 0.67
Tasks: 117 total, 1 running, 116 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 : 1774.8 total, 1369.8 free, 214.0 used, 191.0 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 1406.7 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
767	root	20	0	471864	18992	16416	S	0.3	1.0	0:00.26	
NetworkManager											
1	root	20	0	105972	17592	10292	S	0.0	1.0	0:01.34	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
...output omitted...											

12. Stop the processes and return to the workstation machine.

12.1. Log out from the root user and close the terminal.

```
[root@serverb ~]# exit
logout
[1] Terminated process101
[2] Terminated process102
[3]- Terminated process103
```

12.2. In the right terminal shell, press q to quit top. 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 processes-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 processes-review
```

Summary

- A process is a running instance of an executable program. Processes are assigned a state, which can be running, sleeping, stopped, or zombie. The `ps` command lists processes.
- Each terminal has its own session, and can have a foreground process and independent background processes. The `jobs` command displays processes within a terminal session.
- A signal is a software interrupt that reports events to an executing program. The `kill`, `kill`, 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` commands.

Chapter 9

Control Services and Daemons

Goal

Control and monitor network services and system daemons with the `systemd` service.

Objectives

- List system daemons and network services that the `systemd` service and socket units started.
- Control system daemons and network services with the `systemctl` command.

Sections

- Identify Automatically Started System Processes (and Guided Exercise)
- Control System Services (and Guided Exercise)

Lab

- Control Services and Daemons

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. By convention, daemon names end with `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 try to start until the network is available.
- A method of tracking related processes together by using Linux control groups.

Service Units Description

A `systemd` *unit* is an abstract concept to define objects that the system knows how to manage.

Units are represented by configuration files called *unit files*, which encapsulate information about system services, listening sockets, and other relevant objects for the `systemd` init system.

A unit has a name and a unit type. The name provides a unique identity to the unit. The unit type enables grouping units together with other similar unit types.

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 often-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 often 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 correctly 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-rpccss-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
auditd.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
auditd.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 is waiting for an event.
inactive	The service is not running.
enabled	The service starts at boot time.

Keyword	Description
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 the state 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 correctly running, or `failed` if an error occurred during startup. If the unit was stopped, then 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

► Guided Exercise

Identify Automatically Started System Processes

List installed service units and identify which services are currently enabled and active on a server.

Outcomes

- List the 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 the 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 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/ssh.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/ssh.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
daemon...
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:M8ikhcEDm2tQ95Z0o7ZvufqEixCFct
+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
auditd.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
```

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 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, the `systemd` and `systemctl` commands 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 stop all 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, some installed services on your system might conflict with each other. For example, many ways exist to manage mail servers (the `postfix` and `sendmail` services). Masking a service prevents an administrator from accidentally starting a service that conflicts with other services. 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 mask` option.

```
[root@host ~]# systemctl mask sendmail.service
Created symlink /etc/systemd/system/sendmail.service → /dev/null.
```

Then, verify 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 does 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 prevent 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 when 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 UNIT</code>	View detailed information about a unit's state.
<code>systemctl stop UNIT</code>	Stop a service on a running system.
<code>systemctl start UNIT</code>	Start a service on a running system.
<code>systemctl restart UNIT</code>	Restart a service on a running system.
<code>systemctl reload UNIT</code>	Reload the configuration file of a running service.
<code>systemctl mask UNIT</code>	Disable a service from being started, both manually and at boot.
<code>systemctl unmask UNIT</code>	Make available a masked service.
<code>systemctl enable UNIT</code>	Configure a service to start at boot time. Use the <code>--now</code> option to also start the service.
<code>systemctl disable UNIT</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

► Guided Exercise

Control System Services

Use the `systemctl` command to stop, start, restart, reload, enable, and disable the `systemd` services.

Outcomes

- Use the `systemctl` command to control the `systemd` 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 the `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
```

► Lab

Control Services and Daemons

Configure several services to be enabled or disabled, and started or stopped, based on a specification that is provided to you.

Outcomes

- Enable, disable, start, and stop 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-review
```

Instructions

1. Use the `ssh` command to log in to the `serverb` machine as the `student` user. On the `serverb` machine, start the `psacct` service.
2. Configure the `psacct` service to start at system boot.
3. Stop the `rsyslog` service.
4. Configure the `rsyslog` service so that it does not start at system boot.
5. Reboot the `serverb` machine before evaluating the lab.

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 services-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 services-review
```

► Solution

Control Services and Daemons

Configure several services to be enabled or disabled, and started or stopped, based on a specification that is provided to you.

Outcomes

- Enable, disable, start, and stop 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-review
```

Instructions

1. Use the `ssh` command to log in to the `serverb` machine as the `student` user. On the `serverb` machine, start the `psacct` service.

- 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. Verify the status of the `psacct` service. The `psacct` service is in an inactive and disabled state. The disabled state confirms that the service does not start at boot time.

```
[root@serverb ~]# systemctl status psacct
○ psacct.service - Kernel process accounting
   Loaded: loaded (/usr/lib/systemd/system/psacct.service; disabled; vendor
   preset: disabled)
   Active: inactive (dead)
```

- 1.3. Start the `psacct` service.

```
[root@serverb ~]# systemctl start psacct
```

- 1.4. Verify that the `psacct` service is running.

```
[root@serverb ~]# systemctl is-active psacct
active
```

2. Configure the `psacct` service to start at system boot.

- 2.1. Enable the `psacct` service to start at system boot.

```
[root@serverb ~]# systemctl enable psacct
Created symlink /etc/systemd/system/multi-user.target.wants/psacct.service → /usr/lib/systemd/system/psacct.service.
```

- 2.2. Verify that the `psacct` service is enabled to start at system boot.

```
[root@serverb ~]# systemctl is-enabled psacct
enabled
```

3. Stop the `rsyslog` service.

- 3.1. Verify the status of the `rsyslog` service. Notice that the `rsyslog` service is running and enabled to start at boot time. Press `q` to exit the command.

```
[root@serverb ~]# systemctl status rsyslog
● rsyslog.service - System Logging Service
   Loaded: loaded (/usr/lib/systemd/system/rsyslog.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-05-19 04:04:43 EDT; 38min ago
   ...output omitted...
```

- 3.2. Stop the `rsyslog` service.

```
[root@serverb ~]# systemctl stop rsyslog
```

- 3.3. Verify that the `rsyslog` service is stopped.

```
[root@serverb ~]# systemctl is-active rsyslog
inactive
```

4. Configure the `rsyslog` service so that it does not start at system boot.

- 4.1. Disable the `rsyslog` service so that it does not start at system boot.

```
[root@serverb ~]# systemctl disable rsyslog
Removed /etc/systemd/system/multi-user.target.wants/rsyslog.service.
```

- 4.2. Verify that the `rsyslog` service does not start during the boot process.

```
[root@serverb ~]# systemctl is-enabled rsyslog
disabled
```

5. Reboot the `serverb` machine before evaluating the lab.

```
[root@serverb ~]# systemctl reboot
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 services-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 services-review
```

Summary

- The `systemd` utility provides a method for activating system resources, server daemons, and other processes, both at boot time and on a running system.
- 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 the `systemd` utility 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.

Chapter 10

Configure and Secure SSH

Goal

Configure secure command-line service on remote systems with OpenSSH.

Objectives

- Log in to a remote system and run commands with ssh.
- Configure a user account to use key-based authentication to log in to remote systems securely without a password.
- Disable direct logins as `root` and password-based authentication for the OpenSSH service.

Sections

- Access the Remote Command Line with SSH (and Guided Exercise)
- Configure SSH Key-based Authentication (and Guided Exercise)
- Customize OpenSSH Service Configuration (and Guided Exercise)

Lab

- Configure and Secure SSH

Access the Remote Command Line with SSH

Objectives

- Log in to a remote system and run commands with `ssh`.

Describe Secure Shell

The OpenSSH package provides the Secure Shell, or SSH protocol, in Red Hat Enterprise Linux. With SSH protocol, systems can communicate in an encrypted and secure channel over an insecure network.

Use the `ssh` command to create a secure connection to a remote system, authenticate as a specific user, and obtain an interactive shell session on the remote system. The `ssh` command can run a session on a remote system without running an interactive shell.

Secure Shell Examples

The following `ssh` command logs you in on the `hosta` remote server by using the same username as the current local user. The remote system prompts you to authenticate with the `developer1` user's password in this example.

```
[developer1@host ~]$ ssh hosta
developer1@hosta's password: redhat
...output omitted...
[developer1@hosta ~]$
```

Use the `exit` command to log out of the remote system.

```
[developer1@hosta ~]$ exit
logout
Connection to hosta closed.
[developer1@host ~]$
```

The following `ssh` command logs you in on the `hosta` remote server with the `developer2` username. The remote system prompts you to authenticate with the `developer2` user's password.

```
[developer1@host ~]$ ssh developer2@hosta
developer2@hosta's password: shadowman
...output omitted...
[developer2@hosta ~]$
```

The following `ssh` command runs the `hostname` command on the `hosta` remote system as the `developer2` user without accessing the remote interactive shell.

```
[developer1@host ~]$ ssh developer2@hosta hostname
developer2@hosta's password: shadowman
hosta.lab.example.com
[developer1@host ~]$
```

This command displays the output in the local system's terminal.

Identifying Remote Users

The `w` command displays a list of users that are currently logged in to the system. It also displays the remote system location and commands that the user ran.

```
[developer1@host ~]$ ssh developer1@hosta
developer1@hosta's password: redhat
[developer1@hosta ~]$ w -f
16:13:38 up 36 min, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
developer2 pts/0    172.25.250.10   16:13    7:30   0.01s  0.01s -bash
developer1 pts/1    172.25.250.10   16:24    3.00s  0.01s  0.00s w
[developer2@hosta ~]$
```

The output shows that the `developer2` user logged in to the system on the pseudo-terminal 0 at 16:13 today from the host with the 172.25.250.10 IP address, and has been idle at a shell prompt for seven minutes and thirty seconds. The output also shows that the `developer1` user logged in to the system on the pseudo-terminal 1, and has been idle for three seconds after executing the `w` command.

SSH Host Keys

SSH secures communication through public-key encryption. When an SSH client connects to an SSH server, the server sends a copy of its public key to the client before logging in. This key helps to set up secure encryption for the communication channel and to authenticate the client's system.

When a user runs the `ssh` command for connecting to an SSH server, the command checks for a copy of the server's public key in its local known hosts file. The key might be preconfigured in the `/etc/ssh/ssh_known_hosts` file, or the user might have the `~/.ssh/known_hosts` file that contains the key in their home directory.

If the client has a copy of the key, then the `ssh` command compares the key from the known hosts server files to the key that it received. If the keys do not match, then the client assumes that the network traffic to the server is compromised, and prompts the user to confirm whether to continue with the connection.

Strict Host Key Checking

The `StrictHostKeyChecking` parameter is set in the user-specific `~/.ssh/config` file, or in the system-wide `/etc/ssh/ssh_config` file, or by specifying the `-o StrictHostKeyChecking=` option of the `ssh` command.

- If the `StrictHostKeyChecking` parameter is set to `yes`, then the `ssh` command always aborts the SSH connection if the public keys do not match.
- If the `StrictHostKeyChecking` parameter is set to `no`, then the `ssh` command enables the connection and adds the key of the target host to the `~/.ssh/known_hosts` file.

If the target host SSH key changed since the last time that you connected to it, then the `ssh` command asks for confirmation to log in and accept the new key.

If you accept the new key, then a copy of the public key is saved in the `~/.ssh/known_hosts` file to automatically confirm the server's identity on subsequent connections.

**Note**

Red Hat recommends to set the `StrictHostKeyChecking` parameter to `yes` in the user-specific `~/.ssh/config` file or in the system-wide `/etc/ssh/ssh_config` file, so that the `ssh` command always aborts the SSH connection if the public keys do not match.

```
[developer1@host ~]$ ssh hostb
The authenticity of host 'hostb (172.25.250.12)' can't be established.
ECDSA key fingerprint is SHA256:qaS0PTOLrqlC02XGklA0iY7CaP7aPKimerDoaUkv720.
Are you sure you want to continue connecting (yes/no)? no
[developer1@host ~]$
```

Verify the fingerprint of the target server's SSH host key by using the `ssh-keygen` command. In this example, the `ssh-keygen` command is run on the `hostb` target server.

The `ssh-keygen` command displays the key fingerprint so that you can match it to the output of the `ssh` command and verify that the key is valid. Use the `-lf` options to list the public key fingerprint in the host's default public key file.

Because you cannot connect over SSH, you must verify the target host's key fingerprint by logging in locally. Use an out-of-band communication method to share public keys, such as a phone call or video conference.

```
[developer1@hostb ~]$ ssh-keygen -lf /etc/ssh/ssh_host_ecdsa_key.pub
256 SHA256:qaS0PTOLrqlC02XGklA0iY7CaP7aPKimerDoaUkv720 root@server (ECDSA)
```

After you verify the key on the target host, you can accept the key and connect to the target host.

```
[developer1@host ~]$ ssh hostb
The authenticity of host 'hostb (172.25.250.12)' can't be established.
ECDSA key fingerprint is SHA256:qaS0PTOLrqlC02XGklA0iY7CaP7aPKimerDoaUkv720.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'hostb,172.25.250.12' (ECDSA) to the list of known
hosts.
developer1@hostb's password: redhat
...output omitted...
[developer1@hostb ~]$
```

SSH Known Hosts Key Management

Information about known remote systems and their keys are stored in either of the following places:

- The system-wide `/etc/ssh/ssh_known_hosts` file.
- The `~/.ssh/known_hosts` file in each user's home directory.

The `/etc/ssh/ssh_known_hosts` file is a system-wide file that stores the public keys for hosts that the system knows. You must create and manage this file, either manually or through some automated method such as by using Ansible or a script that uses the `ssh-keyscan` utility.

A server's public key might have changed because the key was lost due to hard drive failure or it was replaced for some legitimate reason. In that case, to successfully log in to that system, the `/etc/ssh/ssh_known_hosts` file must be modified to replace the previous public key entry with the new public key.

If you connect to a remote system and the public key of that system is not in the `/etc/ssh/ssh_known_hosts` file, then the SSH client searches for the key in the `~/.ssh/known_hosts` file.

Each known host key entry consists of one line that contains three fields:

- The first field is the list of hostnames and IP addresses that share the public key.
- The second field is the encryption algorithm that is used for the key.
- The last field is the key itself.

```
[developer1@host ~]$ cat ~/.ssh/known_hosts
server1 ssh-ed25519
    AAAAC3NzaC1lZDI1NTE5AAAAIOmILKMEXRnsS1g7OTxMs0mgHuUSGQBUXHhuUGcv19uT
server1 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC8WD0ooY+rh6NPa9yhLsNQXBqcQknTL/
WSd3zPvHLLd7KaC4IiEUxnwbflBit8tRcirbQFx020Am
...output omitted...
```

Troubleshooting Host Key Issues

If the IP address or the public key of the remote system changes, and you try to connect to that system again via SSH, then the SSH client detects that the key entry for that system in the `~/.ssh/known_hosts` file is no longer valid. A warning message states that the remote host identification changed, and that you must modify the key entry.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:hxttxb/qVi1/ycUU2wXF6mfGH++Ya7WYZv0r+tIkq4I.
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/user/.ssh/known_hosts:12
ECDSA host key for server1.example.com has changed and you have requested strict
checking.
Host key verification failed.
```

If you do not know why the key changed, then verify the new key fingerprint, because this key might be an actual attack on your network. Use an out-of-band method for verification, such as speaking with the system administrator of the target system.

If you know why the key changed, such as an IP address change, then resolve this problem by removing the relevant key entry from the `~/.ssh/known_hosts` file, and then reconnect to the system to receive the new key entry.

The line number of the relevant key entry is specified in the warning message. You can also find and remove the relevant key entry by running the following command:

```
[developer1@host ~]$ ssh-keygen -R remotesystemname -f ~/.ssh/known_hosts
# Host remotesystemname found: line 12
/home/user/.ssh/known_hosts updated.
Original contents retained as /home/user/.ssh/known_hosts.old
```



References

ssh(1), w(1), and hostname(1) man pages

For more information, refer to *Using Secure Communications Between Two Systems with OpenSSH* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/securing_networks/assembly_using-secure-communications-between-two-systems-with-openssh_securing-networks

► Guided Exercise

Access the Remote Command Line

Log in to a remote system as different users and execute commands.

Outcomes

- Log in to a remote system.
- Execute commands with the OpenSSH secure shell.

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-access
```

Instructions

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

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

- 2. Open an SSH session to the `serverb` machine as the `student` user. Accept the host key. Use `student` as the password when prompted for the password of the `student` user on the `serverb` machine.

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ED25519 key fingerprint is SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KW0TK80.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'serverb' (ED25519) to the list of known hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

The `ssh` command records the host key on the `/home/student/.ssh/known_hosts` file in the `servera` machine to identify the `serverb` machine. The `student` user initiated the SSH connection from the `servera` machine. If the `/home/student/.ssh/known_hosts` file does not exist, then it is created along with the new entry in it. The `ssh` command fails to execute correctly if the remote host appears to have a different key from the recorded key.

- ▶ 3. Display the users that are currently logged in to the `serverb` machine. The `student` user is logged in to the system from the host with an IP address of `172.25.250.10`, which is the `servera` machine in the classroom network.

```
[student@serverb ~]$ w --from
03:39:04 up 16 min,  1 user,  load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
student   pts/0    172.25.250.10    20:40    1.00s  0.01s  0.00s w --from
```

- ▶ 4. Exit the `student` user's shell on the `serverb` machine.

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

- ▶ 5. Open an SSH session to the `serverb` machine as the `root` user. Use `redhat` as the password of the `root` user. The command did not ask you to accept the host key, because it was found among the known hosts. If the identity of the `serverb` machine changes, then OpenSSH prompts you to challenge the new host key.

```
[student@servera ~]$ ssh root@serverb
root@serverb's password: redhat
...output omitted...
[root@serverb ~]#
```

- ▶ 6. Run the `w` command to display the users that are currently logged in to the `serverb` machine. The output indicates that the `root` user is logged in to the system from the host with an IP address of `172.25.250.10`, which is the `servera` machine in the classroom network.

```
[root@serverb ~]# w --from
03:46:05 up 23 min,  1 user,  load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
root      pts/0    172.25.250.10    20:44    1.00s  0.02s  0.00s w --from
```

- ▶ 7. Exit the `root` user's shell on the `serverb` machine.

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

- ▶ 8. Remove the `/home/student/.ssh/known_hosts` file from the `servera` machine. This operation causes `ssh` to lose the recorded identities of the remote systems.

```
[student@servera ~]$ rm /home/student/.ssh/known_hosts
```

Host keys can change for legitimate reasons: perhaps the remote machine was replaced because of a hardware failure, or the remote machine was reinstalled. Usually, it is advisable

to remove the key entry only for the particular host in the `known_hosts` file. Because this particular `known_hosts` file has only one entry, you can remove the entire file.

- ▶ 9. Open an SSH session to the `serverb` machine as the `student` user. If asked, accept the host key. Use `student` when prompted for the password of the `student` user on the `serverb` machine.

```
[student@servera ~]$ ssh student@serverb
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ED25519 key fingerprint is SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KW0TK80.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'serverb' (ED25519) to the list of known hosts.
student@serverb's password: student
...output omitted...
[student@serverb ~]$
```

The `ssh` command asked for your confirmation to accept or reject the host key, because it could not find one for the remote host.

- ▶ 10. Exit the `student` user's shell on the `serverb` machine, and confirm that a new instance of the `known_hosts` file exists on the `servera` machine.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@servera ~]$ ls -l /home/student/.ssh/known_hosts
-rw-----. 1 student student 819 Mar 24 03:47 /home/student/.ssh/known_hosts
```

- ▶ 11. Confirm that the new instance of the `known_hosts` file has the host key of the `serverb` machine. The following command output is an example; your `workstation` output might differ.

```
[student@servera ~]$ cat /home/student/.ssh/known_hosts
...output omitted...
serverb ecdsa-sha2-nistp256 AAAAB3NzaC1yc2EAAAADAQ...
...output omitted...
```

- ▶ 12. Run the `hostname` command remotely on the `serverb` machine without accessing the interactive shell.

```
[student@servera ~]$ ssh student@serverb hostname
student@serverb's password: student
serverb.lab.example.com
```

- ▶ 13. 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 ssh-access
```

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 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. The second key is your related public key, which 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 remote server uses your public key to encrypt a challenge message and send it to your SSH client. Your SSH client must then prove that it can decrypt this message, which demonstrates that you have the associated private key. If this verification succeeds, then your request is trusted, and you are granted 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:vxutUNPio3QDCyvkYm1 user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]-----+
|
| . .
| o o o
| . = o o .
| o + = S E .
| ..0 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 to access it. If you set a passphrase, then you must enter the passphrase each time that 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 on 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:w3GGB7EyHURY4aOcNPKmhNKS7dl1YsMVLvFZJ77VxAo user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]-----+
|      . + =.0 ... |
|      = B XEo o. |
|      . o 0 X =... |
|  =  = B = o. |
| = + * * S . |
|. + = o + . |
|   + . |
|   |
|   |
+-----[SHA256]-----+
```

The `ssh-keygen` command `-f` option specifies the files to save the keys in. 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 overwrite that existing key pair, which can be restored only if you have a backup for those files. Overwriting a key pair loses the original private key that is required to access accounts that you configured with the corresponding public key on remote servers.

If you cannot restore your local private key, then you 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, if they are overwritten or lost.

Generated SSH keys are stored by default in the `.ssh` subdirectory of your home directory. To function correctly, the private key must be readable and writable only by the user that it belongs to (octal permissions 600). The public key is not secure, and anyone on the system might also be able to read it (octal permissions 644).

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.
```

After you place the public key, test the remote access, with the corresponding private key. If the configuration is correct, you access 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, then SSH requests the passphrase on first use. However, if the key authentication succeeds, then you are not 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 that 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 is 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, then 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 key manager by 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 to 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 with key pair authentication is not succeeding. The `ssh` command provides three verbosity levels with the `-v`, `-vv`, and `-vvv` options, which respectively provide increasing debugging information during `ssh` command use.

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

```
[user@host ~]$ ssh -v user@remotehost
OpenSSH_8.7p1, OpenSSL 3.0.1 14 Dec 2021 ❶
debug1: Reading configuration data /etc/ssh/ssh_config ❷
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. ❸
```

```

debug1: Connection established.
...output omitted...
debug1: Authenticating to remotesHost:22 as 'user' ❹
...output omitted...
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-
mic,password ❺
...output omitted...
debug1: Next authentication method: publickey ❻
debug1: Offering public key: /home/user/.ssh/id_rsa RSA
SHA256:hDVJjD7xrUjXGZVRJQixxFV6NF/ssMjS6AuQ1+VqUc4 ❼
debug1: Server accepts key: /home/user/.ssh/id_rsa RSA
SHA256:hDVJjD7xrUjXGZVRJQixxFV6NF/ssMjS6AuQ1+VqUc4 ❽
Authenticated to remotesHost ([192.168.1.10]:22) using "publickey".
...output omitted...
[user@remotesHost ~]$

```

- ❶ OpenSSH and OpenSSL versions.
- ❷ OpenSSH configuration files.
- ❸ Connection to the remote host.
- ❹ Trying to authenticate the user on the remote host.
- ❺ Authentication methods that the remote host allows.
- ❻ Trying to authenticate the user by using the SSH key.
- ❼ Using the `/home/user/.ssh/id_rsa` key file to authenticate.
- ❽ The remote hosts accepts the SSH key.

If an attempted authentication method fails, then a remote SSH server falls 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@remotesHost
...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@remotesHost's password: password
Authenticated to remotesHost ([172.25.250.10]:22) using "password".
...output omitted...
[user@remotesHost ~]$

```

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
```



References

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

► Guided Exercise

Configure SSH Key-based Authentication

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+*+000 .      |
|. = 0.0 0 .      |
|0.. = . . 0      |
|+. + * . 0 .      |
|+ = X . S +      |
| + @ + = .      |
|. + = 0          |
|.0 . . . .      |
|0      0..      |
+-----[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/anxp6AP7BmB5azIPVbPNqieh0oKi4KW0TK80.
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 in to 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.

In a following step in this exercise, you make your private key more secure by encrypting it and protecting access to it by adding a passphrase.

- ▶ 6. Generate another set of SSH keys with the default name and without a passphrase, and overwrite 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, and overwrite 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 in to 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:
SHA256:0CtCjfPm5QrbPBgqb operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]-----+
|O=X*                |
|OB=.                |
|E*o.               |
|B000 .             |
|..= . o S          |
|+.o  o             |
|+.00+ o            |
|+0.0.+             |
|+ . =0.            |
+-----[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 in to 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 passphrases.

```
[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 when 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 by 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
```

Customize OpenSSH Service Configuration

Objectives

- Disable direct logins as `root` and password-based authentication for the OpenSSH service.

Configure the OpenSSH Server

The `sshd` daemon provides the OpenSSH service. You can configure the service by editing the `/etc/ssh/sshd_config` file.

The default configuration of the OpenSSH server works well for many use cases. However, you might want to make some changes to strengthen the security of your system. You might want to prohibit direct remote login to the `root` account, and you might want to prohibit password-based authentication (in favor of SSH private key authentication).

Prohibit the Superuser from Logging In

It is a good practice to prohibit direct login to the `root` user account from remote systems. Some risks of allowing direct login as the `root` user include the following cases:

- The `root` username exists on every Linux system by default, so a potential attacker needs only to guess the password, instead of a valid username and password combination. This scenario reduces complexity for an attacker.
- The `root` user has unrestricted privileges, so its compromise can lead to maximum damage to the system.
- From an auditing perspective, it can be hard to track which authorized user logged in as the `root` user and made changes. If users must log in as a regular user and switch to the `root` account, then you can view a log event for accountability.



Important

Starting in Red Hat Enterprise Linux 9, the `PermitRootLogin` parameter is set to the `prohibit-password` value by default. This value enforces the use of key-based authentication instead of passwords for logging in as the `root` user, and reduces the risk of brute-force attacks.

The OpenSSH server uses the `PermitRootLogin` configuration setting in the `/etc/ssh/sshd_config` file to allow or prohibit users to log in to the system as the `root` user, as in the following example:

```
PermitRootLogin yes
```

If the `PermitRootLogin` parameter is set to the `yes` value, then anyone can log in as the `root` user remotely. To prevent this situation, set the value to `no`. Alternatively, to prevent password-based authentication but to allow private key-based authentication for `root`, set the `PermitRootLogin` parameter to `without-password`.

The SSH server (`sshd`) must be reloaded to apply any changes.

```
[root@host ~]# systemctl reload sshd
```

Prohibit Password-based Authentication for SSH

Allowing only private key-based logins to the remote command line has advantages:

- Attackers cannot use password-guessing attacks to remotely break into known accounts on the system.
- With passphrase-protected private keys, an attacker needs both the passphrase and a copy of the private key. With passwords, an attacker needs only the password.
- By using passphrase-protected private keys with `ssh-agent`, the passphrase is entered and exposed less often, and logging in is more convenient for the user.

The OpenSSH server uses the `PasswordAuthentication` parameter in the `/etc/ssh/sshd_config` file to control whether users can use password-based authentication to log in to the system.

```
PasswordAuthentication yes
```

With the default value of `yes` for the `PasswordAuthentication` parameter in the `/etc/ssh/sshd_config` file, the SSH server allows users to use password-based authentication when logging in. The value of `no` for `PasswordAuthentication` prevents users from using password-based authentication.

Whenever you change the `/etc/ssh/sshd_config` file, you must reload the `sshd` service to apply the changes.



Important

If you turn off password-based authentication for `ssh`, then you must ensure that the user's `~/.ssh/authorized_keys` file on the remote server is populated with their public key, so that they can log in.



References

`ssh(1)`, `sshd_config(5)` man pages

► Guided Exercise

Customize OpenSSH Service Configuration

Disable direct logins as `root` and disable password-based authentication for the OpenSSH service on one of your servers.

Outcomes

- Disable direct logins as `root` over `ssh`.
- Disable password-based authentication for remote users to connect over `SSH`.

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 ssh-customize
```

Instructions

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

```
[student@workstation ~]$ ssh student@serverb
[student@serverb ~]$
```

- 2. Use the `su` command to switch to the `operator2` user on the `serverb` machine. Use `redhat` as the password for the `operator2` user.

```
[student@serverb ~]$ su - operator2
Password: redhat
[operator2@serverb ~]$
```

- 3. Use the `ssh-keygen` command to generate `SSH` keys. Do not enter any passphrase for the keys.

```
[operator2@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operator2/.ssh/id_rsa): Enter
Created directory '/home/operator2/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/operator2/.ssh/id_rsa.
Your public key has been saved in /home/operator2/.ssh/id_rsa.pub.
The key fingerprint is:
```

```
SHA256:LN5x1irX00Wxgyd/qhATNgZW0tLUj16EZkM1JHkCR+I
operator2@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]-----+
|      *+=      |
|      = 0.o.   |
|      . Eo=B  o |
|      o +.=0+  o |
|      . S..= =  |
|      . o +. + . |
|      . o + . . .|
|      o .  o    |
|      ...      |
+-----[SHA256]-----+
```

- 4. Use the `ssh-copy-id` command to send the public key of the SSH key pair to the `operator2` user on the `servera` machine. Use `redhat` as the password for the `operator2` user on `servera`.

```
[operator2@serverb ~]$ ssh-copy-id operator2@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
operator2/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be established.
ED25519 key fingerprint is SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KW0TK80.
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
operator2@servera's password: redhat
Number of key(s) added: 1
Now try logging into the machine, with: "ssh 'operator2@servera'"
and check to make sure that only the key(s) you wanted were added.
```

- 5. Confirm that you can successfully log in to the `servera` machine as the `operator2` user with the SSH keys.

5.1. Open an SSH session to the `servera` machine as the `operator2` user.

```
[operator2@serverb ~]$ ssh operator2@servera
...output omitted...
[operator2@servera ~]$
```

The preceding `ssh` command used SSH keys for authentication.

5.2. Log out of the `servera` machine.

```
[operator2@servera ~]$ exit
logout
Connection to servera closed.
```

- 6. Confirm that you can successfully log in to the `servera` machine as the `root` user with `redhat` as the password.

- 6.1. Open an SSH session to the `servera` machine as the `root` user with `redhat` as the password.

```
[operator2@serverb ~]$ ssh root@servera
root@servera's password: redhat
...output omitted...
[root@servera ~]#
```

The preceding `ssh` command used the password of the superuser for authentication, because SSH keys do not exist for the superuser.

- 6.2. Log out of the `servera` machine.

```
[root@servera ~]# exit
logout
Connection to servera closed.
[operator2@serverb ~]$
```

- ▶ 7. Confirm that you can successfully log in to the `servera` machine as the `operator3` user with `redhat` as the password.

- 7.1. Open an SSH session to the `servera` machine as the `operator3` user with `redhat` as the password.

```
[operator2@serverb ~]$ ssh operator3@servera
operator3@servera's password: redhat
...output omitted...
[operator3@servera ~]$
```

The preceding `ssh` command used the password of the `operator3` user for authentication, because SSH keys do not exist for the `operator3` user.

- 7.2. Log out of the `servera` machine.

```
[operator3@servera ~]$ exit
logout
Connection to servera closed.
[operator2@serverb ~]$
```

- ▶ 8. Configure the `sshd` service on the `servera` machine to prevent users from logging in as the `root` user. Use `redhat` as the password of the superuser when required.

- 8.1. Open an SSH session to the `servera` machine as the `operator2` user with the SSH keys.

```
[operator2@serverb ~]$ ssh operator2@servera
...output omitted...
[operator2@servera ~]$
```

- 8.2. On the `servera` machine, switch to the `root` user. Use `redhat` as the password for the `root` user.

```
[operator2@servera ~]$ su -
Password: redhat
[root@servera ~]#
```

- 8.3. Set `PermitRootLogin` to `no` in the `/etc/ssh/sshd_config` file and reload the `sshd` service. You can use the `vim /etc/ssh/sshd_config` command to edit the configuration file of the `sshd` service.

```
...output omitted...
PermitRootLogin no
...output omitted...
[root@servera ~]# systemctl reload sshd
```

- 8.4. Open another terminal on `workstation`, and open an SSH session to the `serverb` machine as the `operator2` user. From the `serverb` machine, try to log in to the `servera` machine as the `root` user. This command should fail, because you disabled the `root` user login over SSH in the preceding step.

**Note**

For your convenience, password-less login is already configured between `workstation` and `serverb` in the classroom environment.

```
[student@workstation ~]$ ssh operator2@serverb
...output omitted...
[operator2@serverb ~]$ ssh root@servera
root@servera's password: redhat
Permission denied, please try again.
root@servera's password: redhat
Permission denied, please try again.
root@servera's password: redhat
root@servera: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

By default, the `ssh` command attempts to authenticate with key-based authentication first, and if that method fails, then with password-based authentication.

- ▶ 9. Configure the `sshd` service on the `servera` machine to allow users to authenticate with SSH keys only, rather than with their passwords.

- 9.1. Return to the first terminal with the `root` user's active shell on the `servera` machine. Set the `PasswordAuthentication` parameter to `no` in the `/etc/ssh/sshd_config` file and reload the `sshd` service. You can use the `vim /etc/ssh/sshd_config` command to edit the configuration file of the `sshd` service.

```
...output omitted...
PasswordAuthentication no
...output omitted...
[root@servera ~]# systemctl reload sshd
```

- 9.2. Go to the second terminal with the `operator2` user's active shell on the `serverb` machine, and try to log in to the `servera` machine as the `operator3` user. This command should fail, because SSH keys are not configured for the `operator3` user, and the `sshd` service on the `servera` machine does not allow the use of passwords for authentication.

```
[operator2@serverb ~]$ ssh operator3@servera
operator3@servera: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

**Note**

For more granularity, you can use the explicit `-o PubkeyAuthentication=no` and `-o PasswordAuthentication=yes` options with the `ssh` command. You can then override the `ssh` command's defaults and confidently determine that the preceding command fails based on the settings that you adjusted in the `/etc/ssh/sshd_config` file in the preceding step.

- 9.3. Return to the first terminal with the `root` user's active shell on the `servera` machine. Verify that `PubkeyAuthentication` is enabled in the `/etc/ssh/sshd_config` file. You can use the `vim /etc/ssh/sshd_config` command to view the configuration file of the `sshd` service.

```
...output omitted...
#PubkeyAuthentication yes
...output omitted...
```

The `PubkeyAuthentication` line is commented. Commented lines indicate the default values of a parameter. The public key authentication of SSH is active by default, as the commented line indicates.

- 9.4. Return to the second terminal with the `operator2` user's active shell on the `serverb` machine, and try to log in to the `servera` machine as the `operator2` user. This command should succeed, because the SSH keys are configured for the `operator2` user to log in to the `servera` machine from the `serverb` machine.

```
[operator2@serverb ~]$ ssh operator2@servera
...output omitted...
[operator2@servera ~]$
```

- 9.5. From the second terminal, exit the `operator2` user's shell on both the `servera` and `serverb` machines.

```
[operator2@servera ~]$ exit
logout
Connection to servera closed.
[operator2@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

- 9.6. Close the second terminal on the `workstation` machine.

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

9.7. From the first terminal, exit the `root` user's shell on the `servera` machine.

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

9.8. From the first terminal, exit the `operator2` user's shell on both the `servera` and `serverb` machines.

```
[operator2@servera ~]$ exit
logout
Connection to servera closed.
[operator2@serverb ~]$ exit
logout
[student@serverb ~]$
```

9.9. Log out of `serverb`, and return to the `student` user's shell on `workstation`.

```
[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-customize
```

► Lab

Configure and Secure SSH

Set up key-based authentication for users, and disable direct login as `root` and password authentication for all users for the OpenSSH service on one of your servers.

Outcomes

- Authenticate with SSH keys.
- Prevent users from directly logging in as the `root` user over the `ssh` service.
- Prevent users from logging in to the system with SSH password-based authentication.

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-review
```

Instructions

1. From the `workstation` machine, log in to the `servera` machine as the `student` user.
2. Switch to the `production1` user on the `servera` machine. Enter `redhat` as the password.
3. Generate passphrase-less SSH keys for the `production1` user on the `servera` machine.
4. Send the public key of the SSH key pair to the `production1` user on the `serverb` machine.
5. Verify that the `production1` user can successfully log in to the `serverb` machine with the SSH keys.
6. Configure the `sshd` service on `serverb` to prevent users from logging in as the `root` user. Use `redhat` as the `root` password.
7. Configure the `sshd` service on the `serverb` machine to allow users to authenticate with SSH keys only, rather than with their passwords.

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 ssh-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 ssh-review
```


► Solution

Configure and Secure SSH

Set up key-based authentication for users, and disable direct login as `root` and password authentication for all users for the OpenSSH service on one of your servers.

Outcomes

- Authenticate with SSH keys.
- Prevent users from directly logging in as the `root` user over the `ssh` service.
- Prevent users from logging in to the system with SSH password-based authentication.

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-review
```

Instructions

1. From the `workstation` machine, log in to the `servera` machine as the `student` user.

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

2. Switch to the `production1` user on the `servera` machine. Enter `redhat` as the password.

```
[student@servera ~]$ su - production1
Password: redhat
[production1@servera ~]$
```

3. Generate passphrase-less SSH keys for the `production1` user on the `servera` machine.

```
[production1@servera ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/production1/.ssh/id_rsa): Enter
Created directory '/home/production1/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/production1/.ssh/id_rsa.
Your public key has been saved in /home/production1/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:MCQ8nXCldFSLJV0i5IouUzLzFrbsdz+j08ZIMeST0uQ
production1@servera.lab.example.com
```

The key's randomart image is:

```
+---[RSA 3072]-----+
|  o==B==..  |
|  oB+*..  |
|  o+B.  |
|  =.+*o  |
|  *o*. +S  |
|  o *E .  |
|o . .o.o.  |
|  o  ...+.o  |
|  .o+.o  |
+-----[SHA256]-----+
```

4. Send the public key of the SSH key pair to the `production1` user on the `serverb` machine.

```
[production1@servera ~]$ ssh-copy-id production1@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
production1/.ssh/id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ED25519 key fingerprint is SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KW0TK80.
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
production1@serverb's password: redhat
Number of key(s) added: 1

Now try logging in to the machine, with:  "ssh 'production1@serverb'"
and check to make sure that only the key(s) you wanted were added.
```

5. Verify that the `production1` user can successfully log in to the `serverb` machine with the SSH keys.

```
[production1@servera ~]$ ssh production1@serverb
...output omitted...
[production1@serverb ~]$
```

6. Configure the `sshd` service on `serverb` to prevent users from logging in as the `root` user. Use `redhat` as the `root` password.

- 6.1. Switch to the `root` user on the `serverb` machine.

```
[production1@serverb ~]$ su -
Password: redhat
[root@serverb ~]#
```

- 6.2. Set the `PermitRootLogin` parameter to `no` in the `/etc/ssh/sshd_config` file and reload the `sshd` service. Edit the active uncommented parameter and not a commented example.

```
...output omitted...
PermitRootLogin no
...output omitted...
[root@serverb ~]# systemctl reload sshd.service
```

- 6.3. Open another terminal on the workstation machine and log in to the `servera` machine as the `production1` user. From `servera`, try to log in to the `serverb` machine as the `root` user. This command should fail, because you disabled SSH root user login.

The command exited after three failed attempts to log in to the `servera` machine as the `root` user. By default, the `ssh` command prefers to use SSH keys for authentication, and if it does not find the necessary keys of the user, then it requests the user's password for authentication.



Note

For course convenience, the password-less `root` login is already configured between `workstation` and `servera` in the classroom environment. However, this configuration is highly insecure, and is not recommended for any production environment.

```
[student@workstation ~]$ ssh production1@servera
...output omitted...
[production1@servera ~]$ ssh root@serverb
root@serverb's password: redhat
Permission denied, please try again.
root@serverb's password: redhat
Permission denied, please try again.
root@serverb's password: redhat
root@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[production1@servera ~]$
```

7. Configure the `sshd` service on the `serverb` machine to allow users to authenticate with SSH keys only, rather than with their passwords.
 - 7.1. Return to the first terminal with the `root` active shell on the `serverb` machine. Set the `PasswordAuthentication` parameter to `no` in the `/etc/ssh/sshd_config` file and reload the `sshd` service. Edit the active uncommented parameter and not a commented example.

```
...output omitted...
PasswordAuthentication no
...output omitted...
[root@serverb ~]# systemctl reload sshd
```

- 7.2. Go to the second terminal with the `production1` active shell on the `servera` machine, and try to log in to the `serverb` machine as the `production2` user. This command should fail, because SSH keys are not configured for the `production2` user, and the `sshd` service on the `serverb` machine does not allow the use of passwords for authentication.

```
[production1@servera ~]$ ssh production2@serverb
production2@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

Note

For more granularity, you can use the explicit ssh command `-o PubkeyAuthentication=no` and `-o PasswordAuthentication=yes` options. With these options, you can override the ssh command defaults, and confidently establish whether the preceding command failed based on the settings that you adjusted in the `/etc/ssh/sshd_config` file in the preceding step.

- 7.3. Return to the first terminal with the `root` active shell on the `serverb` machine. Verify that `PubkeyAuthentication` is enabled in the `/etc/ssh/sshd_config` file.

```
[root@serverb ~]$ cat /etc/ssh/sshd_config
...output omitted...
#PubkeyAuthentication yes
...output omitted...
```

The `PubkeyAuthentication` line is commented. Commented lines indicate the default values of a parameter. The public key authentication of SSH is active by default, as the commented line indicates.

- 7.4. Return to the second terminal with the `production1` active shell on the `servera` machine, and try to log in to the `serverb` machine as the `production1` user. This command should succeed, because SSH keys are configured for the `production1` user to log in to the `serverb` machine from the `servera` machine.

```
[production1@servera ~]$ ssh production1@serverb
...output omitted...
[production1@serverb ~]$
```

- 7.5. Exit and close the extra terminal.

```
[production1@serverb ~]$ exit
logout
Connection to serverb closed.
[production1@servera ~]$ exit
logout
[student@workstation ~]$ exit
```

- 7.6. Return to the `workstation` system as the `student` user.

```
[root@serverb ~]# exit
logout
[production1@serverb ~]$ exit
logout
Connection to serverb closed.
[production1@servera ~]$ exit
logout
[student@servera ~]$ exit
```

```
logout  
Connection to servera 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 ssh-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 ssh-review
```

Summary

- With the `ssh` command, users can access remote systems securely with the SSH protocol.
- A client system stores the identities of remote servers in the `~/.ssh/known_hosts` and `/etc/ssh/ssh_known_hosts` files.
- SSH supports both password-based and key-based authentication.
- The `ssh-keygen` command generates an SSH key pair for authentication. The `ssh-copy-id` command exports the public key to remote systems.
- The `sshd` service implements the SSH protocol on Red Hat Enterprise Linux systems.
- Configure advanced SSH settings in the `/etc/ssh/sshd_config` configuration file.
- It is a recommended practice to configure `sshd` to disable remote logins as `root` and to require public key authentication rather than password-based authentication.

Chapter 11

Manage Networking

Goal

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

Objectives

- Describe fundamental concepts of network addressing and routing for a server.
- Test and inspect the current network configuration with command-line utilities.
- Manage network settings and devices with the `nmc li` command.
- Modify network configuration by editing configuration files.
- Configure a server's static hostname and its name resolution and test the results.

Sections

- Describe Networking Concepts (and Quiz)
- 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

Describe Networking Concepts

Objectives

- Describe fundamental concepts of network addressing and routing for a server.

TCP/IP Network Model

The TCP/IP network model is a four-layered set of communication protocols that describes how data communications are packetized, addressed, transmitted, routed, and received between computers over a network.

The protocol is specified by RFC 1122, *Requirements for Internet Hosts - Communication Layers*.

Following are the four layers of the TCP/IP network model:

- Application

Each application has specifications for communication so that clients and servers can communicate across platforms. Common protocols include SSH, HTTPS (secure web), FTP (file sharing), and SMTP (electronic mail delivery).

- Transport

TCP and UDP are transport protocols. TCP is a reliable connection-oriented communication, whereas UDP is a connectionless datagram protocol. Application protocols can use either TCP or UDP ports. A list of well-known and registered ports is in the `/etc/services` file.

When a packet is sent on the network, the combination of the service port and IP address forms a socket. Each packet has a source socket and a destination socket. This information can be used when monitoring and filtering network traffic.

- Internet

The internet layer, or network layer, carries data from the source host to the destination host. The IPv4 and IPv6 protocols are internet layer protocols. Each host has an IP address and a prefix to determine network addresses. Routers are used to connect networks.

- Link

The link layer, or media access layer, provides the connection to physical media. The most common types of networks are wired Ethernet (802.3) and wireless Wi-Fi (802.11). Each physical device has a *Media Access Control (MAC)* address, which is also known as a hardware address, to identify the destination of packets on the local network segment.

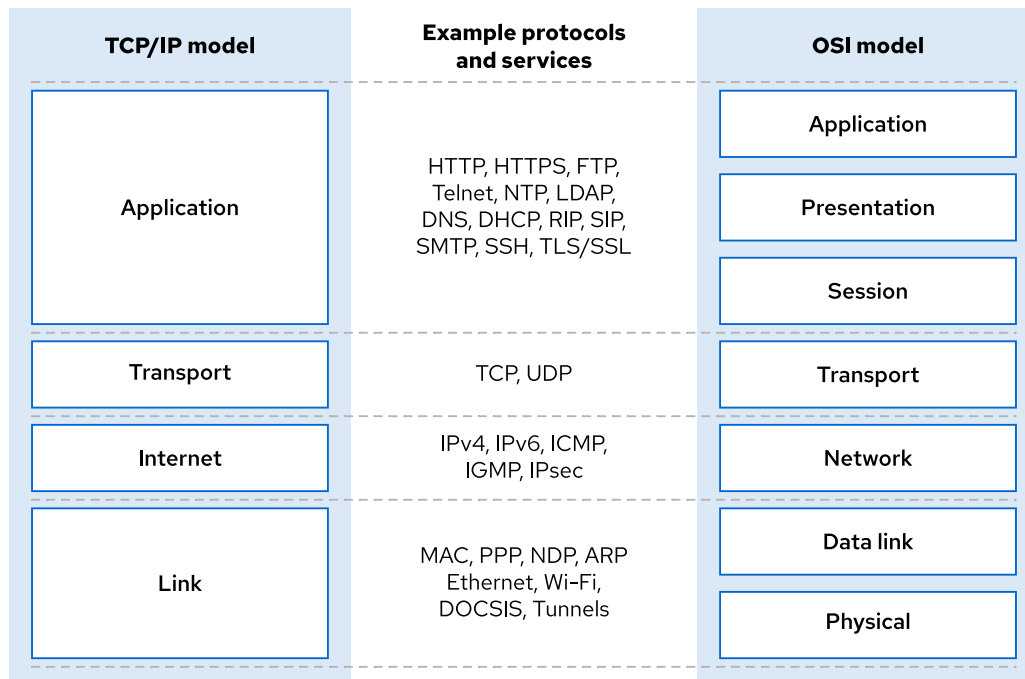


Figure 11.1: Comparison of the TCP/IP and OSI network models

Describe Network Interface Names

Each network port on a system has a name, which you use to configure and identify it.

Earlier versions of Red Hat Enterprise Linux used names such as `eth0`, `eth1`, and `eth2` for each network interface. The `eth0` interface name was the first network port that the operating system detected; `eth1` was the second interface, and so on. However, as devices were added and removed, the mechanism that detected and named devices could change which interface was assigned to which name. Furthermore, the PCIe standard does not guarantee the order in which PCIe devices are detected on boot, which could change device naming unexpectedly due to variations during device or system startup.

In Red Hat Enterprise Linux 7 and later, the default naming system generates names that are consistent across reboots. Instead of being based on the detection order, the assignment of network interface names is based on information from the firmware, the PCI bus topology, and the type of network device. Without consistent device naming, the Linux kernel assigns names to network interfaces used in earlier versions of Red Hat Enterprise Linux by combining a fixed prefix and an index.

Network interface names start with the type of interface:

- Ethernet interfaces begin with `en`
- WLAN interfaces begin with `wl`
- WWAN interfaces begin with `ww`

The rest of the interface name after the type is based on information from the server's firmware, or is determined by the location of the device in the PCI topology.

- `oN` indicates an on-board device with unique index `N` from the server's firmware. The `eno1` name is on-board Ethernet device 1.

- *sN* indicates a device in PCI hotplug slot *N*. For example, `ens3` is an Ethernet card in PCI hotplug slot 3.
- *pMsN* indicates a PCI device on bus *M* in slot *N*. A `wlp4s0` interface is a WLAN card on PCI bus 4 in slot 0. If the card is a multi-function device (such as an Ethernet card with multiple ports, or a device with both Ethernet and another function), then you might see *fN* in the device name. An `enp0s1f0` interface is function 0 of the Ethernet card on bus 0 in slot 1. A second card interface would be named `enp0s1f1`, which is function 1 of that same device.

Persistent naming means that when the name is set for a network interface on the system, the name of the interface does not change, even if you add or remove hardware. A behavior of persistent naming is that a system with a single interface generates a device name by using a hardware information scheme, and is not expected to use the `eth0` kernel naming scheme.

IPv4 Networks

Although IPv4 remains the most common addressing scheme in enterprise networks today, IPv6 has surpassed IPv4 usage in cellular networks. You need a basic understanding of IPv4 networking to manage networking on your servers.

IPv4 Addresses

An *IPv4 address* is a 32-bit number, which is expressed as four 8-bit octets in a decimal format that ranges in value from 0 to 255, separated by single dots. The address is divided into two parts: the network prefix and the host number. The network prefix identifies a unique physical or virtual subnet. The host number identifies a specific host on the subnet. All hosts on the same subnet have the same network prefix and can talk to each other directly without a router. A *network gateway* connects different networks, and a network router commonly operates as the gateway for a subnet.



Note

A *subnet* is a segment of a larger network, and the use of the term depends on the context. An IP network is partitioned into multiple, smaller network segments. Typically, *segment* refers to the physical or virtual link layer, whereas *subnet* refers to the logical, network-layer addressing for the corresponding segment.

Additionally, *subnetting* an assigned large network address subdivides it into multiple, smaller network segments. This IPv4 section introduces network addresses that are implemented as single subnets. The upcoming IPv6 section includes another context, where large networks are *subnetted* into multiple subnets.

In the original IPv4 specification, the allowed network prefixes were one of three fixed sizes for *unicast* packets that have a single source and destination. The network prefix might be 8 bits (*class A*), 16 bits (*class B*), or 24 bits (*class C*). Today, the number of bits in the network prefix is variable, which means that the prefix can be any number in the supported range, and this later specification is called *Classless Inter-Domain Routing (CIDR)*. Although fixed-address classes are no longer in use, many network professionals still refer to networks with 8-bit, 16-bit, or 24-bit network prefixes by using the original class A, B, or C designation.

A *network mask (netmask)* is a binary mask whose length indicates how many bits belong to the network prefix that identifies the subnet. Because an IPv4 address is always 32 bits long, a subnet with a longer network mask has fewer available bits to identify hosts, which means fewer possible hosts. A subnet with a shorter network mask has more available bits to identify hosts, which means more possible hosts and a larger subnet.

Network masks are expressed in one of two forms, which are both used routinely. The first form, which is known as *CIDR notation*, appends a forward slash (/) and an integer up to 32 that indicates the number of bits in the binary mask. The second notation displays the number of bits in the binary mask as four 8-bit octets in decimal format.

IPv4 Subnets and Netmasks

The number of available host addresses in a subnet depends on the network prefix size. For example, a network prefix of /24 leaves 8 bits, or 255 possible host addresses in the subnet. A network prefix of /16 leaves 16 bits, or 65536 possible host addresses in the subnet.

- The *network address* for a subnet is the lowest possible address on a subnet, where the host number is all binary zeros.
- The *broadcast address* for a subnet is the highest possible address on a subnet, where the host number is all binary ones, and is a special address for broadcasting packets to all subnet hosts.
- The *gateway address* for a subnet can be any unique host number in the subnet, but is commonly set to the first available host number, which is a binary number of all zeroes except for a '1' in the last bit. This gateway numbering convention is not mandatory, and subnets that do not need external communication do not set a network gateway.

The following figures illustrate the use of an IP address and a netmask to calculate the network prefix and the host number for a subnet. Perform a *binary AND* calculation where each bit in the IP address and netmask is binary, and compare each bit to its corresponding bit in the IP address and netmask through the prefix length. In an AND calculation, both bits must be a '1' for the result to be a '1', and all other combinations result in '0'. Perform a *binary OR* calculation on the remaining bits in the host number, where either bit can be a '1' for the result to be a '1'. In a binary OR calculation, only two '0' bits result in a '0'.

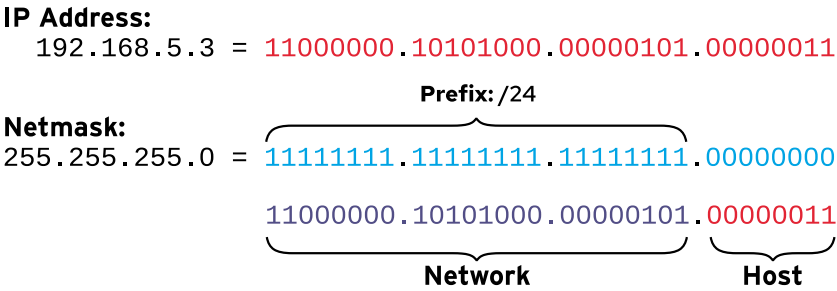


Figure 11.2: IPv4 netmask calculation for a small network

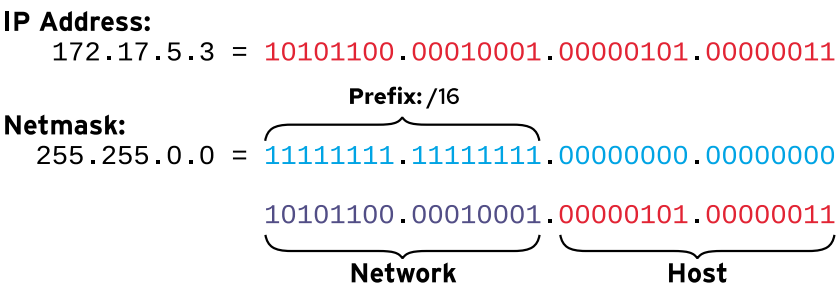


Figure 11.3: IPv4 netmask calculation for a larger network

Example Network Calculations

In the following example, identify the netmask first, and then perform the binary calculations. A netmask of /24 means that the leading 24 bits of the address define the network address (192.168.1.0). In this scenario, 8 bits, or 254 addresses, are available for host addressing.

IPv4 address of 192.168.1.107/24

Network prefix	/24 or 255.255.255.0	11111111.11111111.11111111.00000000
Host address	192.168.1.107	11000000.10101000.00000001.01101011
Network address	192.168.1.0	11000000.10101000.00000001.00000000
Address range for hosts on subnet	192.168.1.1 - 192.168.1.254	11000000.10101000.00000001.00000001 to 11000000.10101000.00000001.11111110
Broadcast address	192.168.1.255	11000000.10101000.00000001.11111111

In the following example, a /19 netmask is a valid network prefix that uses only a partial octet. Variable length netmasks allow subnets with a different quantity of hosts than the full-octet netmasks. The remaining 13 bits, or 8190 addresses, are available for host addressing.

IPv4 address of 172.16.181.23/19

Network prefix	/19 or 255.255.224.0	11111111.11111111.11100000.00000000
Host address	172.16.181.23	10101100.00010000.10110101.00010111
Network address	172.16.160.0	10101100.00010000.10100000.00000000
Address range for hosts on subnet	172.16.160.1 - 172.16.191.254	10101100.00010000.10100000.00000001 to 10101100.00010000.10111111.11111110
Broadcast address	172.16.191.255	10101100.00010000.10111111.11111111

In the following example, the /8 indicates a large network. Only the first octet is used for the network prefix (10.0.0.0). The remaining 24 bits, or 16,777,214 addresses, are available for host addressing. The 10.255.255.255 broadcast address is the last address of the network.

IPv4 address of 10.1.1.18/8

Network prefix	/8 or 255.0.0.0	11111111.00000000.00000000.00000000
Host address	10.1.1.18	00001010.00000001.00000001.00010010
Network address	10.0.0.0	00001010.00000000.00000000.00000000
Address range for hosts on subnet	10.0.0.1 - 10.255.255.254	00001010.00000000.00000000.00000001 to 00001010.11111111.11111111.11111110
Broadcast address	10.255.255.255	00001010.11111111.11111111.11111111

IPv4 Routes

Network packets move from host to host on a subnet and through routers from network to network. Each host has a routing table, which determines which network interface is correct for sending packets to particular networks. A routing table entry lists the destination network, which network interface to use, and the IP address of the router to forward the packet to the final destination. The routing table entry that matches the network prefix of the destination address is used to route the packet. If multiple entries are valid for the destination address, then the entry with the longer prefix is used.

If the destination network does not match a more specific entry, then the packet is routed by using the `0.0.0.0/0` default entry. This default route points to the gateway router on a local subnet that the host can reach.

When a router receives packets that are not addressed to itself, the router forwards the traffic based on its own routing table. Forwarding might send the packet directly to the destination host if this router is on the destination's subnet, or might forward the packet again to another router network. Packet forwarding on routers continues until the packet reaches the requested destination network and host.

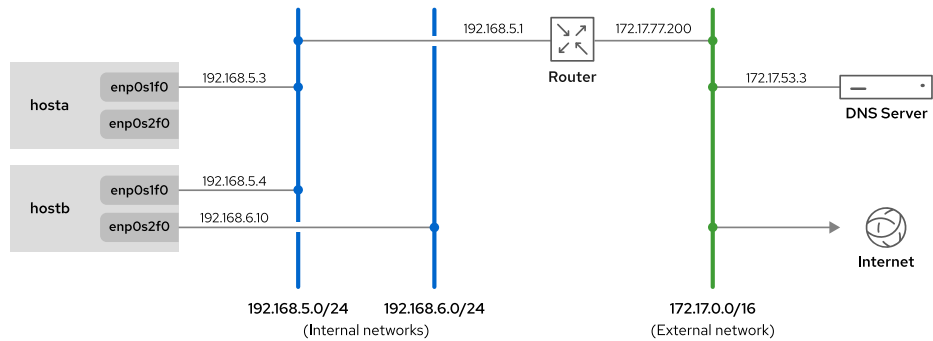


Figure 11.4: Example network topology

Example routing table for the host b machine

Destination	Interface	Router (if needed)
192.168.5.0/24	enp0s1f0	
192.168.6.0/24	enp0s2f0	
172.17.0.0/16	enp0s1f0	192.168.5.1
0.0.0.0/0 (default)	enp0s1f0	192.168.5.1

Consider the preceding network diagram and network routing table.

- Network traffic from the `hostb` machine to any host in the `192.168.6.0/24` network is transmitted directly via the `enp0s2f0` interface.
 - This traffic is because the `hostb` machine has an interface attached to that network, and is the closest match to the route entry.
- Network traffic from the `hostb` machine to a host with the `172.17.50.120` IP address uses the `enp0s1f0` interface, because the traffic matches the third entry in the routing table.

- The `hostb` machine does not have an interface that is directly attached to this network, so this traffic is sent to the next hop router with the address of `192.168.5.1`, which is reachable via the `enp0s1f0` interface. The traffic is then forwarded to its destination.
- Because the `hostb` machine does not have an interface that is directly connected to the `172.17.0.0/16` network, someone with knowledge of the network topology must add this route entry to the routing table.
- Network traffic with a destination that does not match any entry in the routing table is sent to the default route. The default route, which is designated with `0.0.0.0/0`, is shown in the fourth entry.
 - For example, all traffic from the `hostb` machine to the internet is forwarded to the next hop router with the address of `192.168.5.1`, which is reachable via the `enp0s1f0` interface. The traffic is then forwarded to its destination.

IPv4 Address and Route Configuration

A server can automatically configure its IPv4 network settings by communicating with a DHCP server. A local DHCP client queries the subnet by using a link layer protocol to locate a DHCP server or proxy, and negotiates to use a unique address and other settings for a specific lease period. The client must periodically request lease renewal to maintain use of the assigned network configuration.

As an alternative, you can configure a server to use a static network configuration. Static network settings are read from local configuration files. The settings that you use must be appropriate for your subnet. Coordinate with your network administrator to avoid conflicts with other servers in the same subnets.

IPv6 Networks

IPv6 is designed to greatly expand the number of total available device addresses. IPv6 is used in both enterprise networks and for mobile communications. Most if not all *Internet Service Providers* (ISPs) use IPv6 extensively for assigning to internal equipment and for dynamic assignment for customer devices.

IPv6 can also be used in parallel with IPv4 in a dual-stack mode. A network interface can have both IPv6 and IPv4 addresses. Red Hat Enterprise Linux operates in a dual-stack mode by default.

IPv6 Addresses

An IPv6 address is a 128-bit number, which is normally expressed as eight colon-separated groups of four hexadecimal *nibbles* (half-bytes). Each nibble represents four bits of the IPv6 address, so each group represents 16 bits of the IPv6 address.

```
2001:0db8:0000:0010:0000:0000:0000:0001
```

To simplify writing IPv6 addresses, leading zeros in a colon-separated group are not needed. However, at least one hexadecimal digit must be written in each colon-separated group.

```
2001:db8:0:10:0:0:0:1
```

Because addresses with long strings of zeros are common, one or more consecutive groups of zeros only can be combined with *exactly one* block of two colon `::` characters.

```
2001:db8:0:10::1
```

The 2001:db8::0010:0:0:0:1 IPv6 address, though a valid representation, is a less convenient way to write the example address. This different representation can confuse administrators who are new to IPv6. The following list shows tips for writing consistently readable addresses:

- Suppress leading zeros in a group.
- Use a two-colon :: block to shorten the address as much as possible.
- If an address contains two consecutive groups of zeros, which are equal in length, then shorten the leftmost groups of zeros to :: and the rightmost groups to :0: for each group.
- Although it is allowed, do not use :: to shorten a single group of zeros. Use :0: instead, and save :: for consecutive groups of zeros.
- Always use lowercase letters for a through f hexadecimal characters.



Important

When including a TCP or UDP network port after an IPv6 address, always enclose the IPv6 address in square brackets so that the port does not appear to be part of the address.

```
[2001:db8:0:10::1]:80
```

IPv6 Subnets

A normal IPv6 unicast address is divided into two parts: the *network prefix* and *interface ID*. The network prefix identifies the subnet. Two network interfaces on the same subnet cannot have the same interface ID; the interface ID identifies a particular interface on the subnet.

Unlike IPv4, IPv6 has a standard subnet mask, /64, which is used for almost all normal addresses. In this case, half of the 128 bit address is the network prefix and the other half is the interface ID. With 64 bits for host addresses, a single subnet could theoretically contain 2⁶⁴ hosts.

Typically, the network provider allocates a shorter prefix to an organization, such as a /48. This prefix leaves the rest of the network part for assigning subnets (up to the maximum /64 length) from that allocated prefix. For example, when a /48 allocation prefix is assigned, 16 bits are available for up to 65536 subnets.

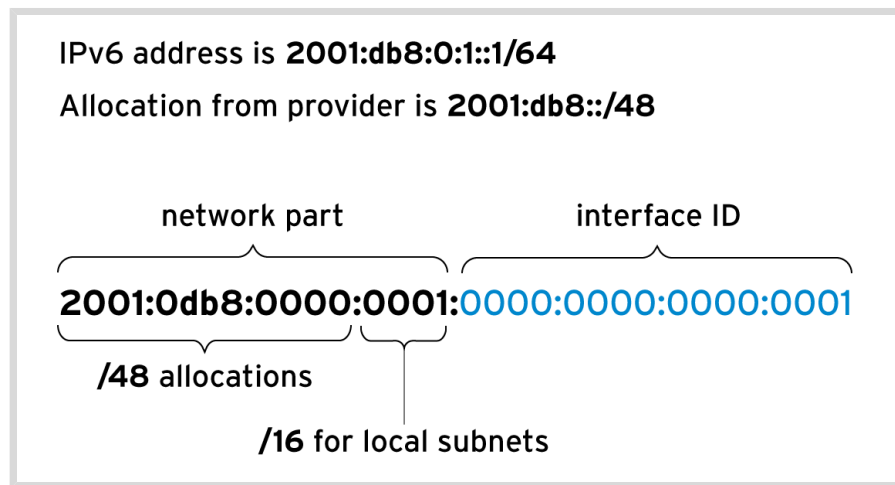


Figure 11.5: IPv6 address parts and subnetting

Common IPv6 Addresses and Networks

IPv6 address or network	Purpose	Description
::1/128	localhost	The IPv6 equivalent to the 127.0.0.1/8 address, which is set on the loopback interface.
::	The unspecified address	The IPv6 equivalent to 0.0.0.0. For a network service, it might indicate that it is listening on all configured IP addresses.
::/0	The default route (the IPv6 internet)	The IPv6 equivalent to the 0.0.0.0/0 address. The default route in the routing table matches this network; the router for this network is where all traffic is sent in the absence of a better route.
2000::/3	Global unicast addresses	The <i>Internet Assigned Numbers Authority (IANA)</i> currently allocates "normal" IPv6 addresses from this space. The addresses include all the networks that range from 2000::/16 through 3fff::/16.
fd00::/8	Unique local addresses (RFC 4193)	IPv6 has no direct equivalent of the RFC 1918 private address space, although this network range is close. A site can use these networks to self-allocate a private routable IP address space inside the organization. However, these networks cannot be used on the global internet. The site must randomly select a /48 from this space, but it can subnet the allocation into /64 networks normally.

IPv6 address or network	Purpose	Description
fe80::/10	Link-local addresses	Every IPv6 interface automatically configures a link-local unicast address that works only on the local link on the fe80::/64 network. However, the entire fe80::/10 range is reserved for future use by the local link. This topic is discussed in more detail later.
ff00::/8	Multicast	The IPv6 equivalent to the 224.0.0.0/4 address. Multicast is used to transmit to multiple hosts at the same time, and is particularly important in IPv6 because it has no broadcast addresses.



Important

The previous table lists network address *allocations* that are reserved for specific purposes. These allocations might consist of many networks. IPv6 networks are allocated from the global unicast and link-local unicast address spaces that have a standard /64 network mask.

A link-local address in IPv6 is an unroutable address that the system uses only to talk to other systems on the same network link. To ensure that the IP address is unique, the system uses a specific method to compute the interface ID of the link-local address.



Note

Originally, the interface ID for the IPv6 link-local address was constructed from the MAC address of the network device. Exposing the MAC address as part of the IPv6 address might cause some security and privacy issues, because it becomes possible to identify and follow a computer on the network.

By default in Red Hat Enterprise Linux 9, NetworkManager generates a random but stable interface ID for the interface, according to the algorithm in RFC 7217. This algorithm is controlled by the `ipv6.addr-gen-mode` connection setting, which defaults to `stable-privacy`.

IPv6 Privacy Extensions (RFC 4941) are a different solution to the same concern and are controlled by different settings, which are disabled by default.

Use the `ip addr show` command to retrieve the link-local IPv6 address, as in the following example. Add the `-br` option for a brief listing of addresses only.

```
[user@host ~]$ ip addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 52:54:00:01:fa:0a brd ff:ff:ff:ff:ff:ff
    inet 10.42.0.1/16 brd 10.42.255.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::7418:cf98:c742:3681/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Add the `-br` option for a brief listing of IPv4 and IPv6 addresses only.

```
[user@host ~]$ ip -br addr show dev eth0
eth0      UP      10.42.0.1/16 fe80::7418:cf98:c742:3681/64
```

To operate correctly, IPv6 relies on the link-local address. The interface always keeps that address, even when you assign a routable IPv6 address manually or with an automated method.

With multicast, one system can send traffic to a special IP address that multiple systems receive. Multicast differs from broadcast, because broadcast packets are not routable and reach only local subnet hosts. Conversely, multicast packets are routed to specific hosts that announced a request for the uniquely addressed multicast packets. Multicast packets can be routed to other subnets, if all intermediary routers support forwarding multicast requests and routing.

Multicast plays a larger role in IPv6 than in IPv4, because IPv6 has no broadcast address. The `ff02::1` IPv6 address is a key multicast address that is used as the `all-nodes` link-local address, and behaves like a broadcast address. You can ping this address to send traffic to all nodes on the link. Link-scope multicast addresses (which start with `ff02::/8`) must be specified with a scope identifier, as for a link-local address.

```
[user@host ~]$ ping6 ff02::1%ens3
PING ff02::1%ens3(ff02::1) 56 data bytes
64 bytes from fe80::211:22ff:feaa:bbcc: icmp_seq=1 ttl=64 time=0.072 ms
64 bytes from fe80::200:aaff:fe33:2211: icmp_seq=1 ttl=64 time=102 ms (DUP!)
64 bytes from fe80::bcd:efff:fea1:b2c3: icmp_seq=1 ttl=64 time=103 ms (DUP!)
64 bytes from fe80::211:22ff:feaa:bbcc: icmp_seq=2 ttl=64 time=0.079 ms
...output omitted...
```

IPv6 Address Configuration

IPv4 has two ways to configure addresses on network interfaces. The administrator can manually configure network addresses on interfaces, or addresses can be configured dynamically with DHCP. IPv6 supports manual configuration, and two methods of dynamic configuration, one of which is DHCPv6.

You can select interface IDs for static IPv6 addresses, similar to IPv4. In IPv4, two addresses on a network cannot be used: the lowest address, which is the network address; and the highest address, which is the broadcast address. In IPv6, two interface IDs are reserved, and cannot be used as normal host interface addresses:

- The all-zeros identifier `0000:0000:0000:0000` ("subnet router anycast") that all routers on the link use. For example, on the `2001:db8::/64` network, the anycast address is `2001:db8::`.
- The identifiers `fdff:ffff:ffff:ff80` through `fdff:ffff:ffff:ffff`.

DHCPv6 lease negotiations work differently from IPv4 DHCP, because DHCPv6 has no broadcast address. A host sends a DHCPv6 request from a link-local address to port 547/UDP on the dedicated `ff02::1:2 all-dhcp-servers` link-local multicast group. A listening DHCPv6 server can choose to reply with appropriate information to port 546/UDP on the client's provided link-local address.

The `dhcp` package in Red Hat Enterprise Linux 9 provides support for a DHCPv6 server.

In addition to DHCPv6, IPv6 also supports another dynamic configuration method, which is called *Stateless Address Autoconfiguration (SLAAC)*. To use SLAAC, a host configures its interface with a link-local `fe80::/64` address, and sends a "router solicitation" to the `ff02::2 all-routers` link-local multicast group. An IPv6 router on the local link responds to the host's link-local address with the subnet's previously configured network prefix and other relevant information. The host uses the provided network prefix with an interface ID that is constructed the same as for link-local addresses. The router periodically sends multicast updates (*router advertisements*) to confirm or to update the provided network information.

With the `radvd` package in Red Hat Enterprise Linux 9, a Red Hat Enterprise Linux based IPv6 router can provide SLAAC through router advertisements.



Important

A typical Red Hat Enterprise Linux 9 system that is configured for dynamic IPv4 addresses with DHCP is typically configured for dynamic IPv6 by using SLAAC. Hosts with a dynamic IPv6 configuration might unexpectedly obtain additional IPv6 addresses when a new IPv6 router is added to the network.

Some IPv6 deployments combine SLAAC and DHCPv6, and use SLAAC to provide the network address information, where DHCPv6 provides more network options, such as DNS servers and search domains.

Hostnames and IP Addresses

IP addresses are not human-friendly in daily use. Users generally prefer to work with hostnames rather than with number strings. Linux has *name resolution* mechanisms to map a hostname to an IP address.

One method is to create static entries for each hostname in each system's `/etc/hosts` file. With this method, you must manually update each server's copy of the `hosts` file.

When configured, you can look up the address for a hostname (or a hostname from an address) by using the *Domain Name System (DNS)* network service. DNS is a distributed network of servers that provides name resolution mappings. For name resolution to work, a host must be configured to know where to contact a *nameserver*. The nameserver does not need to be on the same subnet, but the host must be able to reach it. A nameserver configuration is typically obtained through DHCP or by creating static address settings in the `/etc/resolv.conf` file. Later sections of this chapter discuss how to configure name resolution.

**References**

services(5), ping(8), biosdevname(1), and udev(7) 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/

Understanding systemd's Predictable Network Device Names

<https://major.io/2015/08/21/understanding-systemds-predictable-network-device-names/>

Selected IETF RFC references:

RFC2460: Internet Protocol, Version 6 (IPv6) Specification

<https://datatracker.ietf.org/doc/html/rfc2460>

RFC4291 IP Version 6 Addressing Architecture

<https://datatracker.ietf.org/doc/html/rfc4291>

RFC5952: A Recommendation for IPv6 Address Text Representation

<https://datatracker.ietf.org/doc/html/rfc5952>

RFC4862: IPv6 Stateless Address Autoconfiguration

<https://datatracker.ietf.org/doc/html/rfc4862>

RFC3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

<https://datatracker.ietf.org/doc/html/rfc3315>

RFC3736: Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6

<https://datatracker.ietf.org/doc/html/rfc3736>

RFC4193: Unique Local IPv6 Unicast Addresses

<https://datatracker.ietf.org/doc/html/rfc4193>

RFC7217: A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)

<https://datatracker.ietf.org/doc/html/rfc7217>

RFC8415: Dynamic Host Configuration Protocol for IPv6 (DHCPv6)

<https://datatracker.ietf.org/doc/html/rfc8415>

► Quiz

Describe Networking Concepts

Choose the correct answer to the following questions:

- 1. What is the size, in bits, of an IPv4 address?
 - a. 4
 - b. 8
 - c. 16
 - d. 32
 - e. 64
 - f. 128

- 2. Which term determines how many leading bits in the IP address contribute to its network address?
 - a. netscope
 - b. netmask
 - c. subnet
 - d. multicast
 - e. netaddr
 - f. network

- 3. Which address represents a valid IPv4 *host* address on the 192.168.1.0/24 network?
 - a. 192.168.1.188
 - b. 192.168.1.0
 - c. 192.168.1.255
 - d. 192.168.1.256

- 4. Which number is the size, in bits, of an IPv6 address?
 - a. 4
 - b. 8
 - c. 16
 - d. 32
 - e. 64
 - f. 128

- 5. Which address does not represent a valid IPv6 address?
- a. 2000:0000:0000:0000:0000:0000:0000:0001
 - b. 2::1
 - c. ::
 - d. ff02::1:0:0
 - e. 2001:3::7:0:2
 - f. 2001:db8::7::2
 - g. 2000::1
- 6. Which term refers to the ability of one system to send traffic to a special IP address that multiple systems receive?
- a. netscope
 - b. netmask
 - c. subnet
 - d. multicast
 - e. netaddr
 - f. network

► Solution

Describe Networking Concepts

Choose the correct answer to the following questions:

- 1. What is the size, in bits, of an IPv4 address?
 - a. 4
 - b. 8
 - c. 16
 - d. 32
 - e. 64
 - f. 128

- 2. Which term determines how many leading bits in the IP address contribute to its network address?
 - a. netscope
 - b. netmask
 - c. subnet
 - d. multicast
 - e. netaddr
 - f. network

- 3. Which address represents a valid IPv4 *host* address on the 192.168.1.0/24 network?
 - a. 192.168.1.188
 - b. 192.168.1.0
 - c. 192.168.1.255
 - d. 192.168.1.256

- 4. Which number is the size, in bits, of an IPv6 address?
 - a. 4
 - b. 8
 - c. 16
 - d. 32
 - e. 64
 - f. 128

- 5. Which address does not represent a valid IPv6 address?
- a. 2000:0000:0000:0000:0000:0000:0000:0001
 - b. 2::1
 - c. ::
 - d. ff02::1:0:0
 - e. 2001:3::7:0:2
 - f. 2001:db8::7::2
 - g. 2000::1
- 6. Which term refers to the ability of one system to send traffic to a special IP address that multiple systems receive?
- a. netscope
 - b. netmask
 - c. subnet
 - d. multicast
 - e. netaddr
 - f. network

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. Often, 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 1000
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.2/24 brd 192.0.2.255 scope global ens3
        valid_lft forever preferred_lft forever
    inet6 2001:db8:0:1:5054:ff:fe00:b/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe00:b/64 scope link
        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.

- ❸ The `inet` string shows an IPv4 address, its network prefix length, and scope.
- ❹ The `inet6` string shows an IPv6 address, its network prefix length, and scope. This address is of *global* scope and is normally used.
- ❺ 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 ~]$
```

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(ff02::1) 56 data bytes
64 bytes from fe80::78cf:7fff: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:7fff: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 the `tracpath` 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 ~]$ tracpath 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 `tracpath` 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 `tracpath6` and `traceroute -6` commands are the equivalent IPv6 commands to the `tracpath` and `traceroute` commands.

```
[user@host ~]$ tracepath6 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, whereas 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 earlier `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)`, `tracpath(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

► Guided Exercise

Validate Network Configuration

Inspect the network configuration of one of your servers.

Outcomes

- Identify the current network interfaces and 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 **52:54:00:00:fa:0a** Ethernet address. 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 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 **52:54:00:00:fa:0a** Ethernet address. 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 the `classroom.example.com` system.

```
[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
```

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* has 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
eno2    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 abbreviate `nmcli connection modify` as `nmcli con mod`. The abbreviation can be as short as a single letter, provided that it uses 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 data for the added network connection is stored in the `/etc/NetworkManager/system-connections/` directory as a file with a `.nmconnection` suffix.

The following example adds an `eno2` connection of the `ethernet` type for the `eno2` network interface:

```
[root@host ~]# nmcli con add con-name eno2 \
type ethernet ifname eno2
Connection 'eno2' (8159b66b-3c36-402f-aa4c-2ea933c7a5ce) successfully added
```

The next example creates an `eno3` connection of the `ethernet` type for the `eno3` network interface 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 ifname eno3 \
ipv4.method manual 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 \
  ipv6.method manual ipv4.addresses 192.0.2.7/24 ipv4.gateway 192.0.2.1 \
  ipv4.method manual
```

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 setting types. 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/`. Whereas `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 lists 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

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

Use the `nmc li` 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 `52:54:00:00:fa:0a` Ethernet address. 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
...output omitted...
```

► 3. Use the `nmcli` command to view network settings.

- 3.1. 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         5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
...output omitted...
```

- 3.2. 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:                     172.25.250.220
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...
GENERAL.NAME:                 System eth0
GENERAL.UUID:                 5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
GENERAL.DEVICES:              eth0
GENERAL.IP-IFACE:             eth0
GENERAL.STATE:                activated
GENERAL.DEFAULT:              yes
...output omitted...
```

3.3. Show the device status.

```
[root@servera ~]# nmcli dev status
```

DEVICE	TYPE	STATE	CONNECTION
eth0	ethernet	connected	System eth0
eth1	ethernet	connected	System eth1
lo	loopback	connected (externally)	lo

3.4. 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/2
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]:	172.25.250.220
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.dns 172.25.250.220 \
ipv4.addresses 172.25.250.10/24 ipv4.gateway 172.25.250.254
Connection 'static-addr' (3ef7c2c6-8c72-4cb6-9b19-2878e50285b1) successfully
added.
```

- ▶ 5. Display and activate the new connection.

5.1. View 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	eth1
lo	d22fdd46-f935-4212-8908-7cdd37b9f723	loopback	lo
static-addr	3ef7c2c6-8c72-4cb6-9b19-2878e50285b1	ethernet	--

5.2. View the active connections.

```
[root@servera ~]# nmcli con show --active
```

NAME	UUID	TYPE	DEVICE
System eth0	5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03	ethernet	eth0
System eth1	9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04	ethernet	eth1
lo	d22fdd46-f935-4212-8908-7cdd37b9f723	loopback	lo

5.3. 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)
```

5.4. Verify the new active connection.

```
[root@servera ~]# nmcli con show --active
```

NAME	UUID	TYPE	DEVICE
static-addr	3ef7c2c6-8c72-4cb6-9b19-2878e50285b1	ethernet	eth0
System eth1	9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04	ethernet	eth1
lo	d22fdd46-f935-4212-8908-7cdd37b9f723	loopback	lo

- 6. Update the previous connection so that it does not start at boot. Verify that the `static-addr` connection is used when the system reboots.

6.1. Disable the original connection so that it does not start automatically at boot.

```
[root@servera ~]# nmcli con mod "System eth0" \
connection.autoconnect no
```

6.2. Reboot the system.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

6.3. Log in to the `servera` machine and verify that the `static-addr` connection is the active connection.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ nmcli con show --active
```

NAME	UUID	TYPE	DEVICE
static-addr	3ef7c2c6-8c72-4cb6-9b19-2878e50285b1	ethernet	eth0
System eth1	9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04	ethernet	eth1
lo	6e328ad2-7ea6-411a-a0e8-b7658c71b4d7	loopback	lo

► 7. Test connectivity by using the new network addresses.

7.1. Verify the IP address.

```
[student@servera ~]$ ip -br addr show eth0
eth0          UP          172.25.250.10/24 fe80::1be9:a14f:c8d9:8af8/64
```

7.2. Verify the default gateway.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev eth0 proto static metric 100
10.80.0.0/16 dev eth1 proto kernel scope link src 10.80.3.1 metric 101
169.254.169.254 via 10.80.0.10 dev eth1 proto dhcp src 10.80.3.1 metric 101
172.25.250.0/24 dev eth0 proto kernel scope link src 172.25.250.10 metric 100
```

7.3. Ping the DNS address.

```
[student@servera ~]$ ping -c3 172.25.250.220
PING 172.25.250.220 (172.25.250.220) 56(84) bytes of data.
64 bytes from 172.25.250.220: icmp_seq=1 ttl=64 time=0.999 ms
64 bytes from 172.25.250.220: icmp_seq=2 ttl=64 time=0.250 ms
64 bytes from 172.25.250.220: icmp_seq=3 ttl=64 time=0.293 ms

--- 172.25.250.220 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.250/0.514/0.999/0.343 ms
```

7.4. 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
```

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 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

nmcli con mod	*.nmconnection file	Effect
ipv4.method manual	[ipv4] method=manual	Configure IPv4 addresses statically.
ipv4.method auto	[ipv4] method=auto	Look for configuration settings from a DHCPv4 server. It shows static addresses only when it has information from DHCPv4.
ipv4.addresses 192.0.2.1/24	[ipv4] address1=192.0.2.1/24	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.
ipv4.gateway 192.0.2.254	[ipv4] gateway=192.0.2.254	Set the default gateway.

nmcli con mod	*.nmconnection file	Effect
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 by using SLAAC.
ipv6.addresses 2001:db8::a/64	[ipv6] address1=2001:db8::a/64	Set a 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.
connection.autoconnect yes	[connection] autoconnect=true	Automatically activate this connection at boot.

nmcli con mod	*.nmconnection file	Effect
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/NetworkManager/system-connections/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=manual

[802-3-ethernet]
mac-address=00:23:5a:47:1f:71
```


IPv4 Configuration Options for Key File Format

Static	Dynamic	Either
[ipv4] address1=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=ethernet

After you modify the configuration files, run the `nmcli con reload` command so that NetworkManager loads the configuration changes. If the `autoconnect` variable is set to `false` in the connection profile, then activate the connection manually.

```
[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

► Guided Exercise

Edit Network Configuration Files

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 `en_X_` placeholder in subsequent commands. The active connection is called `System eth0`, and the configuration is in the `/etc/NetworkManager/system-connections/"System eth0.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
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: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8942 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
link/ether fa:16:3e:15:d4:0d brd ff:ff:ff:ff:ff:ff
altname enp0s4
altname ens4
[student@servera ~]$ nmcli con show --active
NAME                UUID                                  TYPE      DEVICE
System eth0         5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
System eth1         9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  ethernet  eth1
lo                  0e77d074-dd20-465e-87f5-e3974e6f42ab  loopback   lo
[student@servera ~]$ ls /etc/NetworkManager/system-connections/
'System eth0.nmconnection'
```

- ▶ 3. On the servera machine, switch to the root user, and then edit the `/etc/NetworkManager/system-connections/"System eth0.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 `/etc/NetworkManager/system-connections/"System eth0.nmconnection"` configuration file. Add the `10.0.1.1/24` address as the second address below the first address in the file.

```

...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 "System eth0"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/4)
```

- ▶ 5. Verify that the new IP address is assigned successfully.

```
[root@servera ~]# ip -br addr show eth0
eth0          UP          172.25.250.10/24 10.0.1.1/24
fe80::5054:ff:fe00:fa0a/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/"System eth0.nmconnection"` file to add an address of `10.0.1.2/24` and load the new configuration.

- 7.1. 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 ~]#
```

- 7.2. Edit the `/etc/NetworkManager/system-connections/"System eth0.nmconnection"` configuration file. Add the `10.0.1.2/24` address as the second address below the first address in the file.

```
...output omitted...
[ipv4]
address1=172.25.250.11/24,172.25.250.254
address2=10.0.1.2/24
...output omitted...
```

- 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 "System eth0"
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/4)
```

- 7.5. Verify that the new IP address is assigned successfully.

```
[root@serverb ~]# ip -br addr show eth0
eth0          UP          172.25.250.11/24 10.0.1.2/24
fe80::5054:ff:fe00:fa0b/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=1.23 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.372 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.440 ms

--- 10.0.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.372/0.679/1.225/0.387 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=0.726 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.349 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=0.342 ms

--- 10.0.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2087ms
rtt min/avg/max/mdev = 0.342/0.472/0.726/0.179 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
```

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 hostname host.example.com
[root@host ~]# hostnamectl status
  Static hostname: host.example.com
        Icon name: computer-vm
        Chassis: vm #
  Machine ID: ace63d6701c2489ab9c0960c0f1afe1d
    Boot ID: 0edf5ba1830c48adbd6babfa08f0b867
  Virtualization: kvm
  Operating System: Red Hat Enterprise Linux 9.0 (Plow)
    CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
      Kernel: Linux 5.14.0-70.13.1.el9_0.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 based on the configuration in the `/etc/nsswitch.conf` file. Under the default configuration, the command starts by checking 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 DNS 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/NetworkManager/system-connections/ID
...output omitted...
[ipv4]
...output omitted...
dns=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
```

In the following example, 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 listed at least one IPv4-reachable and an IPv6 name server (assuming a dual-stack system), in the event of networking issues with either stack.

Test DNS Name Resolution

The host `HOSTNAME` command can test DNS server connectivity.

```
[root@host ~]# host servera.lab.example.com
servera.lab.example.com has address 172.25.250.10
[root@host ~]# host 172.25.250.10
10.250.25.172.in-addr.arpa domain name pointer servera.lab.example.com.
```

**Important**

DHCP automatically rewrites the `/etc/resolv.conf` file when interfaces are started, unless you specify `ignore-auto-dns = yes` in the relevant interface configuration file in the `/etc/NetworkManager/system-connections/` directory.

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 servera.lab.example.com

; <<>> DiG 9.16.23-RH <<>> servera.lab.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26621
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 5df1b934c1dafb17384d979466685ea98912fd5cbfd21352 (good)
;; QUESTION SECTION:
;servera.lab.example.com. IN A

;; ANSWER SECTION:
servera.lab.example.com. 86400 IN A 172.25.250.10
...output omitted...
```

Neither the `host` nor the `dig` commands view the configuration in the `/etc/hosts` file. To test the `/etc/hosts` file, use the `getent hosts HOSTNAME` command.


```
[root@host ~]# getent hosts servera.lab.example.com
172.25.250.10 servera.lab.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

► Guided Exercise

Configure Hostnames and Name Resolution

Manually configure the system's static hostname, the `/etc/hosts` file, and the 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@testa ~]$ sudo -i
[sudo] password for student: student
[root@testa ~]#
```

- 2. View the current hostname settings.

- 2.1. Display the current hostname.

```
[root@testa ~]# hostname
testa
```

- 2.2. Display the hostname status. The persistent, locally configured hostname is displayed in the `Static hostname` field. The current runtime hostname, which is obtained from DHCP or DNS network services, is displayed in the `Transient hostname` field.

```
[root@testa ~]# hostnamectl status
Static hostname: servera.lab.example.com
Transient hostname: testa
Icon name: computer-vm
```

```

Chassis: vm #
Machine ID: ace63d6701c2489ab9c0960c0f1afe1d
Boot ID: 03bf1d5518bd43b4a25cfe9a18d5a46a
Virtualization: kvm
Operating System: Red Hat Enterprise Linux 9.0 (Plow)
CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
Kernel: Linux 5.14.0-70.13.1.el9_0.x86_64
Architecture: x86-64
Hardware Vendor: Red Hat
Hardware Model: OpenStack Compute

```

- 3. Set a static hostname to match the current static hostname.

- 3.1. Change the hostname and the hostname configuration file.

```

[root@testa ~]# hostnamectl hostname \
servera.lab.example.com

```

- 3.2. View the content of the `/etc/hostname` file, which provides the hostname at network start.

```

[root@testa ~]# cat /etc/hostname
servera.lab.example.com

```

- 3.3. Log out and log in to `servera` as the `student` user. Switch to the `root` user to change the command prompt to show the updated hostname.

```

[root@testa ~]# exit
logout
[student@testa ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#

```

- 3.4. 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.

```
[root@servera ~]# cat /etc/hostname
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 the `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 the `classroom.example.com` server.

```
[root@servera ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
```

- 5.2. Update the `/etc/hosts` file to add the `class` server to access the `172.25.254.254` IP address. 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 the `class` server.

```
[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
```

► Lab

Manage Networking

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

1. Log in to the `serverb` machine as the `student` user. Switch to the `root` user.
2. Create a connection with a static network configuration by using the settings in the table.

Parameter	Setting
Connection name	lab
Interface name	ethX (might vary; use the interface with the 52:54:00:00:fa:0b MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

3. Configure the new connection to start automatically. Other connections should not start automatically.
4. Modify the new connection to use also the 10.0.1.1/24 IP address.
5. Configure the `hosts` file so that you can reference the 10.0.1.1 IP address with the `private` name.
6. Reboot the system.
7. 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
```

► Solution

Manage Networking

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

1. Log in to the `serverb` machine as the `student` user. Switch to the `root` user.
 - 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 ~]#
```

2. Create a connection with a static network configuration by using the settings in the table.

Parameter	Setting
Connection name	lab
Interface name	ethX (might vary; use the interface with the 52:54:00:00:fa:0b MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

- 2.1. Determine the interface name and the current active connection's name.

**Important**

The name of your connection might vary according to the course platform and hardware in use. The solution assumes that the interface name is `eth0` and that the connection name is `System eth0`. Locate the current connection name (such as `System eth0` or `Wired connection 1`) that is associated with the interface in the table. Use the located connection name to replace `System eth0` throughout this exercise if different.

```
[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
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8942 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
    link/ether fa:16:3e:a1:e8:d9 brd ff:ff:ff:ff:ff:ff
    altname enp0s4
    altname ens4

[root@serverb ~]# nmcli con show --active
```

NAME	UUID	TYPE	DEVICE
System eth0	5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03	ethernet	eth0
System eth1	9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04	ethernet	eth1
lo	f3266717-b049-4b7a-a37b-0b34671a1d77	loopback	lo

- 2.2. Create the `lab` connection profile based on the table information in the instructions. Associate the profile with your network interface, as listed in the output of the previous `ip link` command.

```
[root@serverb ~]# nmcli con add con-name lab ifname eth0 type ethernet \
ip4.method manual ip4.dns '172.25.250.254' \
ip4.addresses '172.25.250.11/24' ip4.gateway '172.25.250.254'
Connection 'lab' (8321cc19-5bdf-46fb-8046-10cc4203732c) successfully added.
```

3. 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
```

4. Modify the new connection to use also the `10.0.1.1/24` IP address.

```
[root@serverb ~]# nmcli con mod "lab" +ip4.addresses '10.0.1.1/24'
[root@serverb ~]# nmcli con reload
```

5. 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
```

6. 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=0.635 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
time=0.432 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64
time=0.428 ms

--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2048ms
rtt min/avg/max/mdev = 0.428/0.498/0.635/0.096 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
```

Summary

- The TCP/IP network model is a simplified, four-layered set of abstractions that describes how different protocols interoperate for computers to send traffic from one machine to another over the internet.
- IPv4 is the primary network protocol on the internet today.
- IPv6 is intended as an eventual replacement for the IPv4 network protocol.
- By default, Red Hat Enterprise Linux operates in dual-stack mode, and uses both network protocols in parallel.
- Network routes determine the correct network interface to send packets to a particular network.
- The `NetworkManager` daemon monitors and manages network configuration.
- The `nmc li` command-line tool configures 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 12

Install and Update Software Packages

Goal

Download, install, update, and manage software packages from Red Hat and DNF package repositories.

Objectives

- Register a system by using your Red Hat account and assign it entitlements for software updates and support services provided by Red Hat.
- Explain how software is provided as RPM packages, and investigate the DNF and RPM installed system packages.
- 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)
- Explain and Investigate RPM Software Packages (and Guided Exercise)
- 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 by using your Red Hat account and assign it entitlements for software updates and support services provided by Red Hat.

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 it is 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. A repository is a central location for storing and maintaining software packages.
- *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 also includes the subscription commands, as needed, if you have not yet enabled SCA.

Activation Keys

An *activation key* is a preconfigured subscription management file that is available for use with both Red Hat Satellite Server and subscription management through the Red Hat Customer Portal. Use the command-line tools 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.

Subscribe a System to Red Hat

Red Hat is migrating from `access.redhat.com` to the Hybrid Cloud Console `console.redhat.com` by the end of 2024. By the end of 2024, Red Hat Subscription Management users who haven't switched to using Simple Content Access will be converted to use this experience. This migration unifies system and subscription management experiences, regardless of deployment or registration method. The `console.redhat.com` provides features such as richer permissions, simplification of client registration, and better alignment with cloud-native management.

Additionally, Red Hat is transitioning from entitlement based (system-level) subscription management to account level subscription management (Simple Content Access).

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.

Subscribe a System with the GNOME Application

To register a system by using a GNOME application, 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 12.1: 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 RHEL Web Console

To register a system with the web console, you must log in as a privileged user. Click **Subscriptions** and then click **Register**:

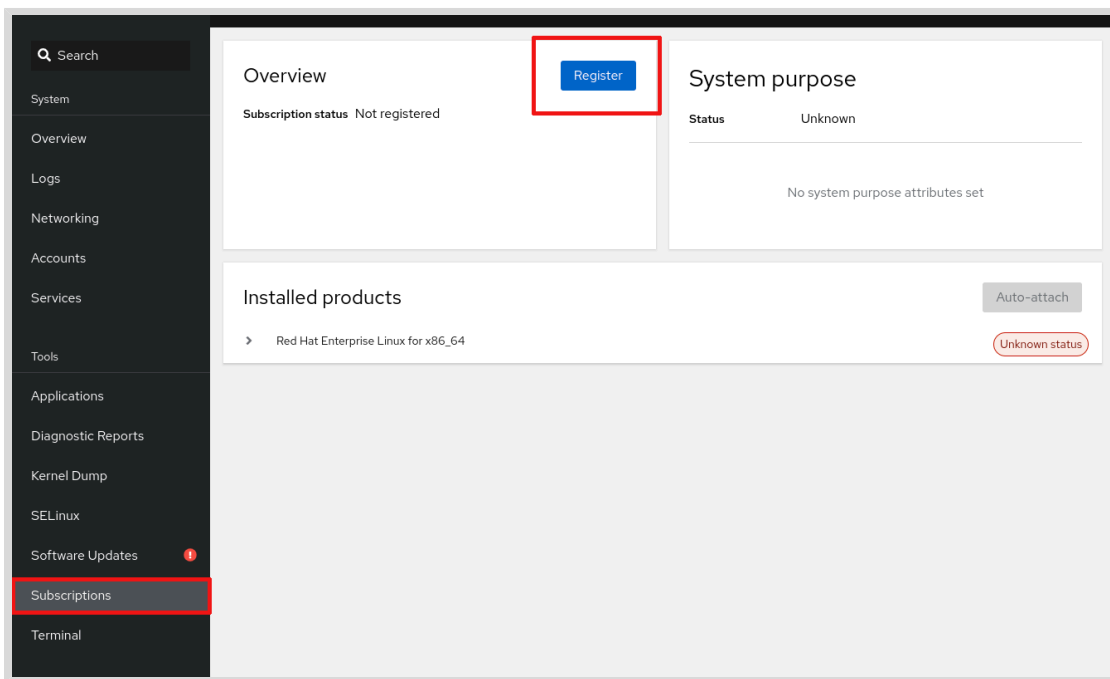


Figure 12.2: Web console subscriptions

The **Register System** dialog box in the web console is similar to the GNOME application. Provide the login and the password for your Red Hat Customer Portal account and click **Register** to register the system. You can then install new software or update your system, according to its attached subscription.

Subscribe a System with the Command-line Tool

There are two command-line tools available to register your system.

Use the `subscription-manager` command to register RHEL 8.7 or earlier systems, or if you are registering systems with a Red Hat Satellite Server.

The `rhc` command registers RHEL 8.8 and later systems to the Red Hat Customer Portal, and can also enable Red Hat Insights service with a single command.

Subscribe a System by Using the `rhc` Command

The `rhc` command replaces the earlier `subscription-manager` tool and can be used on recent RHEL systems to register them to Red Hat Customer Portal and to enable Red Hat Insights service with a single command.

Use the `rhc connect` command along with an activation key and your numeric organization identifier (organization ID) to register a system and enable Red Hat Insights service. You can use a pre-configured activation key to apply selected system-level features, such as attributes that specify the system purpose, during the registration process.

```
[root@host ~]# rhc connect -a host_key -o 117018
Connecting host.lab.example.com to Red Hat.
This might take a few seconds.
```

- Connected to Red Hat Subscription Management
- Connected to Red Hat Insights

- Activated the Red Hat connector daemon

Manage your Red Hat connector systems: <https://red.ht/connector>

You can also use username and password to register your system:

```
[root@host ~]# rhc connect -u <_username_> -p <_password_>
```

The following example shows to unregister a system:

```
[root@host ~]# rhc disconnect
```

Subscribe a System by Using the subscription-manager Command

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.
```

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 indicate installed Red Hat products.
- **`/etc/pki/consumer`** certificates identify 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

`rhc(8)`, `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

► 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 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 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.

Explain and Investigate RPM Software Packages

Objectives

- Explain how software is provided as RPM packages, and investigate the DNF and RPM installed system packages.

Software Packages and RPM

The RPM Package Manager, which Red Hat originally developed, provides a standard way to package software for distribution. Managing software in the form of RPM packages is simpler than working with software that is extracted to a file system from an archive. With RPM packages, administrators can track which files the software package installs, which files the software package removes if you uninstall it, and it verifies that supporting packages are present when you install it. The local RPM database on your system stores the information about installed packages. Red Hat provides all software for Red Hat Enterprise Linux as an RPM package.

RPM package file names consist of four elements (plus the `.rpm` suffix): `name-version-release.architecture`:

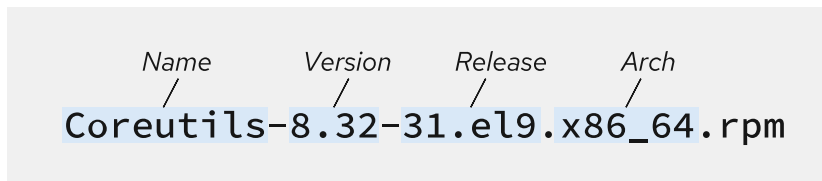


Figure 12.3: RPM file name elements

- NAME is one or more words to describe the contents (`coreutils`).
- VERSION is the version number of the original software (`8.32`).
- RELEASE is the release number of the package based on that version, and is set by the packager, who might not be the original software developer (`31.el9`).
- ARCH is the processor architecture that the package is compiled to run on. The `x86_64` value indicates that this package is built for the 64-bit version of the x86 instruction set (as opposed to `aarch64` for 64-bit ARM, and so on).

RPM packages are often downloaded from repositories. A repository is a central location for storing and maintaining RPM software packages.

You require only the package name to install RPM packages from repositories.

- If multiple versions exist, then the RPM Package Manager installs the package with the later version number.
- If multiple releases of a single version exist, then the RPM Package Manager installs the package with the later release number.

Each RPM package is an archive with the following components:

- The files that the package installs in your system.

- Information about the package (metadata), such as the name, version, release, and architecture; a summary and description of the package; whether it requires other packages to be installed; licensing; a package change log; and other details.
- Scripts that might run when you install, update, or remove the package. These scripts might also run when you install, update, or remove other packages.

Typically, software providers digitally sign RPM packages with GPG (GNU Privacy Guard) keys. (Red Hat digitally signs all packages that it releases.) The RPM system verifies package integrity by confirming that the package is signed with the appropriate GPG key. The RPM system fails to install a package if the GPG signature does not match.

Update Software with RPM Packages

Red Hat generates a complete RPM package to update software. An administrator who installs that package gets only the most recent version of the package. You do not need to install an earlier version of a package to patch it. To update software, RPM removes the earlier version of the package and installs the latest version. Updates usually retain configuration files, but the packager of the new version defines the exact behavior.

Typically, only one version of a package is installed at a time. If a package is built with non-conflicting file names, then you might install multiple versions. The `kernel` package is an example of installing multiple package versions. Because you test a new kernel only by booting to that kernel, the package is designed to support installing multiple versions. If the new kernel fails to boot, then you can revert to the previous kernel.

Examine RPM Packages

The `rpm` utility is a low-level tool that can retrieve information about the contents of package files and installed packages. By default, the tool gets information from a local database of installed packages. Use the `rpm` command `-p` option to get information about a downloaded but uninstalled package file. Use this option to inspect the package contents before installing.

Retrieve general information about installed packages:

- **`rpm -qa`** : List all installed packages.
- **`rpm -qf FILENAME`** : Determine which package provides *FILENAME*.

```
[user@host ~]$ rpm -qf /etc/yum.repos.d
redhat-release-9.1-1.0.el9.x86_64
```

Get information about specific packages:

- **`rpm -q`** : List the currently installed package version.

```
[user@host ~]$ rpm -q dnf
dnf-4.10.0-4.el9.noarch
```

- **`rpm -qi`** : Get detailed package information.
- **`rpm -ql`** : List the files that the package installs.

```
[user@host ~]$ rpm -ql dnf
/usr/bin/dnf
/usr/lib/systemd/system/dnf-makecache.service
/usr/lib/systemd/system/dnf-makecache.timer
/usr/share/bash-completion
/usr/share/bash-completion/completions
/usr/share/bash-completion/completions/dnf
...output omitted...
```

- **rpm -qc** : List only the configuration files that the package installs.

```
[user@host ~]$ rpm -qc openssh-clients
/etc/ssh/ssh_config
/etc/ssh/ssh_config.d/50-redhat.conf
```

- **rpm -qd** : List only the documentation files that the package installs.

```
[user@host ~]$ rpm -qd openssh-clients
/usr/share/man/man1/scp.1.gz
/usr/share/man/man1/sftp.1.gz
/usr/share/man/man1/ssh-add.1.gz
/usr/share/man/man1/ssh-agent.1.gz
...output omitted...
```

- **rpm -q --scripts** : List the shell scripts that run before or after you install or remove the package.

```
[user@host ~]$ rpm -q --scripts openssh-server
preinstall scriptlet (using /bin/sh):
getent group sshd >/dev/null || groupadd -g 74 -r sshd || :
getent passwd sshd >/dev/null || \
    useradd -c "Privilege-separated SSH" -u 74 -g sshd \
    -s /sbin/nologin -r -d /usr/share/empty.sshd sshd 2> /dev/null || :
postinstall scriptlet (using /bin/sh):

if [ $1 -eq 1 ] && [ -x "/usr/lib/systemd/systemd-update-helper" ]; then
    # Initial installation
    /usr/lib/systemd/systemd-update-helper install-system-units sshd.service
    sshd.socket || :
fi
...output omitted...
```

- **rpm -q --changelog** : List the change log information for the package.

```
[user@host ~]$ rpm -q --changelog audit
* Tue Feb 22 2022 Sergio Correia <scorreia@redhat.com> - 3.0.7-101
- Adjust sample-rules dir permissions
  Resolves: rhbz#2054432 - /usr/share/audit/sample-rules is no longer readable by
  non-root users

* Tue Jan 25 2022 Sergio Correia <scorreia@redhat.com> - 3.0.7-100
```

```
- New upstream release, 3.0.7
  Resolves: rhbz#2019929 - capability=unknown-capability(39) in audit messages
  ...output omitted...
```

Query local package files:

- **rpm -qlp** : List the files that the local package installs.

```
[user@host ~]$ ls -l podman-4.0.0-6.el9.x86_64.rpm
-rw-r--r--. 1 student student 13755101 Mar 22 11:35
podman-4.0.0-6.el9.x86_64.rpm2637-15.el9.x86_64.rpm
```

```
[user@host ~]$ rpm -qlp podman-4.0.0-6.el9.x86_64.rpm
/etc/cni/net.d
/etc/cni/net.d/87-podman-bridge.conflist
/usr/bin/podman
...output omitted...
```

Install RPM Packages

Use the `rpm` command to install an RPM package that you downloaded to your local directory.

```
[root@host ~]# rpm -ivh podman-4.0.0-6.el9.x86_64.rpm
Verifying...                               ##### [100%]
Preparing...                               ##### [100%]
Updating / installing...
      podman-2:4.0.0-6                       ##### [100%]
```



Warning

Be careful when installing packages from third parties, not just because of the software that the packages might install, but because the RPM package might include arbitrary scripts that run as the `root` user as part of the installation process.

Extracting RPM packages

Use the `rpm2cpio` command to extract files from an RPM package file without installing the package.

The `rpm2cpio` command converts an RPM package to a `cpio` archive. After the RPM package is converted to a `cpio` archive, the `cpio` command can extract a list of files.

Use the `cpio` command with the `-i` option to extract files from standard input. Use the `-d` option to create subdirectories as needed, starting in the current working directory. Use the `-v` option for verbose output.

```
[user@host tmp-extract]$ rpm2cpio httpd-2.4.51-7.el9_0.x86_64.rpm | cpio -idv
./etc/httpd/conf
./etc/httpd/conf.d/autoindex.conf
./etc/httpd/conf.d/userdir.conf
./etc/httpd/conf.d/welcome.conf
./etc/httpd/conf.modules.d
```



```

./etc/httpd/conf.modules.d/00-base.conf
./etc/httpd/conf.modules.d/00-dav.conf
./etc/httpd/conf.modules.d/00-mpm.conf
./etc/httpd/conf.modules.d/00-optional.conf
./etc/httpd/conf.modules.d/00-proxy.conf
./etc/httpd/conf.modules.d/00-systemd.conf
./etc/httpd/conf.modules.d/01-cgi.conf
./etc/httpd/conf.modules.d/README
./etc/httpd/conf/httpd.conf
...output omitted...
9774 blocks
[user@host tmp-extract]$ ls -l
total 1552
drwxr-xr-x. 5 user user      55 Feb  3 15:06 etc
-rw-r--r--. 1 user user 1588633 Feb  3 15:06 httpd-2.4.51-7.el9_0.x86_64.rpm
drwxr-xr-x. 3 user user      19 Feb  3 15:06 run
drwxr-xr-x. 7 user user      70 Feb  3 15:06 usr
drwxr-xr-x. 5 user user      41 Feb  3 15:06 var

```

Extract individual files by specifying the path of the file:

```

[user@host ~]$ rpm2cpio httpd-2.4.51-7.el9_0.x86_64.rpm | cpio -id
"*/etc/httpd/conf/httpd.conf"
9774 blocks
[user@host ~]$ ls etc/httpd/conf/
httpd.conf

```

Use the `rpm2cpio` and `cpio -t` commands to list the files in an RPM package. Use the `-v` option of the `cpio` command for verbose output.

```

[student@servera ~]$ rpm2cpio httpd-2.4.51-7.el9_0.x86_64.rpm | cpio -tv
drwxr-xr-x  1 root    root           0 Mar 21  2022 ./etc/httpd/conf
-rw-r--r--  1 root    root        2893 Mar 21  2022 ./etc/httpd/conf.d/
autoindex.conf
-rw-r--r--  1 root    root        1252 Mar 21  2022 ./etc/httpd/conf.d/
userdir.conf
-rw-r--r--  1 root    root         653 Mar 21  2022 ./etc/httpd/conf.d/
welcome.conf
drwxr-xr-x  1 root    root           0 Mar 21  2022 ./etc/httpd/conf.modules.d
-rw-r--r--  1 root    root       3372 Mar 21  2022 ./etc/httpd/
conf.modules.d/00-base.conf
-rw-r--r--  1 root    root         139 Mar 21  2022 ./etc/httpd/
conf.modules.d/00-dav.conf
-rw-r--r--  1 root    root         948 Mar 21  2022 ./etc/httpd/
conf.modules.d/00-mpm.conf
-rw-r--r--  1 root    root         787 Mar 21  2022 ./etc/httpd/
conf.modules.d/00-optional.conf
-rw-r--r--  1 root    root       1073 Mar 21  2022 ./etc/httpd/
conf.modules.d/00-proxy.conf
-rw-r--r--  1 root    root          88 Mar 21  2022 ./etc/httpd/
conf.modules.d/00-systemd.conf
-rw-r--r--  1 root    root         367 Mar 21  2022 ./etc/httpd/
conf.modules.d/01-cgi.conf

```

```
-rw-r--r--  1 root    root          496 Mar 21  2022 ./etc/httpd/conf.modules.d/
README
-rw-r--r--  1 root    root       12005 Mar 21  2022 ./etc/httpd/conf/httpd.conf
...output omitted...
9774 blocks
```



References

rpm(8), rpm2cpio(8), cpio(1), and rpmkeys(8) man pages

► Guided Exercise

Explain and Investigate RPM Software Packages

Gather information about a package from a third party, extract files from it for inspection, and then install it on a server.

Outcomes

- Install on a server a package that is not from the software repositories.

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-rpm
```

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. View package information and list files in the `rhcsa-script-1.0.0-1.noarch.rpm` package. Also view the script that runs when you install or uninstall the package.

2.1. View information for the `rhcsa-script-1.0.0-1.noarch.rpm` package.

```
[student@servera ~]$ 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.

```

**Note**

The preceding package modifies the *MOTD*, or "Message of the Day". A system displays the MOTD to users as they log in to systems.

- 2.2. List files in the `rhcsa-script-1.0.0-1.noarch.rpm` package.

```

[student@servera ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm -l
/opt/rhcsa-script/mymotd

```

- 2.3. View the script that runs when you install or uninstall the `rhcsa-script-1.0.0-1.noarch.rpm` package.

```

[student@servera ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm --scripts
preinstall scriptlet (using /bin/sh):
if [ "$1" == "2" ]; then
    if [ -e /etc/motd.orig ]; then
        mv -f /etc/motd.orig /etc/motd
    fi
fi
postinstall scriptlet (using /bin/sh):
...output omitted...

```

- ▶ 3. Extract the contents of the `rhcsa-script-1.0.0-1.noarch.rpm` package to the `/home/student` directory.

- 3.1. Use the `rpm2cpio` and `cpio -tv` commands to list the files in the `rhcsa-script-1.0.0-1.noarch.rpm` package.

```

[student@servera ~]$ rpm2cpio rhcsa-script-1.0.0-1.noarch.rpm | cpio -tv
-rw-r--r--  1 root    root          593 Mar 23 08:24 ./opt/rhcsa-script/mymotd
2 blocks

```

- 3.2. Extract all files from the `rhcsa-script-1.0.0-1.noarch.rpm` package to the `/home/student` directory. Use the `rpm2cpio` and `cpio -idv` commands to extract the files and create the parent directories where needed in verbose mode.

```

[student@servera ~]$ rpm2cpio rhcsa-script-1.0.0-1.noarch.rpm | cpio -idv
./opt/rhcsa-script/mymotd
2 blocks

```

- 3.3. List the files in the `/home/student/opt` directory to verify that the extracted files are the same as the files inside the package.

```
[student@servera ~]$ ls -lR opt
opt:
total 0
drwxr-xr-x. 2 student student 20 Mar 23 09:22 rhcsa-script

opt/rhcsa-script:
total 4
-rw-r--r--. 1 student student 593 Mar 23 09:22 mymotd
```

- ▶ 4. Install the `rhcsa-script-1.0.0-1.noarch.rpm` package. Use the `sudo` command to gain superuser privileges to install the package.

- 4.1. Use the `sudo rpm -ivh` command to install the `rhcsa-script-1.0.0-1.noarch.rpm` RPM package.

```
[student@servera ~]$ sudo rpm -ivh rhcsa-script-1.0.0-1.noarch.rpm
[sudo] password for student: student
Verifying... ##### [100%]
Preparing... ##### [100%]
Updating / installing...
   1:rhcsa-script-1.0.0-1 ##### [100%]
[student@servera ~]$
```

- 4.2. Use the `rpm` command to verify that you correctly installed the package.

```
[student@servera ~]$ rpm -q rhcsa-script
rhcsa-script-1.0.0-1.noarch
```

- **5.** Exit from the `servera` machine and connect again to test the new message of the day.

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

Activate the web console with: `systemctl enable --now cockpit.socket`

```
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Wed Mar 23 09:21:26 2022 from 172.25.250.9
[student@servera ~]$
```

- ▶ 6. 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 software-rpm
```

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 that are 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 PACKAGENAME` 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 PACKAGENAME` 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.rp 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 PACKAGENAME` command obtains and installs a later 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 earlier name will not work after the update. If no `PACKAGENAME` is specified, then 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 earlier 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 PACKAGENAME` 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 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.

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 PACKAGENAME</code>
Install a package.	<code>dnf install PACKAGENAME</code>
Install a package group.	<code>dnf group install GROUPNAME</code>
Update all packages.	<code>dnf update</code>
Remove a package.	<code>dnf remove PACKAGENAME</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 the same lifecycle as 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, because 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 similar modules to regular packages.

See the following list for some important commands to manage modules:

- **`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

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

Modularity

<https://docs.fedoraproject.org/en-US/modularity/>

► Guided Exercise

Install and Update Software Packages with DNF

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

- 2. Search for a specific package.
 - 2.1. Try 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
  htmaccalc
  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     : 7743aed72ac79f632442c9028aafd2499a1591f92a660b3f09219b422ca95f02
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
```

Enable DNF Software Repositories

Objectives

- Enable and disable server use of Red Hat or third-party DNF repositories.

Enable Red Hat Software Repositories

Systems often have access to many 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 buy, 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 verify 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
```

The `rpm` command uses GPG keys to sign packages, and imports public keys to verify the integrity and authenticity of packages. The `dnf` command uses repository configuration files to provide the GPG public key locations, and imports the keys to verify the packages. 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 for 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 a GPG public key as part of an RPM package to simplify their installation. You can import the GPG public key by using the `rpm --import` command. The `dnf install` command can download and install these RPM packages.

For example, the following command imports the RPM-GPG-KEY-EPEL-9 (EPEL) GPG public key and installs the RHEL9 Extra Packages for Enterprise Linux (EPEL) repository RPM:

```
[user@host ~]$ rpm --import \
https://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
```



Warning

Import the RPM GPG key before installing signed packages, to ensure that packages come from a trusted source. If the RPM GPG key is not imported, 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.

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 enable and disable repositories temporarily while the command runs.



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

► Guided Exercise

Enable DNF Software Repositories

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 at `http://content.example.com/rhel9.3/x86_64/rhcsa-practice/rht`
- Updates of the custom packages at `http://content.example.com/rhel9.3/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/rhel9.3/x86_64/rhcsa-practice/rht"
Adding repo from: http://content.example.com/rhel9.3/x86_64/rhcsa-practice/rht
```

- 2.2. Examine the software repository file that the previous command created in the `/etc/yum.repos.d` directory. Edit the `/etc/yum.repos.d/`

content.example.com_rhel9.3_x86_64_rhcsa-practice_rht.repo file, and add the `gpgcheck=0` parameter to disable the GPG key check for the repository.

```
[content.example.com_rhel9.3_x86_64_rhcsa-practice_rht]
name=created by dnf config-manager from http://content.example.com/rhel9.3/x86_64/
rhcsa-practice/rht
baseurl=http://content.example.com/rhel9.3/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/rhel9.3/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_rhel9.3_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_rhel9.3_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_rhel9.3_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 rht-updates` 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
```

► Lab

Install and Update Software Packages

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/rhel9.3/x86_64/rhcsa-practice/errata` repository. Do not verify 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 by 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
```

► Solution

Install and Update Software Packages

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/rhel9.3/x86_64/rhcsa-practice/errata` repository. Do not verify 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/rhel9.3/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 by 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
```


Summary

- RHEL 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.
- Software is provided as RPM packages, to install, upgrade, and uninstall software on the system.
- The `rpm` command can query a local database to provide information about the contents of installed packages and to install downloaded package files.
- The `dnf` utility is a powerful command-line tool to install, update, remove, and query software packages.
- Red Hat Enterprise Linux uses Application Streams to provide a single repository to host multiple versions of an application's packages and its dependencies.

Chapter 13

Access Linux File Systems

Goal

Access, inspect, and use existing file systems on storage that is attached to a Linux server.

Objectives

- Identify a directory in the file-system hierarchy and the device where it is stored.
- Access the contents of file systems by adding and removing file systems in the file-system hierarchy.
- Search for files on mounted file systems with the `find` and `locate` commands.

Sections

- Identify File Systems and Devices (and Quiz)
- Mount and Unmount File Systems (and Guided Exercise)
- Locate Files on the System (and Guided Exercise)

Lab

- Access Linux File Systems

Identify File Systems and Devices

Objectives

- Identify a directory in the file-system hierarchy and the device where it is stored.

Storage Management Concepts

Red Hat Enterprise Linux (RHEL) uses the *Extents File System (XFS)* as the default local file system. RHEL supports the *Extended File System (ext4)* file system for managing local files. Starting with RHEL 9, the Extensible File Allocation Table (exFAT) file system is supported for removable media use. In an enterprise server cluster, shared disks use the *Global File System 2 (GFS2)* file system to manage concurrent multi-node access.

File Systems and Mount Points

Access the contents of a file system by mounting it on an empty directory. This directory is called a mount point. When the directory is mounted, use the `ls` command to list its contents. Many file systems are automatically mounted when the system boots.

A mount point differs slightly from a Microsoft Windows drive letter, where each file system is a separate entity. Mount points enable multiple file system devices to be available in a single tree structure. This mount point is similar to NTFS mounted folders in Microsoft Windows.

File Systems, Storage, and Block Devices

A block device is a file that provides low-level access to storage devices. A block device must be optionally partitioned, and a file system that was created before the device can be mounted.

The `/dev` directory stores block device files, which RHEL creates automatically for all devices. In RHEL 9, the first detected SATA, SAS, SCSI, or USB hard drive is called the `/dev/sda` device; the second is the `/dev/sdb` device; and so on. These names represent the entire hard drive.

Block Device Naming

Type of device	Device naming pattern
SATA/SAS/USB-attached storage (SCSI driver)	<code>/dev/sda</code> , <code>/dev/sdb</code> , <code>/dev/sdc</code> , ...
<code>virtio-blk</code> paravirtualized storage (VMs)	<code>/dev/vda</code> , <code>/dev/vdb</code> , <code>/dev/vdc</code> ,...
<code>virtio-scsi</code> paravirtualized storage (VMs)	<code>/dev/sda</code> , <code>/dev/sdb</code> , <code>/dev/sdc</code> , ...
NVMe-attached storage (SSDs)	<code>/dev/nvme0</code> , <code>/dev/nvme1</code> , ...
SD/MMC/eMMC storage (SD cards)	<code>/dev/mmcblk0</code> , <code>/dev/mmcblk1</code> , ...

Disk Partitions

Usually, the entire storage device is not created into one file system. To create a partition, divide the storage devices into smaller chunks.

With partitions, you can compartmentalize a disk: the various partitions might be formatted with different file systems or be used for other purposes. For example, one partition might contain user home directories, whereas another partition might contain system data and logs. Even when the home directory partition is loaded with data, the system partition might still have available space.

Partitions are block devices in their own right. For example, on the first SATA-attached storage, the first partition is the `/dev/sda1` disk. The second partition of the same storage is the `/dev/sda2` disk. The third partition on the third SATA-attached storage device is the `/dev/sdc3` disk, and so on. Paravirtualized storage devices have a similar naming system. For example, the first partition on the first storage device is the `/dev/vda1` disk. The second partition of the second storage device is the `/dev/vdb2` disk, and so on.

An NVMe-attached SSD device names its partitions differently from a SATA-attached device. For NVMe storage devices, the `nvmeX` part of the name refers to the device; the `nY` part refers to the namespace; and the `pZ` part refers to the partition. For example, the first partition for the first namespace on the first disk is the `/dev/nvme0n1p1` partition. The third partition for the first namespace on the second disk is the `/dev/nvme1n1p3` partition, and so on.

SD or MMC cards can sometimes have a similar naming system to the SATA devices (`/dev/sdN`). In some cases, SD or MMC cards might have names such as `/dev/mmcblk0p1`, where the `mmcblkX` part of the name refers to the storage device, and the `pY` part of the name refers to the partition number on that device.

An extended listing of the `/dev/sda1` device file on the host machine reveals the `b` file type, which stands for a block device:

```
[user@host ~]$ ls -l /dev/sda1
brw-rw----. 1 root disk 8, 1 Feb 22 08:00 /dev/sda1
```

Logical Volumes

Another way of organizing disks and partitions is with *Logical Volume Management (LVM)*. With LVM, it is possible to aggregate block devices into a volume group. Disk space in the volume group is separated into logical volumes, which are the functional equivalent of a partition on a physical disk.

The LVM system assigns names to volume groups and logical volumes on their creation. LVM creates a directory in the `/dev` directory that matches the group name, and creates a symbolic link within that new directory with the same name as the logical volume. That logical volume file is then available to be mounted. For example, when a `myvg` volume group and the `mylv` logical volume are present, the full path to the logical volume is the `/dev/myvg/mylv` file.



Note

The previously mentioned logical volume device name establishes a symbolic link to the device file that accesses it, which might vary between boots. Another form of logical volume device name, which is linked from files in the `/dev/mapper` directory, is often used for symbolic links to the device file.

Examine File Systems

Use the `df` command to display an overview of local and remote file-system devices, which includes the total disk space, used disk space, free disk space, and the percentage of the entire disk space.

The following example displays the file systems and mount points on the host machine:

```
[user@host ~]$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
devtmpfs         912584         0    912584   0% /dev
tmpfs            936516         0    936516   0% /dev/shm
tmpfs            936516    16812    919704   2% /run
tmpfs            936516         0    936516   0% /sys/fs/cgroup
/dev/vda3       8377344 1411332 6966012  17% /
/dev/vda1       1038336 169896 868440  17% /boot
tmpfs           187300         0    187300   0% /run/user/1000
```

The partitioning shows that two physical file systems are mounted on the `/` and `/boot` directories that commonly exist on virtual machines. The `tmpfs` and `devtmpfs` devices are file systems in system memory. All files that are written to the `tmpfs` or `devtmpfs` file system disappear after a system reboot.

The `df` command `-h` or `-H` options are human-readable options to improve the readability of the output sizes. The `-h` option reports in KiB (2^{10}), MiB (2^{20}), or GiB (2^{30}), whereas the `-H` option reports in SI units: KB (10^3), MB (10^6), or GB (10^9). Hard drive manufacturers usually use SI units when advertising their products.

View the file systems on the host machine with all units converted to human-readable format:

```
[user@host ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        892M   0  892M   0% /dev
tmpfs           915M   0  915M   0% /dev/shm
tmpfs           915M  17M  899M   2% /run
tmpfs           915M   0  915M   0% /sys/fs/cgroup
/dev/vda3       8.0G  1.4G  6.7G  17% /
/dev/vda1       1014M 166M  849M  17% /boot
tmpfs           183M   0  183M   0% /run/user/1000
```

Use the `du` command for more detailed information about a specific directory tree space. The `du` command `-h` and `-H` options convert the output to a human-readable format. The `du` command shows the size of all files in the current directory tree recursively.

View the disk usage report for the `/usr/share` directory on the host machine:

```
[root@host ~]# du /usr/share
...output omitted...
176 /usr/share/smartmontools
184 /usr/share/nano
8 /usr/share/cmake/bash-completion
8 /usr/share/cmake
356676 /usr/share
```

View the disk usage report in human-readable format for the `/usr/share` directory:

```
[root@host ~]# du -h /usr/share
...output omitted...
176K  /usr/share/smartmontools
184K  /usr/share/nano
8.0K  /usr/share/cmake/bash-completion
8.0K  /usr/share/cmake
369M  /usr/share
```



References

df(1) and du(1) man pages

► Quiz

Identify File Systems and Devices

Choose the correct answers to the following questions:

- 1. What is the device file name of an entire SATA hard drive in the /dev directory?
 - a. /dev/vda
 - b. /dev/sda1
 - c. /dev/vg_install/lv_home
 - d. /dev/sda
- 2. Which command displays the file systems with the mount points?
 - a. du -H
 - b. df
 - c. du
 - d. ls
- 3. Which command displays the disk usage report in human-readable format for the /home directory?
 - a. ls /home
 - b. df
 - c. du -h /home
 - d. du /home
- 4. What is the correct device file name for the third partition on the second virtio-blk disk that is attached to a virtual machine?
 - a. /dev/vdb3
 - b. /dev/vda2
 - c. /dev/sda3
 - d. /dev/vda3
- 5. Which command provides an overview of the file-system mount points and the available free space in SI units?
 - a. df
 - b. df -h
 - c. df -H
 - d. du -h

► Solution

Identify File Systems and Devices

Choose the correct answers to the following questions:

- 1. What is the device file name of an entire SATA hard drive in the /dev directory?
 - a. /dev/vda
 - b. /dev/sda1
 - c. /dev/vg_install/lv_home
 - d. /dev/sda
- 2. Which command displays the file systems with the mount points?
 - a. du -H
 - b. df
 - c. du
 - d. ls
- 3. Which command displays the disk usage report in human-readable format for the /home directory?
 - a. ls /home
 - b. df
 - c. du -h /home
 - d. du /home
- 4. What is the correct device file name for the third partition on the second virtio-blk disk that is attached to a virtual machine?
 - a. /dev/vdb3
 - b. /dev/vda2
 - c. /dev/sda3
 - d. /dev/vda3
- 5. Which command provides an overview of the file-system mount points and the available free space in SI units?
 - a. df
 - b. df -h
 - c. df -H
 - d. du -h

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 use as a mount point is not empty, then the existing files are hidden and not accessible when a file system is mounted there. The original files are 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 to 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

With 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 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. For the `umount` command to succeed, all processes must stop accessing data under the mount point.

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
```

Identify and wait for the processes to complete, or send the `SIGTERM` or `SIGKILL` signal to terminate them. In this case, it is sufficient to change to a current working directory that is 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

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.

```
[root@servera ~]# lsblk -fp /dev/vdb
```

NAME	FSTYPE	LABEL	UUID	MOUNTPOINT
/dev/vdb				
└─/dev/vdb1	xf		a04c511a-b805-4ec2-981f-42d190fc9a65	

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 when 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
```

Locate Files on the System

Objectives

- Search for files on mounted file systems with the `find` and `locate` commands.

Search for Files

A system administrator needs tools to search for files that match specific criteria on the file system. This section discusses two commands to search for files in the file-system hierarchy:

- The `locate` command searches a pre-generated index for file names or file paths, and returns the results instantly.
- The `find` command searches for files in real time by parsing the file-system hierarchy.

Locate Files by Name

The `locate` command searches for files based on the name or path to the file. The command is fast, because it looks up this information from the `mlocate` database. However, this database does not update in real time and requires frequent updates for accurate results. This feature also means that the `locate` command does not search for files that were created since the last database update.

The `locate` database updates automatically every day. However, the `root` user might issue the `updatedb` command to force an immediate update.

```
[root@host ~]# updatedb
```

The `locate` command restricts results for unprivileged users. To see the resulting file name, the user must have search permission on the directory where the file resides. For example, locate the files that the `developer` user can read, and that match the `passwd` keyword in the name or path:

```
[developer@host ~]$ locate passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
...output omitted...
```

The following example shows the file name or path for a partial match with the search query:

```
[root@host ~]# locate image
/etc/selinux/targeted/contexts/virtual+_+image_context
/usr/bin/grub2-mkimage
/usr/lib/sysimage
...output omitted...
```

The `locate` command `-i` option performs a case-insensitive search. This option returns all possible combinations of matching uppercase and lowercase letters:


```
[developer@host ~]$ locate -i messages
...output omitted...
/usr/share/locale/zza/LC_MESSAGES
/usr/share/makedumpfile/eppic_scripts/ap_messages_3_10_to_4_8.c
/usr/share/vim/vim82/ftplugin/msmessages.vim
...output omitted...
```

The `locate` command `-n` option limits the number of returned search results. The following example limits the search results from the `locate` command to the first five matches:

```
[developer@host ~]$ locate -n 5 passwd
/etc/passwd
/etc/passwd-
/etc/pam.d/passwd
...output omitted...
```

Search for Files in Real Time

The `find` command locates files by searching in real time in the file-system hierarchy. This command is slower but more accurate than the `locate` command. The `find` command also searches for files based on criteria other than the file name, such as the file's permissions, type of file, size, or modification time.

The `find` command looks at files in the file system with the user account that executed the search. The user that runs the `find` command must have read and execute permission on a directory to examine its contents.

The first argument to the `find` command is the directory to search. If the `find` command omits the directory argument, then it starts the search in the current directory and looks for matches in any subdirectory.

To search for files by file name, use the `find` command `-name FILENAME` option to return the path of files that match `FILENAME` exactly. For example, to search for the `sshd_config` files in the root `/` directory, run the following command:

```
[root@host ~]# find / -name sshd_config
/etc/ssh/sshd_config
```



Note

In the `find` command, the complete word options use a single dash for options, unlike a double dash for most other Linux commands.

Wildcards are available to search for a file name and to return all results for a partial match. With wildcards, it is essential to quote the file name, to prevent the terminal from misinterpreting the wildcard.

In the following example, starting in the `/` directory, search for files that end with the `.txt` extension:

```
[root@host ~]# find / -name '*.txt'
...output omitted...
/usr/share/libgpg-error/errorref.txt
/usr/share/licenses/audit-libs/lgpl-2.1.txt
/usr/share/licenses/pam/gpl-2.0.txt
...output omitted...
```

To search for files in the `/etc/` directory that contain the `pass` string, run the following command:

```
[root@host ~]# find /etc -name '*pass*'
/etc/passwd-
/etc/passwd
/etc/security/opasswd
...output omitted...
```

To perform a case-insensitive search for a file name, use the `find` command `-iname` option, followed by the file name to search. To search files with case-insensitive text that matches the `messages` string in their names in the root `/` directory, run the following command:

```
[root@host ~]# find / -iname '*messages*'
/sys/power/pm_debug_messages
/usr/lib/locale/C.utf8/LC_MESSAGES
/usr/lib/locale/C.utf8/LC_MESSAGES/SYS_LC_MESSAGES
...output omitted...
```

Search for Files Based on Ownership or Permission

The `find` command searches for files based on their ownership or permissions. The `find` command `-user` and `-group` options search by a user and group name, or by user ID and group ID.

To search for files in the `/home/developer` directory that the `developer` user owns:

```
[developer@host ~]$ find -user developer
.
./.bash_logout
./.bash_profile
...output omitted...
```

To search for files in the `/home/developer` directory that the `developer` group owns:

```
[developer@host ~]$ find -group developer
.
./.bash_logout
./.bash_profile
...output omitted...
```

To search for files in the `/home/developer` directory that the `1000` user ID owns:

```
[developer@host ~]$ find -uid 1000
.
./.bash_logout
./.bash_profile
...output omitted...
```

To search for files in the `/home/developer` directory that the 1000 group ID owns:

```
[developer@host ~]$ find -gid 1000
.
./.bash_logout
./.bash_profile
...output omitted...
```

The `find` command `-user` and `-group` options search for files where the file owner and group owner are different. The following example lists files that the `root` user owns and with the `mail` group:

```
[root@host ~]# find / -user root -group mail
/var/spool/mail
...output omitted...
```

The `find` command `-perm` option looks for files with a particular permission set. The octal values define the permissions with 4, 2, and 1 for read, write, and execute. Permissions are preceded with a `/` or `-` sign to control the search results.

Octal permission that is preceded by the `/` sign matches files where at least one permission is set for user, group, or other for that permission set. A file with the `r--r--r--` permissions does not match the `/222` permission but matches the `rw-r--r--` permission. A `-` sign before the permission means that all three parts of the permissions must match. For the previous example, files with the `rw-rw-rw-` permissions match. You can also use the `find` command `-perm` option with the symbolic method for permissions.

For example, the following commands match any file in the `/home` directory for which the owning user has read, write, and execute permissions, and members of the owning group have read and write permissions, and others have read-only access. Both commands are equivalent; the first one uses the octal method for permissions, whereas the second one uses the symbolic methods.

```
[root@host ~]# find /home -perm 764
...output omitted...
[root@host ~]# find /home -perm u=rwx,g=rw,o=r
...output omitted...
```

The `find` command `-ls` option is convenient when searching files by permissions, because it provides information for the files that includes their permissions.

```
[root@host ~]# find /home -perm 764 -ls
26207447  0 -rwxrw-r--  1 user  user   0 May 10 04:29 /home/user/file1
```

To search for files for which the user has at least write and execute permissions, and the group has at least write permission, and others have at least read permission, run the following command:

```
[root@host ~]# find /home -perm -324
...output omitted...
[root@host ~]# find /home -perm -u=wx,g=w,o=r
...output omitted...
```

To search for files for which the user has read permissions, or the group has at least read permissions, or others have at least write permission, run the following command:

```
[root@host ~]# find /home -perm /442
...output omitted...
[root@host ~]# find /home -perm /u=r,g=r,o=w
...output omitted...
```

When used with / or - signs, the 0 value works as a wildcard, because it means any permission.

To search for any file in the /home/developer directory for which others have at least read access on the host machine, run the following command:

```
[developer@host ~]$ find -perm -004
...output omitted...
[developer@host ~]$ find -perm -o=r
...output omitted...
```

To search for all files in the /home/developer directory where others have write permission, run the following command:

```
[developer@host ~]$ find -perm -002
...output omitted...
[developer@host ~]$ find -perm -o=w
...output omitted...
```

Find Files Based on Size

The `find` command `-size` option is followed by a numeric value, and the unit looks up files that match a specified size. Use the following list for the units with the `find` command `-size` option:

- For kilobytes, use the `k` unit with `k` always in lowercase.
- For megabytes, use the `M` unit with `M` always in uppercase.
- For gigabytes, use the `G` unit with `G` always in uppercase.

You can use the plus `+` and minus `-` characters to include files that are larger and smaller than the given size, respectively. The following example shows a search for files with an exact size of 10 megabytes:

```
[developer@host ~]$ find -size 10M
...output omitted...
```

To search for files with a size of more than 10 gigabytes:

```
[developer@host ~]$ find -size +10G
...output omitted...
```

To search for files with a size of less than 10 kilobytes:

```
[developer@host ~]$ find -size -10k
...output omitted...
```



Important

The `find` command `-size` option rounds everything to single units. For example, the `find -size 1M` command shows files that are smaller than 1 MB, because it rounds up all files to 1 MB.

Search for Files Based on Modification Time

The `find` command `-mmin` option, followed by the time in minutes, searches for all files with content that changed `n` minutes ago. The file's time stamp is rounded down and supports fractional values with the `+n` and `-n` range.

To search for all files with content that changed 120 minutes ago:

```
[root@host ~]# find / -mmin 120
...output omitted...
```

The `+` modifier in front of the minutes finds all files in the `/` directory that changed more than `n` minutes ago. To search for all files with content that changed more than 200 minutes ago:

```
[root@host ~]# find / -mmin +200
...output omitted...
```

The `-` modifier searches for all files in the `/` directory that changed less than `n` minutes ago. The following example lists files that changed less than 150 minutes ago:

```
[root@host ~]# find / -mmin -150
...output omitted...
```

Search for Files Based on File Type

The `find` command `-type` option limits the search scope to a given file type. Use the following flags to limit the search scope:

- For regular files, use the `f` flag.
- For directories, use the `d` flag.
- For soft links, use the `l` flag.
- For block devices, use the `b` flag.

Search for all directories in the `/etc` directory:

```
[root@host ~]# find /etc -type d
/etc
/etc/tmpfiles.d
/etc/systemd
/etc/systemd/system
/etc/systemd/system/getty.target.wants
...output omitted...
```

Search for all soft links in the / directory:

```
[root@host ~]# find / -type l
...output omitted...
```

Search for all block devices in the /dev directory:

```
[root@host ~]# find /dev -type b
/dev/vda1
/dev/vda
```

The **find** command **-links** option followed by a number looks for all files with a specific hard link count. The number preceded by a **+** modifier looks for files with a higher count than the given hard link count. If the number precedes a **-** modifier, then the search is limited to files with a lower hard link count than the given number.

Search for all regular files with more than one hard link:

```
[root@host ~]# find / -type f -links +1
...output omitted...
```



References

locate(1), updatedb(8), and find(1) man pages

► Guided Exercise

Locate Files on the System

Search for specific files on mounted file systems by using the `find` and `locate` commands.

Outcomes

- Search for files with the `find` and `locate` commands.

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-locate
```

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. Use the `locate` command to search for files on the `servera` machine.

- 2.1. Update the `locatedb` database manually on the `server` machine. Use the `sudo updatedb` command to update the database.

```
[student@servera ~]$ sudo updatedb
[sudo] password for student: student
[student@servera ~]$
```

- 2.2. Locate the `logrotate.conf` configuration file.

```
[student@servera ~]$ locate logrotate.conf
/etc/logrotate.conf
/usr/share/man/man5/logrotate.conf.5.gz
```

- 2.3. Locate the `networkmanager.conf` configuration file, ignoring case sensitivity.

```
[student@servera ~]$ locate -i networkmanager.conf
/etc/NetworkManager/NetworkManager.conf
/etc/dbus-1/system.d/org.freedesktop.NetworkManager.conf
/usr/share/man/man5/NetworkManager.conf.5.gz
```

- 3. Use the `find` command to search in real time on the `servera` machine according to the following requirements:

- List all files in the `/var/lib` directory that the `chrony` user owns.
- List all files in the `/var` directory that the `root` user and the `mail` group own.
- List all files in the `/usr/bin` directory with a file size that is greater than 50 KB.
- List all files in the `/home/student` directory that changed in the last 120 minutes.
- List all the block device files in the `/dev` directory.

- 3.1. Search for all files in the `/var/lib` directory that the `chrony` user owns, with root privilege.

```
[student@servera ~]$ sudo find /var/lib -user chrony
[sudo] password for student: student
/var/lib/chrony
/var/lib/chrony/drift
```

- 3.2. List all files in the `/var` directory that the `root` user owns and that belong to the `mail` group.

```
[student@servera ~]$ sudo find /var -user root -group mail
/var/spool/mail
```

- 3.3. List all files in the `/usr/bin` directory with a greater file size than 50 KB.

```
[student@servera ~]$ find /usr/bin -size +50k
/usr/bin/iconv
/usr/bin/locale
/usr/bin/localedef
/usr/bin/cmp
...output omitted...
```

- 3.4. List all files in the `/home/student` directory that changed in the last 120 minutes.

```
[student@servera ~]$ find /home/student -mmin -120
/home/student/.bash_logout
/home/student/.bash_profile
/home/student/.bashrc
...output omitted...
```

- 3.5. List all block device files in the `/dev` directory.


```
[student@servera ~]$ find /dev -type b
/dev/vdd
/dev/vdc
/dev/vdb
/dev/vda3
/dev/vda2
/dev/vda1
/dev/vda
```

- 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 fs-locate
```

► Lab

Access Linux File Systems

Mount a local file system and locate specific files on that file system.

Outcomes

- Mount a file system.
- Generate a disk usage report.
- Find files in the local 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-review
```

Instructions

1. On the `serverb` machine as the `root` user, identify the UUID for the `/dev/vdb1` device, and mount it by using its UUID on the `/mnt/freespace` directory.
2. Generate a disk usage report for the `/usr/share` directory. Save the result in the `/mnt/freespace/results.txt` file.
3. Locate all the files that match the `rsyslog.conf` keyword, and store the result in the `/mnt/freespace/search1.txt` file.
4. Store in the `/mnt/freespace/search2.txt` file the search result of all files in the `/usr/share` directory that are greater than 50 MB and less than 100 MB.
5. Return to the `workstation` system as the `student` user.

Evaluation

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

```
[student@workstation ~]$ lab grade fs-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 fs-review
```

► Solution

Access Linux File Systems

Mount a local file system and locate specific files on that file system.

Outcomes

- Mount a file system.
- Generate a disk usage report.
- Find files in the local 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-review
```

Instructions

1. On the `serverb` machine as the `root` user, identify the UUID for the `/dev/vdb1` device, and mount it by using its UUID on the `/mnt/freespace` directory.

- 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. Query the UUID of the `/dev/vdb1` device.

```
[root@serverb ~]# lsblk -fp /dev/vdb
NAME                FSTYPE FSVER LABEL UUID                                 FSAVAIL FSUSE%
MOUNTPOINTS
/dev/vdb
└─/dev/vdb1 xfs                                     44bf7c8-970c-4d0b-b53d-90ae31cb27ca
```

- 1.3. Create the `/mnt/freespace` directory.

```
[root@serverb ~]# mkdir /mnt/freespace
```

- 1.4. Mount the `/dev/vdb1` device by using the UUID on the `/mnt/freespace` directory.

```
[root@serverb ~]# mount UUID="44bfb7c8-970c-4d0b-b53d-90ae31cb27ca" /mnt/freespace
```

- 1.5. Verify that the `/dev/vdb1` device is mounted on the `/mnt/freespace` directory.

```
[root@serverb ~]# lsblk -fp /dev/vdb1
```

NAME	FSTYPE	FSVER	LABEL	UUID	FSAVAIL	FSUSE%
MOUNTPOINTS						
/dev/vdb1						
	xfs			44bfb7c8-970c-4d0b-b53d-90ae31cb27ca	4.9G	1% /
mnt/freespace						

2. Generate a disk usage report for the `/usr/share` directory. Save the result in the `/mnt/freespace/results.txt` file.

```
[root@serverb ~]# du /usr/share > /mnt/freespace/results.txt
```

3. Locate all the files that match the `rsyslog.conf` keyword, and store the result in the `/mnt/freespace/search1.txt` file.

- 3.1. Update the `locate` database.

```
[root@serverb ~]# updatedb
```

- 3.2. Use the `locate` command to find all the files that match the `rsyslog.conf` keyword, and save the result in the `/mnt/freespace/search1.txt` file.

```
[root@serverb ~]# locate rsyslog.conf > /mnt/freespace/search1.txt
```

4. Store in the `/mnt/freespace/search2.txt` file the search result of all files in the `/usr/share` directory that are greater than 50 MB and less than 100 MB.

```
[root@serverb ~]# find /usr/share -size +50M -size -100M > \
/mnt/freespace/search2.txt
```

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

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

Evaluation

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

```
[student@workstation ~]$ lab grade fs-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 fs-review
```

Summary

- Storage devices are represented by the *block device* file type.
- The `df` command reports total disk space, used disk space, and free disk space on all mounted regular file systems.
- The `root` user can use the `mount` command to manually mount a file system.
- To successfully unmount a device, all processes must stop accessing the mount point.
- The removable storage devices are mounted in the `/run/media` directory when using the graphical environment.
- The `lsblk` command lists the details of block devices, such as the size and the UUID.
- The `find` command searches in real time in the local file systems for files according to search criteria.

Chapter 14

Analyze Servers and Get Support

Goal

Investigate and resolve issues in the web-based management interface, getting support from Red Hat to help solve problems.

Objectives

- Activate the web console management interface to remotely manage and monitor the performance of a Red Hat Enterprise Linux server.
- Describe and use the 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

- Analyze and Manage Remote Servers (and Guided Exercise)
- Create a Diagnostics Report (and Guided Exercise)
- Detect and Resolve Issues with Red Hat Insights (and Quiz)

Analyze and Manage Remote Servers

Objectives

- Activate the web console management interface to remotely manage and monitor the performance of a Red Hat Enterprise Linux server.

Describe the Web Console

The *web console* is a web-based management interface for Red Hat Enterprise Linux. The interface is designed for managing and monitoring your servers, and is based on the open-source Cockpit service.

You can use the web console to monitor system logs and to view graphs of system performance. Additionally, you can use your web browser to change settings by using graphical tools in the web console interface, including a fully functional interactive terminal session.

Enable the Web Console

Starting from Red Hat Enterprise Linux 7, the web console is installed by default in all installation variants except in a minimal installation. You can use the following command to install the web console:

```
[root@host ~]# dnf install cockpit
```

Then, enable and start the `cockpit.socket` service, which runs a web server. This step is necessary if you need to connect to the system through the web interface.

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

If you are using a custom firewall profile, then you must add the `cockpit` service to `firewalld` to open port 9090 in the firewall:

```
[root@host ~]# firewall-cmd --add-service=cockpit --permanent
success
[root@host ~]# firewall-cmd --reload
success
```

Log in to the Web Console

The web console provides its own web server. Launch your web browser to log in to the web console.

Open `https://servername:9090` in your web browser, where *servername* is the hostname or IP address of your server. The web console protects the connection by a *Transport Layer Security* (TLS) session. By default, the `cockpit` service installs the web console with a self-signed TLS certificate. When you connect to the web console for the first time, the web browser probably

displays a security warning. The `cockpit -ws(8)` man page provides instructions on how to replace the TLS certificate with a correctly signed one.

To log in to the web console, enter your username and password at the login screen. You can log in with the username and password of any local account on the system, including the `root` user.

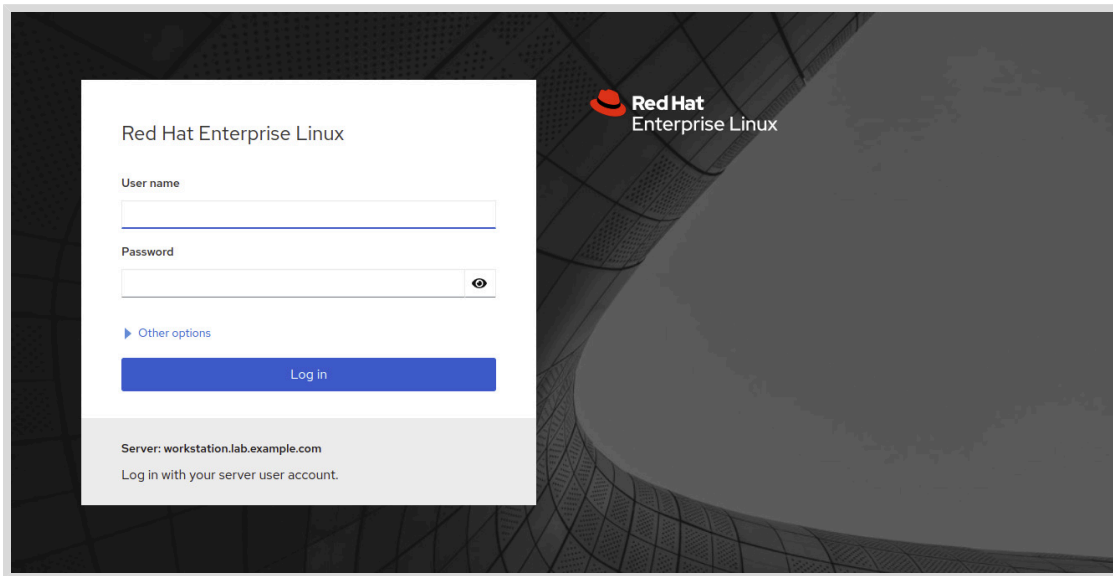


Figure 14.1: The web console login screen

Click **Log In**.

After you log in, the web console displays the username on the left side of the title bar. The default access to the web console is with limited rights, as you can see in the following **Limited access** button and in the "Web console is running in limited access mode" message.



Figure 14.2: Non-privileged user's title bar

If your account is configured with the appropriate privileges, then you can escalate privileges by switching to administrative access, by clicking the **Limited access** or **Turn on administrative access** buttons. During the escalation privileges process, you need to enter your password. When you have escalated privileges, the **Limited access** button changes to **Administrative access**.

You can switch back to limited access mode by clicking the **Administrative access** button and then clicking the **Limit access** button in the pop-up window that it shows.



Figure 14.3: Privileged user's title bar

Change Passwords in the Web Console

You can change your own password when logged in to the web console. Click the **Accounts** button on the navigation bar. Click your account label to open the account details page.

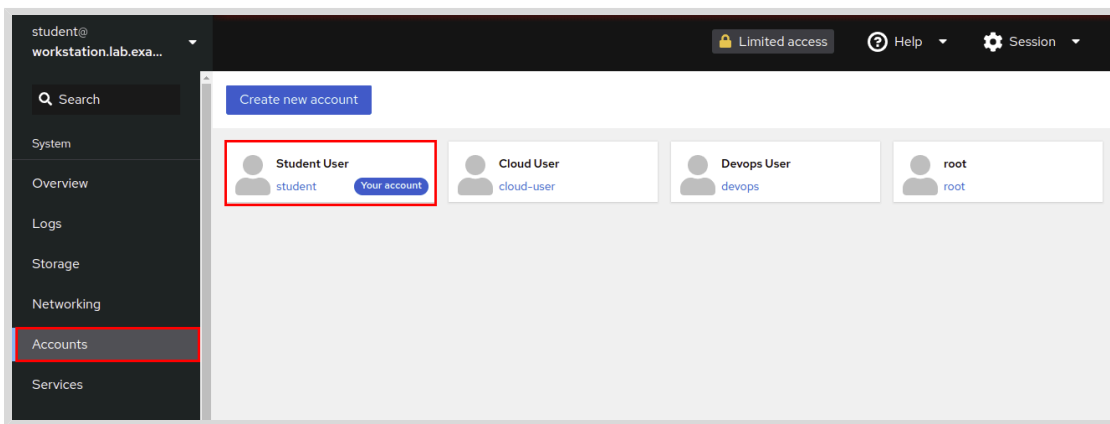


Figure 14.4: User accounts

As a non-privileged user, you are restricted to setting or resetting your password and managing public SSH keys. To set or reset your password, click the **Set password** button.

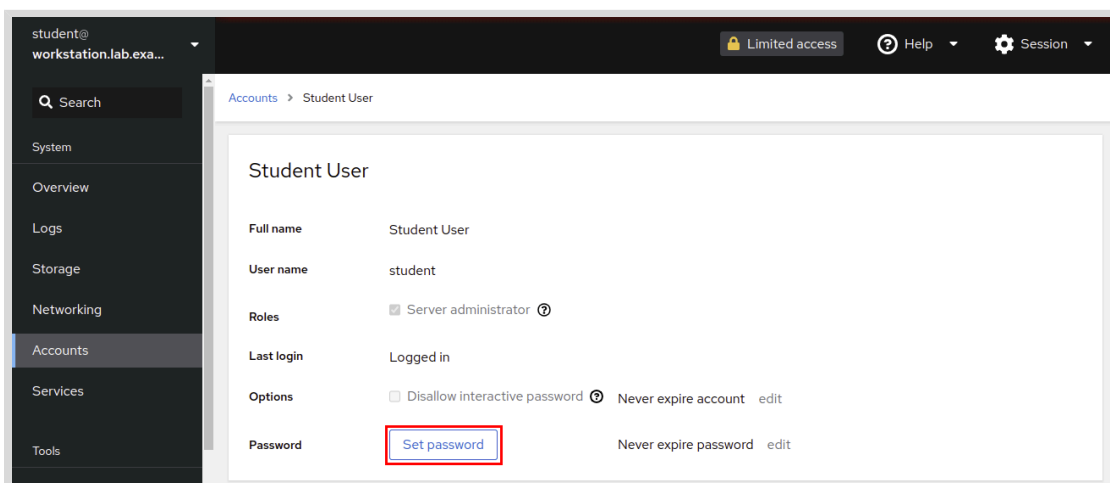


Figure 14.5: User account details

Enter your information in the **Old password**, **New password**, and **Confirm new password** fields. Click the **Set password** button to activate the new password.

 A screenshot of the 'Set password' form. It contains three input fields: 'Old password' (masked with dots), 'New password' (masked with dots, with a green progress bar below it indicating 'Excellent password'), and 'Confirm new password' (masked with dots). At the bottom are two buttons: 'Set password' (blue) and 'Cancel' (light blue).

Figure 14.6: Setting and resetting passwords

Troubleshoot with the Web Console

The web console is a powerful troubleshooting tool. You can monitor system statistics in real time, inspect system logs, and switch to a terminal session within the web console to gather additional information from the command-line interface.

Monitor System Statistics in Real Time

Click the **Overview** button on the navigation bar to view information about the system, such as its type of hardware, operating system, hostname, and more. If you log in as a non-privileged user, then you can see all the information but not modify any value. The following image displays the **Overview** page.

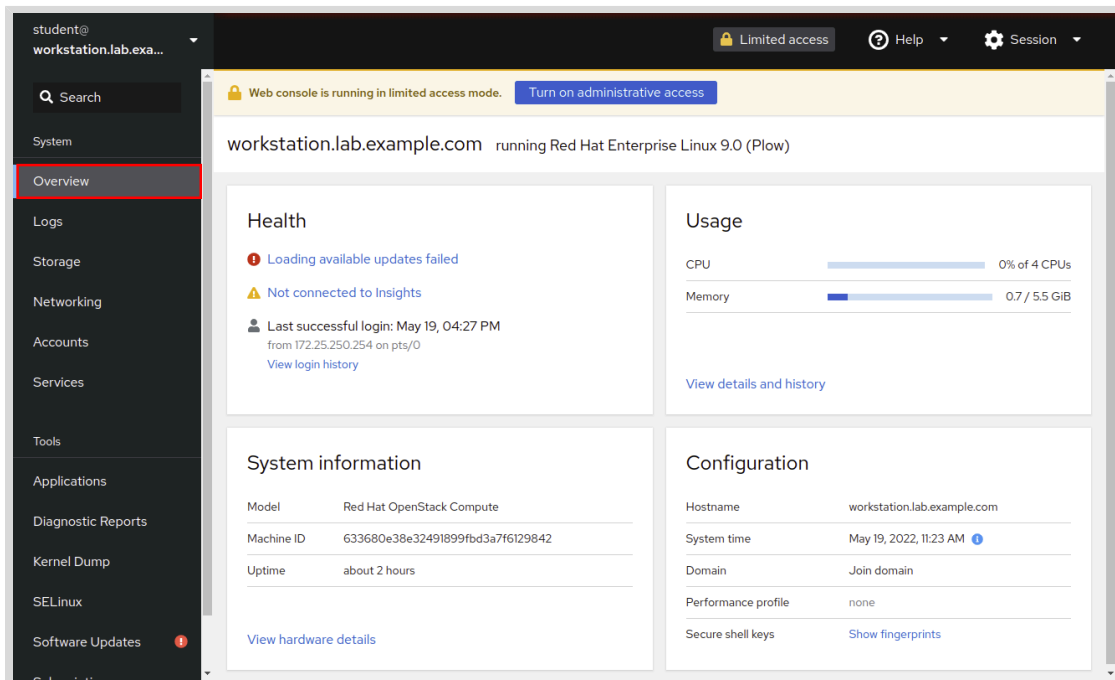


Figure 14.7: Non-privileged user's Overview page

Click **View details and history** on the Overview page to view details of current system performance for CPU activity, memory use, disk I/O, and network usage.

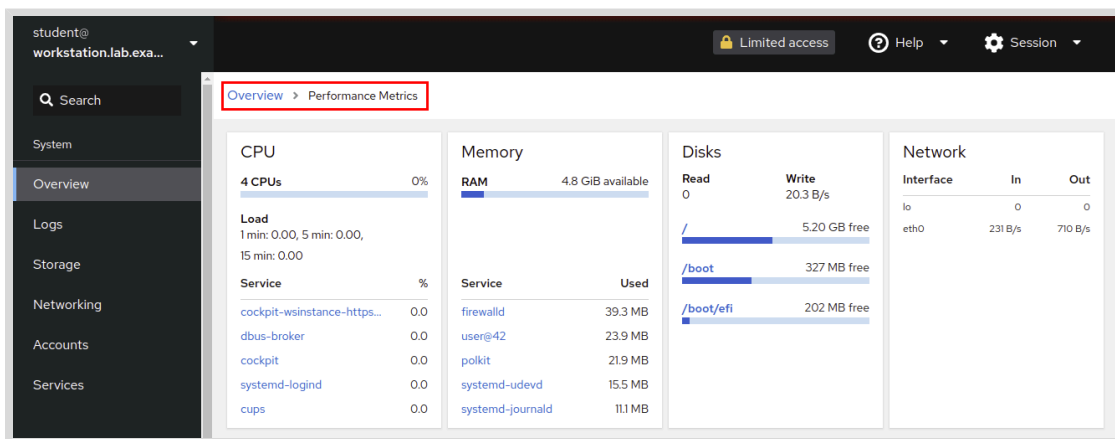


Figure 14.8: Non-privileged user's system performance metrics

Inspect and Filter Syslog Events

The **Logs** section in the navigation bar provides access to analysis tools for the system logs. You can use the scroll menus on the page to filter log messages by a logging date range, or priority, or both. The web console uses the current date as the default; you can click the date menu and specify any range of dates. Similarly, the **Priority** menu provides options that range from **Debug and above** (at the lowest level) to more specific severity conditions such as **Alert and above** or **Error and above**.

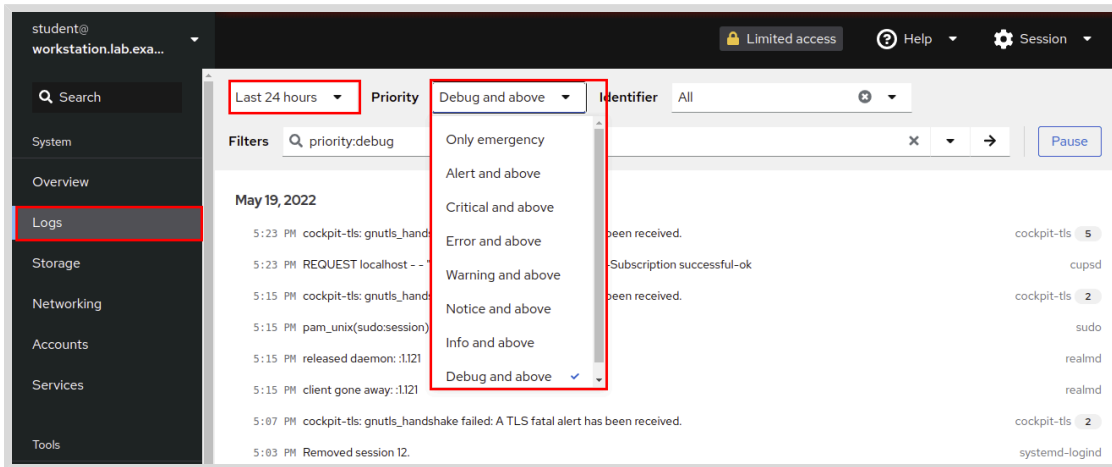


Figure 14.9: Log severity selections

Click a row to view details of the log report. In the following example, note the first row that reports on a **sudo** log message.

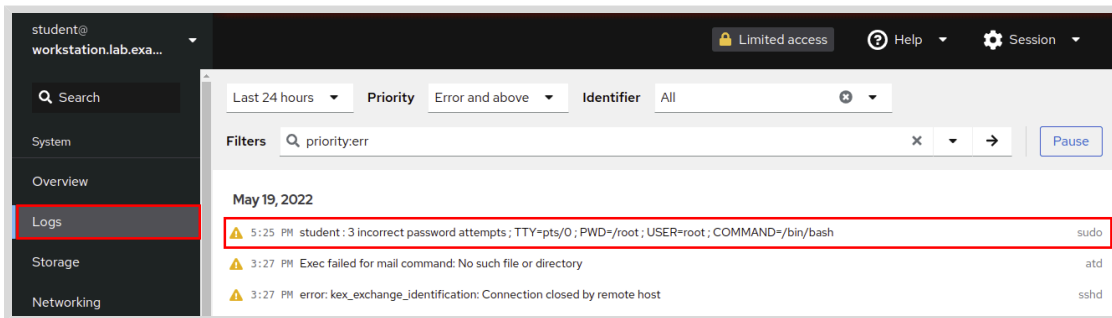


Figure 14.10: Log entry selection

The following example shows the details that the web console displays when you click the **sudo** row. Details of the report include the selected log entry (**sudo**), the date, time, priority, and syslog facility of the log entry, and the hostname of the system that reported the log message.

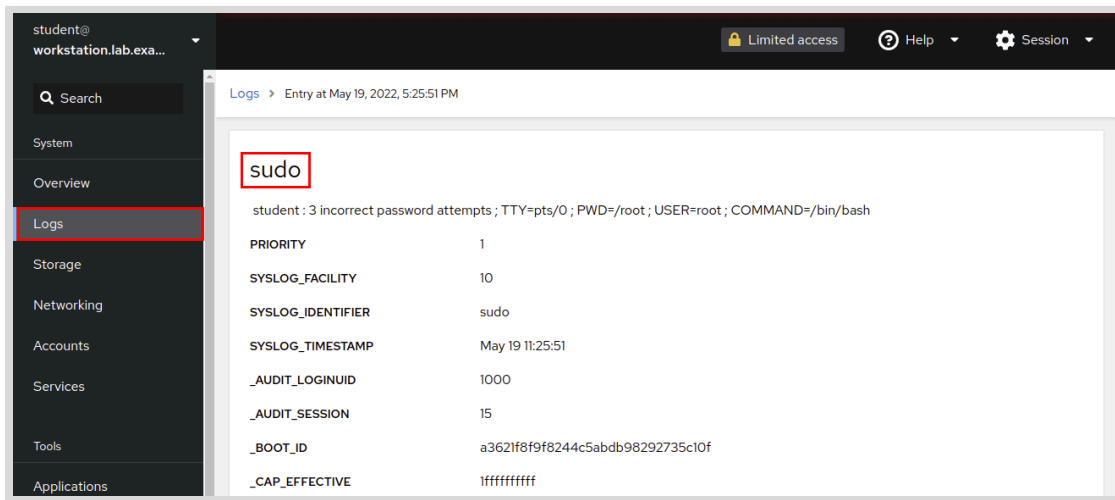


Figure 14.11: Log entry details

Run Commands from a Terminal Session

The **Terminal** button in the navigation bar provides access to a fully functional terminal session within the web console interface. In this web console terminal, you can run arbitrary commands to manage and work with the system, and for tasks that the other web console tools do not support.

The following image displays examples of commands to gather additional information. For example, listing the contents of the `/var/log` directory provides reminders of log files that might have valuable information. The `id` command provides information such as group membership that might help to troubleshoot file access restrictions. The `ps -au` command provides a view of processes that are running in the terminal and of the user that is associated with the process.

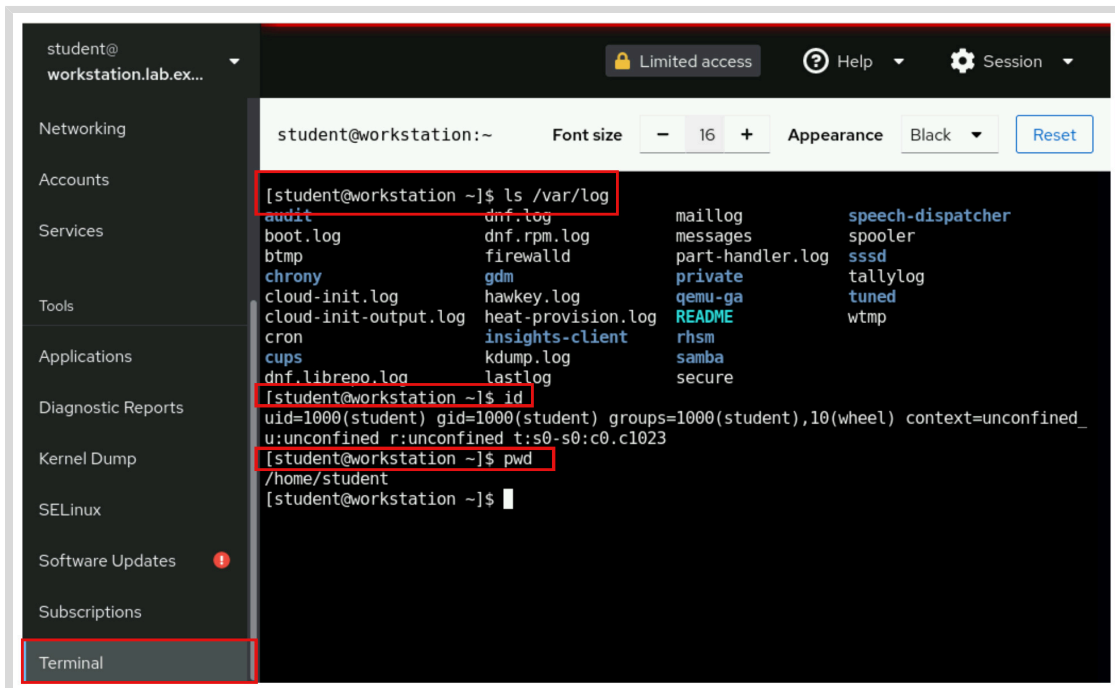


Figure 14.12: Non-privileged terminal session troubleshooting

Create Diagnostic Reports

A diagnostic report is a collection of configuration details, system information, and diagnostic information from a Red Hat Enterprise Linux system. Data that is collected in the report includes system logs and debug information that you can use to troubleshoot issues.

To generate a diagnostic report, log in to the web console as a privileged user. Click the **Diagnostic Reports** button on the navigation bar to open the page that creates these reports. Click the **Create report** button to generate a new diagnostic report.

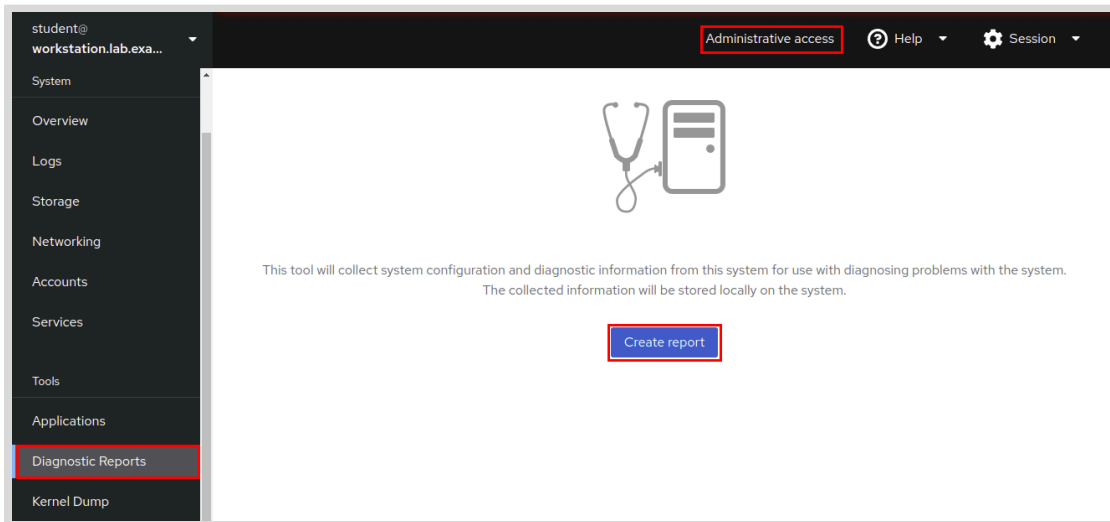


Figure 14.13: Create a diagnostic report

After some minutes, the interface displays **Done!** when the report is complete. Click the **Download report** button to save the report to your local system.

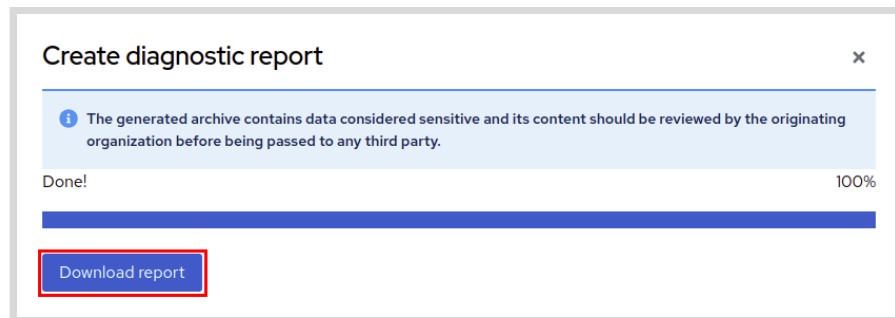


Figure 14.14: Download a completed report

Click **Save File** and complete the process.

Manage System Services with the Web Console

As a privileged web console user, you can stop, start, enable, and restart system services. Additionally, you can configure network interfaces, configure firewall services, administer user accounts, and more.

System Power Options

In the web console, you can restart or shut down the system. To access the system power options, log in to the web console as a privileged user. Click the **Overview** button on the navigation bar to access system power options.

From the menu on the upper right, select the appropriate option to either reboot or shut down a system.

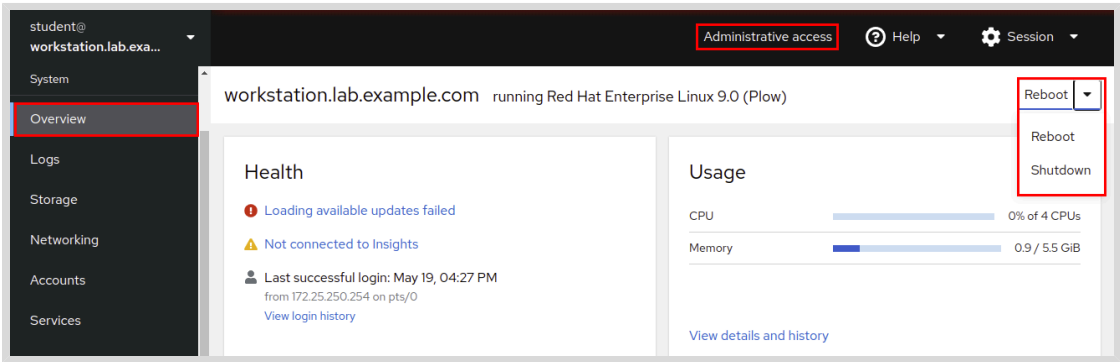


Figure 14.15: System power options

Control Running System Services

You can start, enable, disable, and stop services with graphical tools in the web console. To do so, click the **Services** button on the navigation bar to access the web console's services initial page. The **Services** page shows the system services tab by default. You can change to **Targets** or **Sockets** by clicking the appropriate tab. Use the search bar or scroll through the page to select the service to manage.

In the following example, select the `atd` service row to open the service management page.

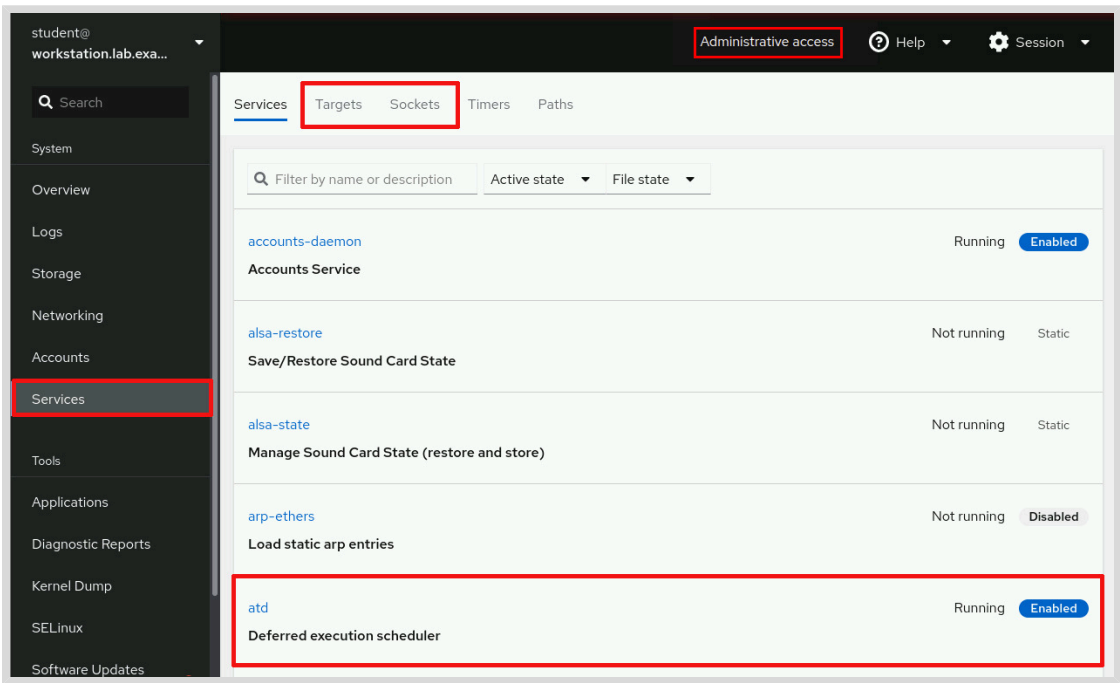


Figure 14.16: Services: Initial view

Click the **Stop**, **Restart**, or **Disallow running (mask)** buttons as appropriate to manage the service. In this view, the service is already running. To view additional information about the service, click any of the highlighted links or scroll through the service logs that are displayed below the service management section.

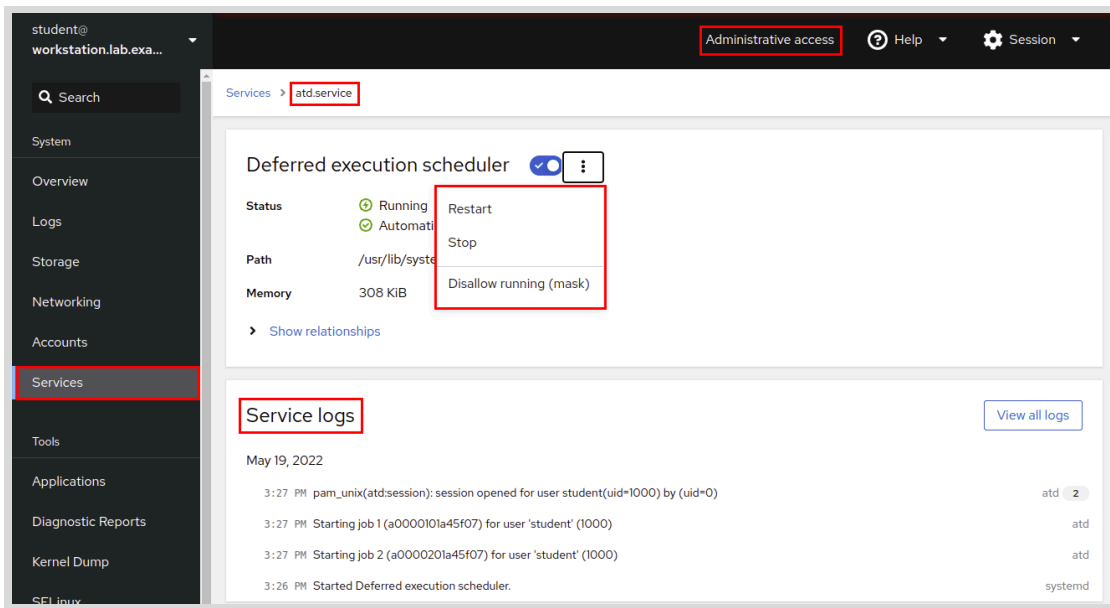


Figure 14.17: Services: Service details and management interface

Configure Network Interfaces and the Firewall

To manage firewall rules and network interfaces, click the **Networking** button on the navigation bar. The following example shows how to gather information about network interfaces and how to manage them.

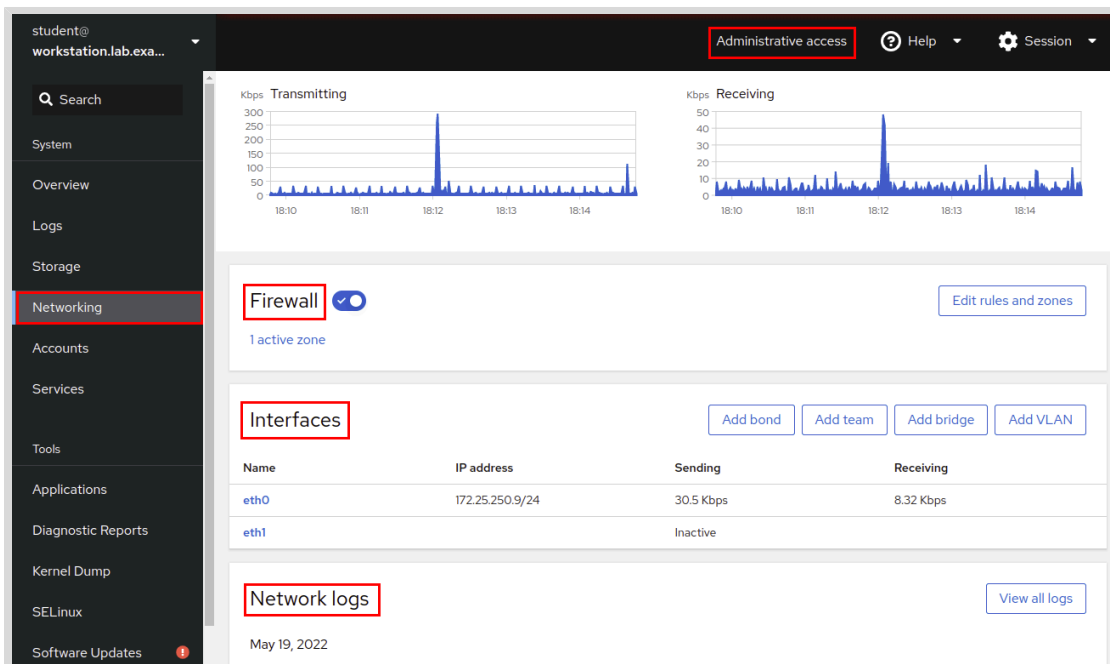


Figure 14.18: Networking: Initial view

Click the appropriate interface name in the **Interfaces** section to access the management page. In this example, the **eth0** interface is selected. The top part of the management page displays network traffic activity for the selected device. Scroll down to view configuration settings and management options.

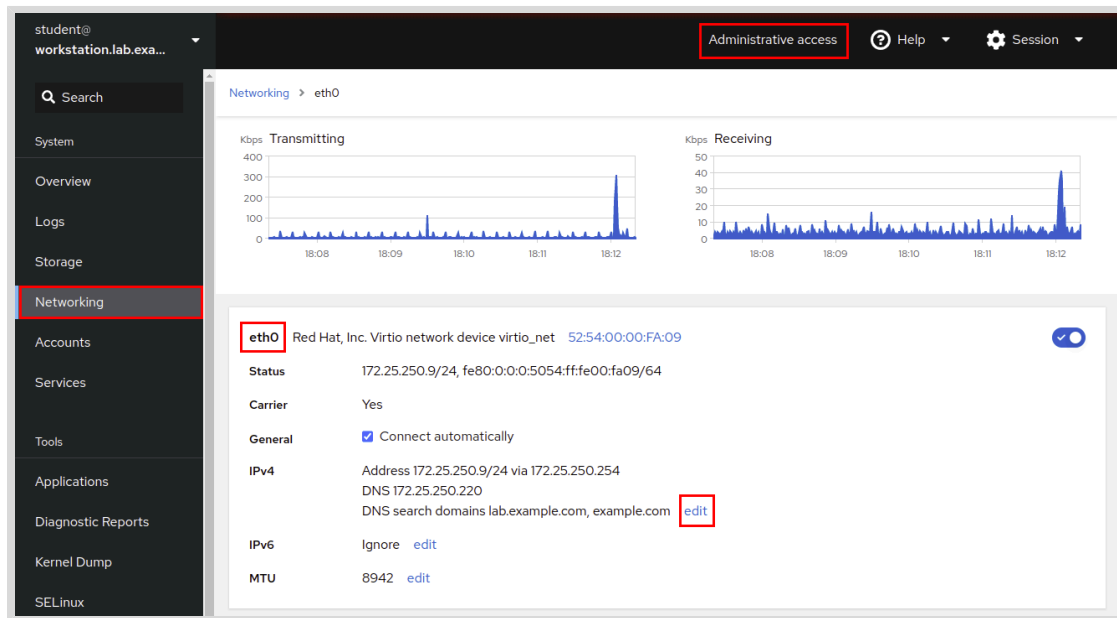


Figure 14.19: Networking: Interface details

To modify or add configuration options in an interface, click the highlighted links for the wanted configuration. In this example, the **IPv4** link shows a single IP address and netmask, 172.25.250.9/24 for the `eth0` network interface. To add an IP address to the `eth0` network interface, click the **edit** link.

Click the + symbol on the right side of the **Manual** list selection to add an IP address. Enter an IP address and network mask in the appropriate fields. Click **Apply** to activate the new settings.

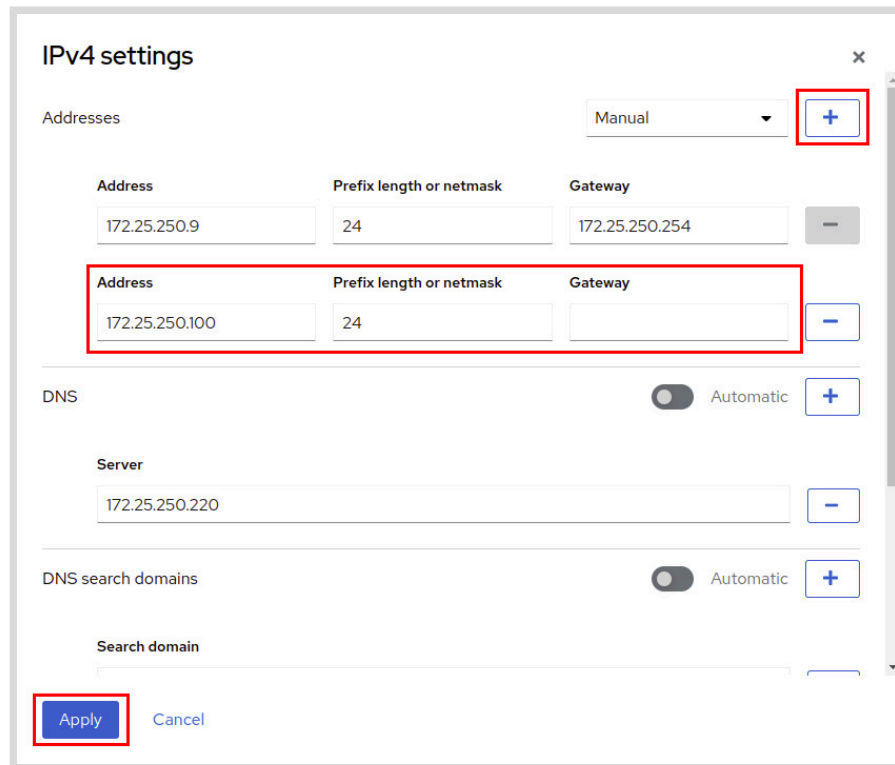


Figure 14.20: Add an IP address to an existing interface

The display automatically switches back to the interface's management page where you can confirm the new IP address.

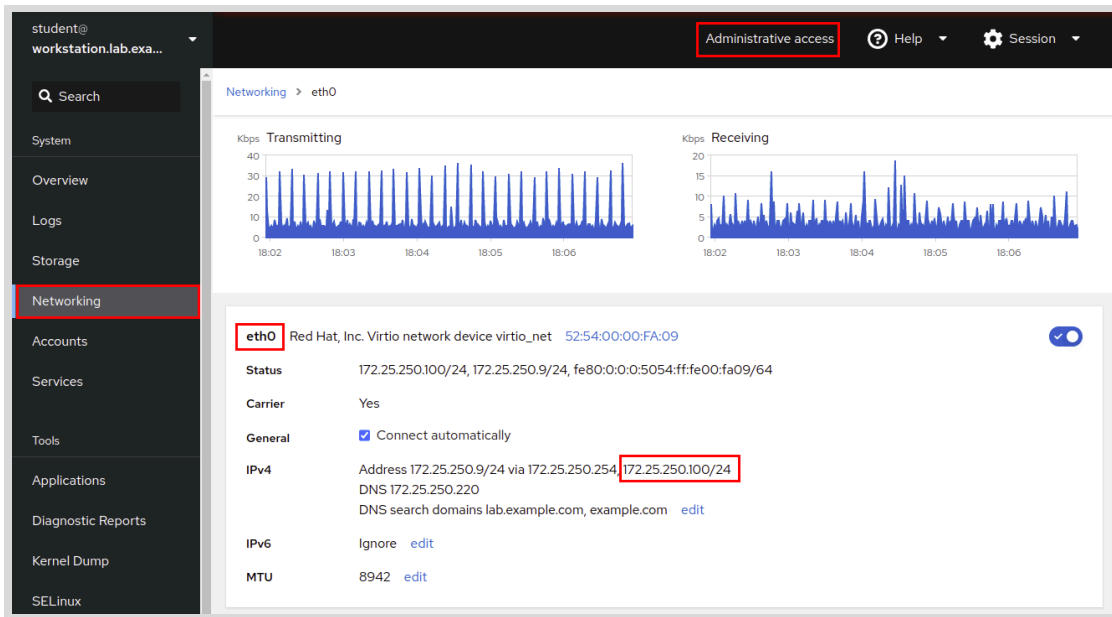


Figure 14.21: Confirm the new IP address

Administer User Accounts

As a privileged user, you can create user accounts in the web console. Click **Accounts** on the navigation bar to view existing accounts. Click **Create new account** to open the account management page.

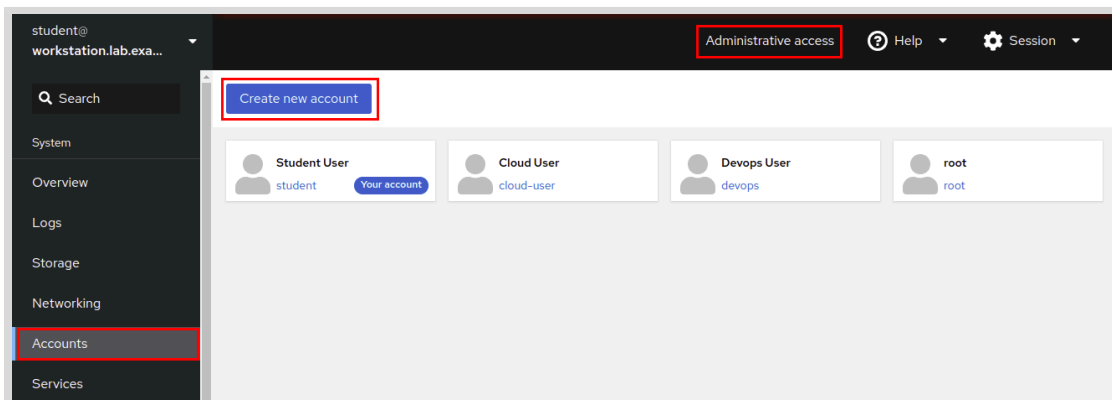


Figure 14.22: Existing user accounts

Enter the information for the new account and then click **Create**.

Figure 14.23: Create an account

The display automatically reverts to the account management page, where you can confirm the new user account.

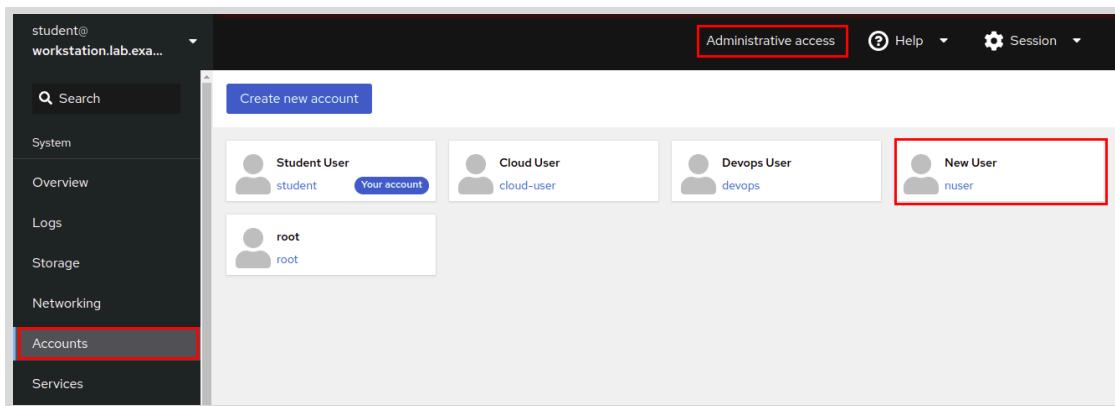


Figure 14.24: Account management page



References

cockpit(1), cockpit-ws(8), and cockpit.conf(5) man pages

For more information, refer to *Managing Systems Using the RHEL 9 Web Console* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_systems_using_the_rhel_9_web_console/index#getting-started-with-the-rhel-9-web-console_managing-systems-using-the-web-console

► Guided Exercise

Analyze and Manage Remote Servers

Enable and access the web console on a server to manage it and to diagnose and resolve issues.

Outcomes

- Use the web console to monitor system features, inspect log files, create user accounts, and access the terminal.

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-cockpit
```

Instructions

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

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

- 2. The web console is already installed on the system, but it is not active. Enable and start the `cockpit` service.

- 2.1. Enable the web console service.

```
[student@servera ~]$ sudo systemctl enable --now cockpit.socket
[sudo] password for student: student
Created symlink /etc/systemd/system/sockets.target.wants/cockpit.socket -> /usr/lib/systemd/system/cockpit.socket.
```

- 3. On the `workstation` machine, open the Firefox web browser and log in to the web console interface at `servera.lab.example.com`. Log in as the `student` user.

- 3.1. Open the browser and navigate to `https://servera.lab.example.com:9090`.

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

- 3.3. Log in as the `student` user, with `student` as the password.

You are now logged in to the web console as a normal user, with minimal privileges.

- 4. Verify your current authorization within the web console interface.

- 4.1. Click the **Terminal** button on the left navigation bar to access the terminal.
A terminal session opens where the **student** user is already logged in. Verify that command execution works in the embedded terminal.

```
[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
```

- 4.2. Click the **Accounts** button on the left navigation bar to manage users.
The **Create new account** button does not appear, because you are logged in with limited access.
- 4.3. Click the **Student User** link.
On the **student** user's account details page, you can only set a new password or add an authorized SSH public key.
- ▶ 5. Escalate privileges in the web console.
 - 5.1. Click the **Limited access** button to switch to administrative access. Use **student** as the **student** user password and click the **Authenticate** button. The web console replaces the **Limited access** button by the **Administrative access** button.
- ▶ 6. To investigate system statistics, click **Overview** on the left navigation bar and the **View metrics and history** button on the **Usage** section.
This page displays various operating system statistics, such as current load, disk usage, disk I/O, and network traffic.
- ▶ 7. To inspect system logs, click the **Logs** button on the left navigation bar.
This page displays the **systemd** system logs. Use the buttons in the upper-left corner to modify how the web console displays the log entries based on date and the priority of the logs.
 - 7.1. Click the **Priority** list and choose **Debug and above**.
 - 7.2. Based on the current day of the month, click any log entry from the list. A log entry detail page opens with additional information about the event, such as the hostname, the SELinux context, or the PID number of the process that the entry corresponds to.
- ▶ 8. Add a second IP address to an existing network interface device.
 - 8.1. Click the **Networking** button on the left navigation bar.
This page displays details of the current network configuration for **servera**, as well as real-time network statistics, firewall configuration, and log entries about networking.
 - 8.2. Scroll down to the **Interfaces** section and click the row for the **eth0** network interface.
A details page displays real-time network statistics, as well as the current configuration for that network interface.
 - 8.3. Click the **edit** link in the **IPv4** section.
An **IPv4 settings** window opens, where you can change the network interface configuration.

- 8.4. In the **IPv4 settings** window, click the + button next to the **Manual** list.
- 8.5. In the **Address** text box, enter **172.25.250.99** as the second IP address.
- 8.6. In the **Prefix length or Netmask** text box, enter **24** as the netmask value.
- 8.7. Click **Save** to apply new network configuration.
The new configuration is applied immediately in the web console. The new IP address is visible in the **IPv4** line.

- ▶ 9. Create a user account.
 - 9.1. Click the **Accounts** button on the left navigation bar. The web console now shows the **Create new account** button, because you have administrative rights.
 - 9.2. Click the **Create new account** button.
 - 9.3. In the **Create new account** window, add the following details:

Field	Value
Full Name	manager1
User Name	manager1
Password	redh@t!23
Confirm	redh@t!23

- 9.4. Click **Create**.

- ▶ 10. Access a terminal session within the web console to add the **manager1** user to the **wheel** group.
 - 10.1. Click the **Terminal** button on the left navigation bar.
 - 10.2. Use the `id manager1` command to view the group membership of the **manager1** user.

```
[student@servera ~]$ id manager1
uid=1002(manager1) gid=1002(manager1) groups=1002(manager1)
```

- 10.3. Use the `sudo usermod -aG wheel manager1` command to add the **manager1** user to the **wheel** group.

```
[student@servera ~]$ sudo usermod -aG wheel manager1
[sudo] password for student: student
```

- 10.4. Use the `id manager1` command again to verify that the **manager1** user is a member of the **wheel** group.

```
[student@servera ~]$ id manager1
uid=1002(manager1) gid=1002(manager1) groups=1002(manager1),10(wheel)
```


- ▶ **11.** Enable and start the Kernel process accounting service (`psacct`).
 - 11.1. Click the **Services** button on the left navigation bar.
 - 11.2. Search for the `psacct` service. Click the service link. A details page displays the service status as disabled.
 - 11.3. Click the **Start and Enable** button next to the service name.
 - 11.4. The service is now enabled and started.
- ▶ **12.** Log off from the web console interface.
- ▶ **13.** Return to the `workstation` system as the `student` user.

```
[student@servera ~]$ exit  
[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 support-cockpit
```

Create a Diagnostics Report

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 enables 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 publicly accessible, whereas 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 to register systems and to ensure 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 to request evaluation for the products with no entitlements.

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.

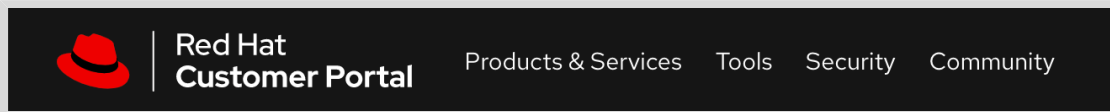


Figure 14.25: Red Hat Customer Portal Menus

The **Products & Services** menu gives access to the Product Hubs, for 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 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

You can generate a diagnostics report to submit to Red Hat technical support by using either the web console or the command line.

Generate an sos Report with the Web Console

Use the following steps to generate a diagnostic report from the web console:

1. To generate an `sos` report from the web console, you must log in as a privileged user.
2. Click **Diagnostic Reports**, and then click **Run report** and provide a label for the report in the `Report label` field.
3. Click **Run report** again to start generating the report. The diagnostic report takes a few minutes to build.
4. Click **Download** to download the diagnostic report.

Generate an sos Report from the Command Line

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 --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 an sos Report to Red Hat Technical Support

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

- Send an sos report by using the `sos report --upload` option.
- Send an 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.

For more information, visit <https://developers.redhat.com/>.



References

`sosreport(1)` man page

Contacting Red Hat Technical Support

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

► Guided Exercise

Create a Diagnostics Report

Generate a diagnostics report that you could submit to the Red Hat Customer Portal as part of a support case, by using the web console.

Outcomes

- Generate a diagnostics report that you could submit as part of a support case to the Red Hat Customer Portal.

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 ====
```

- 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 `student` user with `student` 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 `student` user with `student` as the password.
 - 5.4. Click the **Limited access** button to switch to administrative access. Use `student` as the `student` user's password and click the **Authenticate** button. The web console replaces the **Limited access** button with the **Administrative access** button.
 - 5.5. Click the **Diagnostic Reports** menu on the left navigation bar. Click the **Run report** button.
 - 5.6. Enter `student` in the `Report label` field.
 - 5.7. Click **Run report**. The report takes a few minutes to build.
- 6. When the report is ready, download the file.
- 6.1. Click the **Download** button. The diagnostic report is downloaded.
 - 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

```


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 to 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 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 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.

The Red Hat Insights platform analyzes the received data, and displays the result on the <https://console.redhat.com/insights> site.

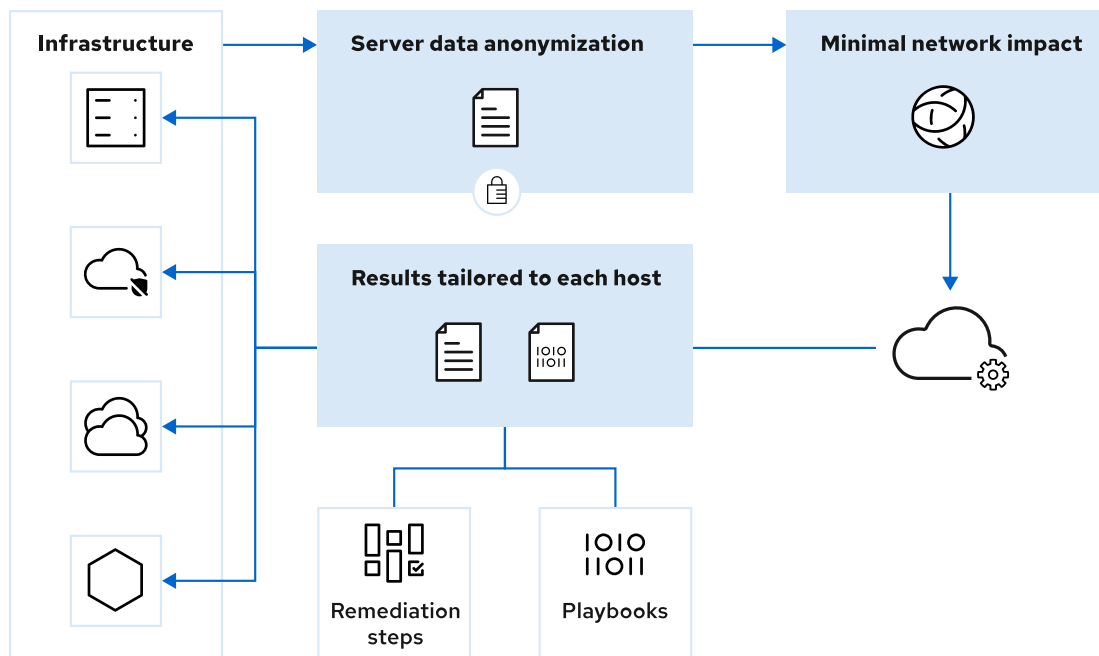


Figure 14.26: Insights high-level architecture

Register a RHEL System with Red Hat Insights

Use the `rhc connect` command to register your system to Red Hat and enable Red Hat Insights with a single command.

```
[root@host ~]# rhc connect -a host_key -o 117018
Connecting host.lab.example.com to Red Hat.
This might take a few seconds.
```

- Connected to Red Hat Subscription Management
- Connected to Red Hat Insights
- Activated the Red Hat connector daemon

Manage your Red Hat connector systems: <https://red.ht/connector>

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 required 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.

The Insights client periodically updates the metadata that is provided to Insights.

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

```
[root@host ~]# insights-client --register
This host has already been registered.
Automatic scheduling for Insights has been enabled.
Starting to collect Insights data for host.lab.example.com
...output omitted...

Writing RHSM facts to /etc/rhsm/facts/insights-client.facts ...
Uploading Insights data.
Successfully uploaded report from host.lab.example.com to account 60766.
View the Red Hat Insights console at https://console.redhat.com/insights/
View details about this system on console.redhat.com:
https://console.redhat.com/insights/inventory/a47b1970-750b-4ff2-b79e-825efea65
```

Use the `insights-client` command to refresh the client's metadata.

```
[root@host ~]# insights-client
Starting to collect Insights data for host.lab.example.com
...output omitted...
Writing RHSM facts to /etc/rhsm/facts/insights-client.facts ...
Uploading Insights data.
Successfully uploaded report from host.lab.example.com to account 60766.
View details about this system on console.redhat.com:
https://console.redhat.com/insights/inventory/a47b1970-750b-4ff2-b79e-825efea65e
```

On Red Hat Insights (<https://console.redhat.com/insights>), ensure that you are logged in and that the system is visible under the **Inventory** section of the web UI.

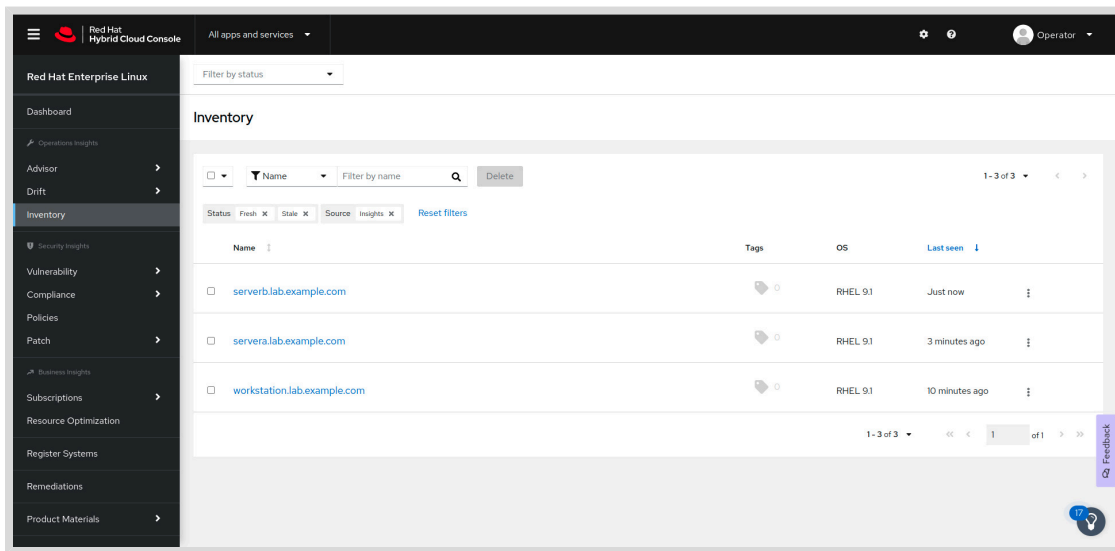


Figure 14.27: Insights inventory on the Cloud Portal

Red Hat Insights Console Navigation

Red Hat Insights provides a family of services that you access with a web browser at the <https://console.redhat.com/insights> website.

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.

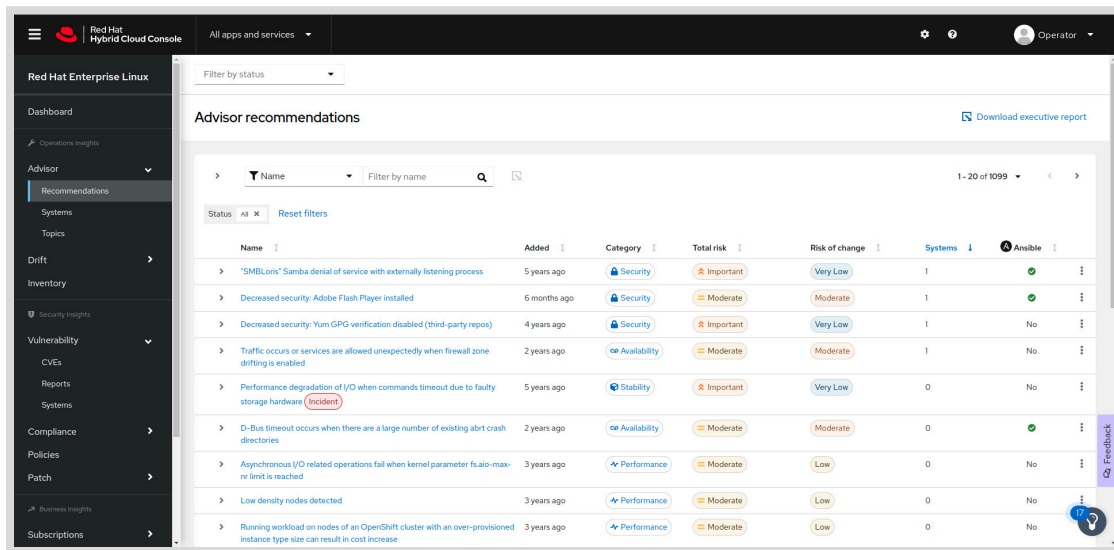


Figure 14.28: Recommendations from the Advisor Service

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.

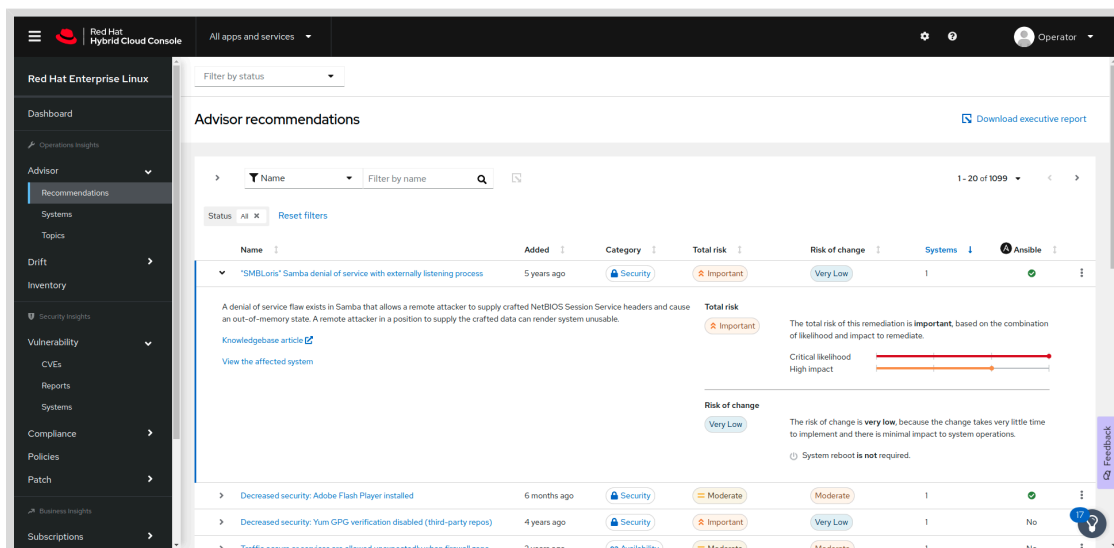


Figure 14.29: Details of an issue

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.

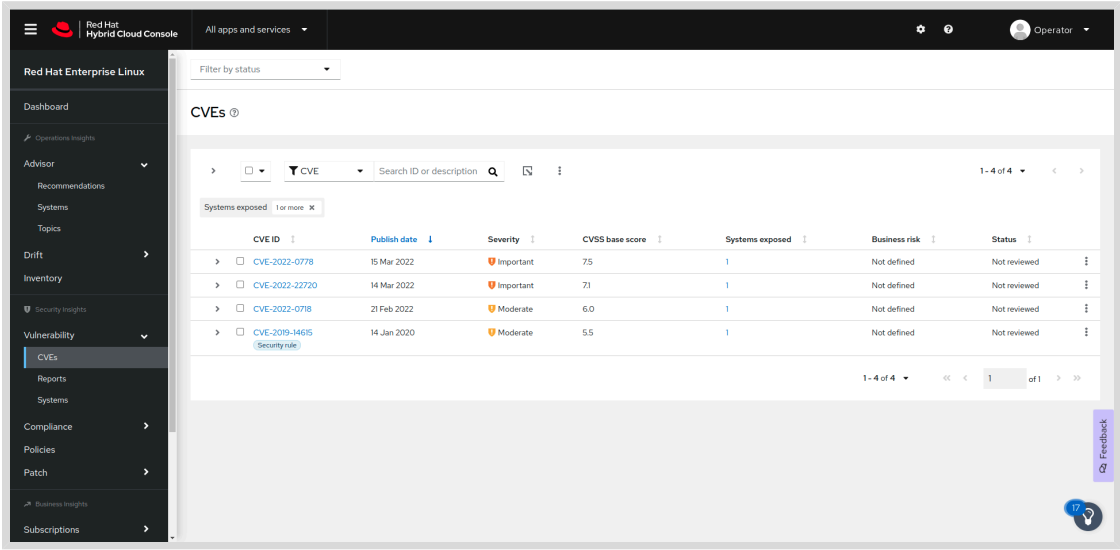


Figure 14.30: Report from the Vulnerability service

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.

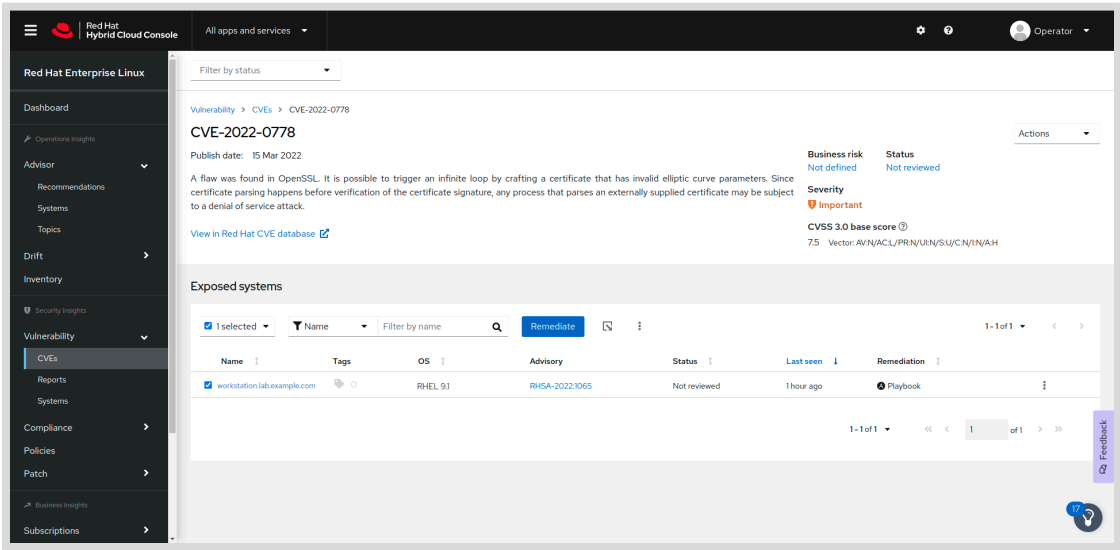


Figure 14.31: Details of a CVE

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 verify 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.

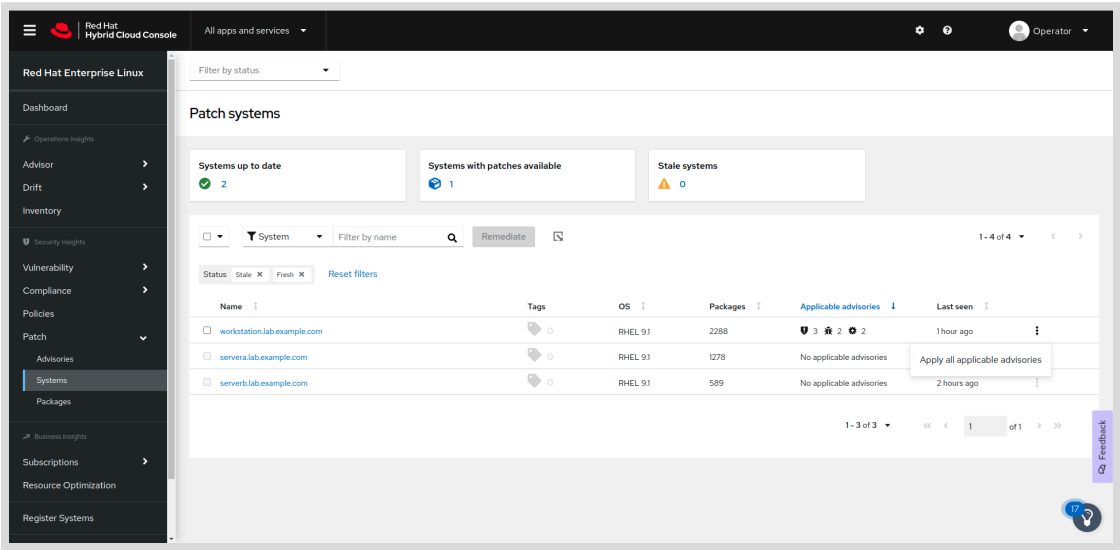


Figure 14.32: Patching a system

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:

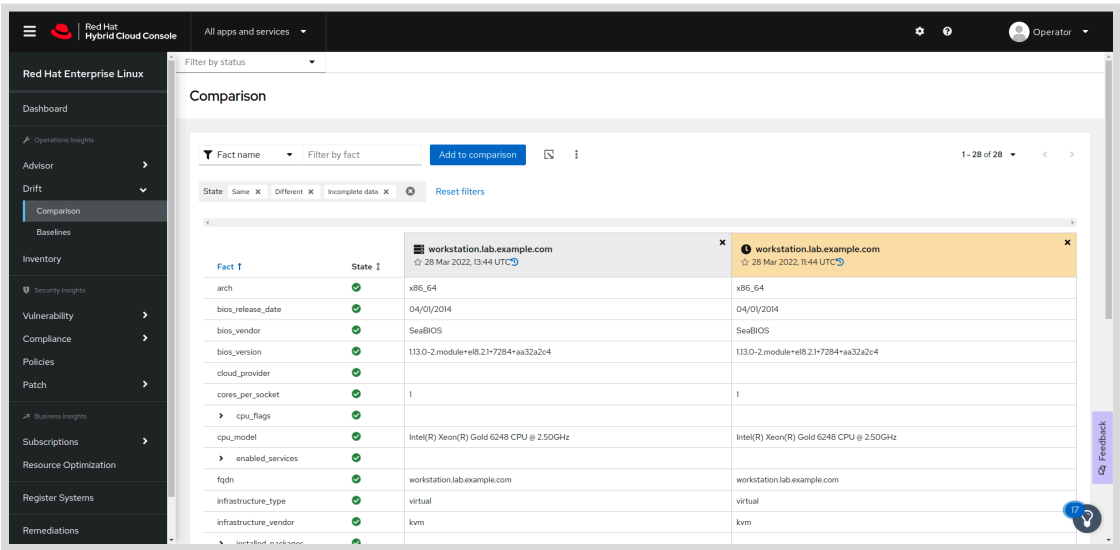


Figure 14.33: Comparing system history

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.

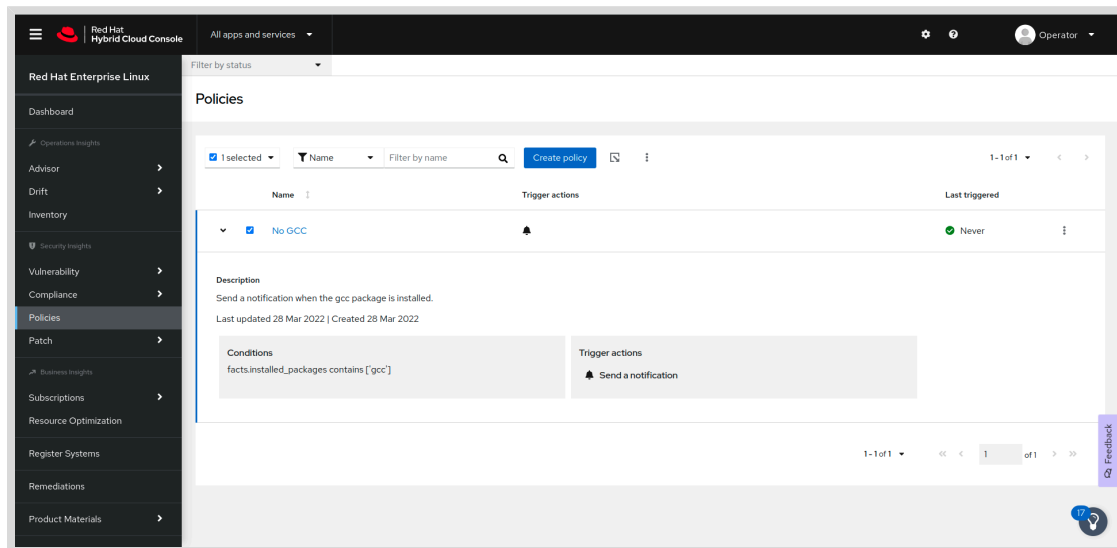


Figure 14.34: Details of a custom rule

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

For more information about excluding data that Insights collects, 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

Information about the data that Red Hat Insights collects 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

- The web console is a web-based management interface to your server, and is based on the open source `cockpit` service.
- The web console provides graphs of system performance, graphical tools to manage system configuration and to inspect logs, and interactive terminal interfaces.
- 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.
- The `redhat-support-tool` command-line tool queries Knowledgebase and works with support cases.
- 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 15

Comprehensive Review

Goal

Review tasks from *Red Hat System Administration I*.

Sections

- Comprehensive Review

Lab

- Manage Files from the Command Line
- Manage Users and Groups, Permissions, and Processes
- Configure and Manage a Server
- Manage Networks
- Mount File Systems and Find Files

Comprehensive Review

Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills that you learned in *Red Hat System Administration I*.

Reviewing Red Hat System Administration I

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter. Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

Chapter 1, Get Started with Red Hat Enterprise Linux

Define open source, Linux, Linux distributions, and Red Hat Enterprise Linux.

- Explain the purpose of open source, Linux, Linux distributions, and Red Hat Enterprise Linux.

Chapter 2, Access the Command Line

Log in to a Linux system and run simple commands from the shell.

- Log in to a Linux system and run simple commands from the shell.
- Log in to the Linux system with the GNOME desktop environment to run commands from a shell prompt in a terminal program.
- Save time when running commands from a shell prompt with Bash shortcuts.

Chapter 3, 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.
- Specify the absolute location and relative location of files to the current working directory, determine and change the working directory, and list the contents of directories.
- Create, copy, move, and remove files and directories.
- Create multiple file name references to 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 4, Get Help in Red Hat Enterprise Linux

Resolve problems by using local help systems.

- Find information in local Linux system manual pages.

Chapter 5, Create, View, and Edit Text Files

Create, view, and edit text files from command output or in a text editor.

- Save output or errors to a file with shell redirection, and process command output through multiple command-line programs with pipes.
- Create and edit text files from the command line with the `vim` editor.
- Set shell variables to run commands, and edit Bash startup scripts to set shell and environment variables to modify the behavior of the shell and programs that are run from the shell.

Chapter 6, 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, manage, 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 7, Control Access to Files

Set Linux file-system permissions on files and interpret the security effects of different permission settings.

- List file-system permissions on files and directories, and interpret the effects of those permissions on access by users and groups.
- Change the permissions and ownership of files with command-line tools.
- Control the default permissions of user-created files, explain the effects of special permissions, and use special and default permissions to set the group owner of files that are created in a directory.

Chapter 8, Monitor and Manage Linux Processes

Evaluate and control processes that run on a Red Hat Enterprise Linux system.

- Determine status, resource use, and ownership of running programs on a system, to control them.
- Use Bash job control to manage multiple processes that were started from the same terminal session.
- 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.

Chapter 9, Control Services and Daemons

Control and monitor network services and system daemons with the `systemd` service.

- List system daemons and network services that the `systemd` service and socket units started.
- Control system daemons and network services with the `systemctl` command.

Chapter 10, Configure and Secure SSH

Configure secure command-line service on remote systems with OpenSSH.

- Log in to a remote system and run commands with `ssh`.
- Configure a user account to use key-based authentication to log in to remote systems securely without a password.
- Disable direct logins as `root` and password-based authentication for the OpenSSH service.

Chapter 11, Manage Networking

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

- Describe fundamental concepts of network addressing and routing for a server.
- Test and inspect the current network configuration with command-line utilities.
- Manage network settings and devices with the `nmtui` command.
- Modify network configuration by editing configuration files.
- Configure a server's static hostname and its name resolution and test the results.

Chapter 12, Install and Update Software Packages

Download, install, update, and manage software packages from Red Hat and DNF package repositories.

- Register a system by using your Red Hat account and assign it entitlements for software updates and support services provided by Red Hat.
- Explain how software is provided as RPM packages, and investigate the DNF and RPM installed system packages.
- Find, install, and update software packages with the `dnf` command.
- Enable and disable server use of Red Hat or third-party DNF repositories.

Chapter 13, Access Linux File Systems

Access, inspect, and use existing file systems on storage that is attached to a Linux server.

- Identify a directory in the file-system hierarchy and the device where it is stored.
- Access the contents of file systems by adding and removing file systems in the file-system hierarchy.
- Search for files on mounted file systems with the `find` and `locate` commands.

Chapter 14, Analyze Servers and Get Support

Investigate and resolve issues in the web-based management interface, getting support from Red Hat to help solve problems.

- Activate the web console management interface to remotely manage and monitor the performance of a Red Hat Enterprise Linux server.
- Describe and use the 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.

► Lab

Manage Files from the Command Line



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.

Manage files, redirect a specific set of lines from a text file to another file, and edit the text files.

Outcomes

- Manage files from the command line.
- Display a specific number of lines from text files and redirect the output to another file.
- Edit text 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-rh124-review1
```

Specifications

- Log in to **serverb** as the **student** user.
- Create the **/home/student/grading** directory.
- Create three empty files called **grade1**, **grade2**, and **grade3**, in the **/home/student/grading** directory.
- Capture the first five lines of the **/home/student/bin/manage** file in the **/home/student/grading/review.txt** file.
- Append the last three lines of the **/home/student/bin/manage** file to the **/home/student/grading/review.txt** file. Do not overwrite any existing text in the **/home/student/grading/review.txt** file.

- Copy the `/home/student/grading/review.txt` file to the `/home/student/grading/review-copy.txt` file.
- Edit the `/home/student/grading/review-copy.txt` file so that the `Test JJ` line appears twice.
- Edit the `/home/student/grading/review-copy.txt` file to remove the `Test HH` line.
- Edit the `/home/student/grading/review-copy.txt` file so that a line with `A new line` exists between the `Test BB` line and the `Test CC` line.
- Create the `/home/student/hardcopy` hard link to the `/home/student/grading/grade1` file. You must create the hard link after completing the earlier step to create the `/home/student/grading/grade1` file.
- Create the `/home/student/softcopy` symbolic link to the `/home/student/grading/grade2` file.
- Save the output of a command that lists the contents of the `/boot` directory to the `/home/student/grading/longlisting.txt` file. The output should be a "long listing" that includes file permissions, owner and group owner, size, and modification date of each file. The output should omit hidden files.

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-rh124-review1
```

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-rh124-review1
```

► Solution

Manage Files from the Command Line



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.

Manage files, redirect a specific set of lines from a text file to another file, and edit the text files.

Outcomes

- Manage files from the command line.
- Display a specific number of lines from text files and redirect the output to another file.
- Edit text 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-rh124-review1
```

1. Log in to `serverb` as the `student` user.

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

2. Create the `/home/student/grading` directory. If the `/home/student` directory is your current directory, then you do not need to specify the absolute path to the `grading` directory when creating it.

```
[student@serverb ~]$ mkdir grading
```

3. In the `/home/student/grading` directory, create three empty files called `grade1`, `grade2`, and `grade3`.

- 3.1. Create the empty files called `grade1`, `grade2`, and `grade3` in the `/home/student/grading` directory. Apply the brace expansion shell feature to create all three files with a single `touch` command.

```
[student@serverb ~]$ touch grading/grade{1,2,3}
```

- 3.2. Verify that the `grade1`, `grade2`, and `grade3` files exist in the `/home/student/grading` directory.

```
[student@serverb ~]$ ls grading/  
grade1 grade2 grade3
```

4. Copy the first five lines of the `/home/student/bin/manage` file to the `/home/student/grading/review.txt` file.

- 4.1. View the first five lines of the `/home/student/bin/manage` file and redirect the output to the `/home/student/grading/review.txt` file. Use the single redirection symbol (`>`) to overwrite any existing content in the file.

```
[student@serverb ~]$ head -5 bin/manage > grading/review.txt
```

- 4.2. Verify that the `/home/student/grading/review.txt` file contains the following text:

```
Test AA  
Test BB  
Test CC  
Test DD  
Test EE
```

5. Append the last three lines of the `/home/student/bin/manage` file to the `/home/student/grading/review.txt` file. Use the double redirection symbol (`>>`) to append the output and preserve the contents of the file.

- 5.1. View the last three lines of the `/home/student/bin/manage` file and append the output to the `/home/student/grading/review.txt` file.

```
[student@serverb ~]$ tail -3 bin/manage >> grading/review.txt
```

- 5.2. Verify that the `/home/student/grading/review.txt` file contains the following text:

```
Test AA  
Test BB  
Test CC  
Test DD  
Test EE  
Test HH  
Test II  
Test JJ
```

6. Copy the `/home/student/grading/review.txt` file to the `/home/student/grading/review-copy.txt` file.

- 6.1. Navigate to the `/home/student/grading` directory.

```
[student@serverb ~]$ cd grading/
[student@serverb grading]$
```

- 6.2. Copy the `/home/student/grading/review.txt` file to the `/home/student/grading/review-copy.txt` file.

```
[student@serverb grading]$ cp review.txt review-copy.txt
```

- 6.3. Navigate back to the home directory of the `student` user.

```
[student@serverb grading]$ cd
[student@serverb ~]$
```

7. Edit the `/home/student/grading/review-copy.txt` file to have two sequential `Test JJ` lines.

- 7.1. Use the `vim` text editor to open the `/home/student/grading/review-copy.txt` file.

```
[student@serverb ~]$ vim grading/review-copy.txt
```

- 7.2. From the command mode in `vim`, scroll down to the `Test JJ` line. Press the `y` key twice to copy the line of text, and press the `p` key to paste it below the cursor. Type `wq` to save the changes and quit `vim`. Verify that the `/home/student/grading/review-copy.txt` file contains the following text:

```
Test AA
Test BB
Test CC
Test DD
Test EE
Test HH
Test II
Test JJ
Test JJ
```

8. Edit the `/home/student/grading/review-copy.txt` file to remove the `Test HH` line.

- 8.1. Use the `Vim` text editor to open the `/home/student/grading/review-copy.txt` file.

```
[student@serverb ~]$ vim grading/review-copy.txt
```

- 8.2. From the command mode in `Vim`, scroll down to the `Test HH` line. Press the `d` key twice on your keyboard to delete the line of text. Type `wq` to save the changes and quit `vim`. Verify that the `/home/student/grading/review-copy.txt` file contains the following text:

```
Test AA
Test BB
Test CC
Test DD
Test EE
Test II
Test JJ
Test JJ
```

9. Edit the `/home/student/grading/review-copy.txt` file so that the line with `A new line` exists between the `Test BB` line and the `Test CC` line.

- 9.1. Use the Vim text editor to open the `/home/student/grading/review-copy.txt` file.

```
[student@serverb ~]$ vim grading/review-copy.txt
```

- 9.2. From the command mode in Vim, scroll down to the `Test CC` line. Press the `i` key to switch to the insert mode while keeping the cursor at the beginning of the `Test CC` line. From the insert mode, press `Enter` to create a blank line above the cursor. Use the up arrow to navigate to the blank line and create the line of `A new line` text. Press the `Esc` key to switch back to the command mode. Type `wq` to save the changes and to quit Vim. Verify that the `/home/student/grading/review-copy.txt` file contains the following text.

```
Test AA
Test BB
A new line
Test CC
Test DD
Test EE
Test II
Test JJ
Test JJ
```

10. Create the `/home/student/hardcopy` hard link to the `/home/student/grading/grade1` file.

- 10.1. Create the `/home/student/hardcopy` hard link to the `/home/student/grading/grade1` file.

```
[student@serverb ~]$ ln grading/grade1 hardcopy
```

- 10.2. View the link count of the `/home/student/grading/grade1` file.

```
[student@serverb ~]$ ls -l grading/grade1
-rw-r--r--. 2 student student 0 Mar  6 16:45 grading/grade1
```

11. Create the `/home/student/softcopy` symbolic link to the `/home/student/grading/grade2` file.

- 11.1. Create the `/home/student/softcopy` symbolic link to the `/home/student/grading/grade2` file.

```
[student@serverb ~]$ ln -s grading/grade2 softcopy
```

- 11.2. View the properties of the `/home/student/softcopy` symbolic link.

```
[student@serverb ~]$ ls -l softcopy
lrwxrwxrwx. 1 student student 14 Mar  6 17:58 softcopy -> grading/grade2
```

12. List the contents of the `/boot` directory and redirect the output to the `/home/student/grading/longlisting.txt` file. The output should be a long listing that includes the file permissions, owner and group owner, size, and modification date of each file. The output should omit hidden files.

- 12.1. View the contents of the `/boot` directory in the long listing format, and omit hidden files. Redirect the output to the `/home/student/grading/longlisting.txt` file.

```
[student@serverb ~]$ ls -l /boot > grading/longlisting.txt
```

- 12.2. Return to the `workstation` system 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-rh124-review1
```

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-rh124-review1
```

► Lab

Manage Users and Groups, Permissions, and Processes



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.

Manage user and group accounts, set permissions on files and directories, and manage processes.

Outcomes

- Manage user accounts and groups.
- Set permissions on files and directories.
- Identify and manage high CPU-consuming processes.

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-rh124-review2
```

Specifications

- Log in to `serverb` as the `student` user.
- Identify and terminate the process that currently uses the most CPU time.
- Create the `database` group with a GID of 50000.
- Create the `dbadmin1` user and configure it with the following requirements:
 - Add the `database` group as a supplementary group.
 - Set the password to `redhat` and force a password change on first login.
 - Allow the password to change after 10 days since the day of the last password change.

- Set the password expiration to 30 days since the day of the last password change.
- Allow the user to use the `sudo` command to run any command as the superuser.
- Configure the default umask as 007 for the `dbadmin1` user.
- Create the `/home/dbadmin1/grading/review2` directory with `dbadmin1` as the owning user and the `database` group as the owning group.
- Configure the `/home/dbadmin1/grading/review2` directory so that the `database` group owns any file or sub-directory that is created in this directory, irrespective of which user created the file. Configure the permissions on the directory to allow members of the `database` group to access the directory and to create contents in it. All other users should have read and execute permissions on the directory.
- Ensure that users are allowed to delete only files that they own from the `/home/dbadmin1/grading/review2` 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-rh124-review2
```

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-rh124-review2
```


► Solution

Manage Users and Groups, Permissions, and Processes



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.

Manage user and group accounts, set permissions on files and directories, and manage processes.

Outcomes

- Manage user accounts and groups.
- Set permissions on files and directories.
- Identify and manage high CPU-consuming processes.

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-rh124-review2
```

1. Log in to `serverb` as the `student` user.

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

2. Identify and terminate the process that currently uses the most CPU time.
 - 2.1. Use the `top` command to view the real-time system CPU consumption.

```
[student@serverb ~]$ top
```

- 2.2. From the interactive interface of the `top` command, look at the `%CPU` column and confirm that a `dd` process is consuming the most CPU resources.

```
...output omitted...
PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+  COMMAND
2303 student    20   0  217048   944   876  R   99.7   0.1  100:11.64  dd
...output omitted...
```

The `dd` process in the preceding output has the 2303 PID. This process is consuming 99.7% of the CPU resources. The PID and the percentage of CPU resource consumption would vary in your system.

- 2.3. From the interactive interface of the `top` command, type `k` to kill the `dd` process with the 2303 PID, as you determined in the preceding step. After you type `k` in the `top` command, if the default PID that is shown in the prompt matches the PID of the process to terminate, then press the `Enter` key. If the suggested PID does not match, then specify the PID interactively.

```
...output omitted...
PID to signal/kill [default pid = 2303] Enter
...output omitted...
```

- 2.4. Use the default `SIGTERM` signal to terminate the process.

```
...output omitted...
Send pid 2833 signal [15/sigterm] Enter
...output omitted...
```

- 2.5. Press the `q` key to quit the interactive interface of the `top` command.

3. Create the `database` group with a GID of 50000.

- 3.1. Switch to the `root` user.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 3.2. Create the `database` group with a GID of 50000.

```
[root@serverb ~]# groupadd -g 50000 database
```

4. Create the `dbadmin1` user. Add the `database` group as a supplementary group. Set the password to `redhat` and force a password change on the user's first login. Allow the password to change after 10 days since the day of the last password change. Set the password expiration to 30 days since the day of the last password change. Allow the user to use the `sudo` command to run any command as the superuser. Configure the default `umask` as `007`.

- 4.1. Create the `dbadmin1` user. Add the `database` group as a supplementary group.

```
[root@serverb ~]# useradd -G database dbadmin1
```

- 4.2. Set the password of the `dbadmin1` user to `redhat`.

```
[root@serverb ~]# passwd dbadmin1
Changing password for user dbadmin1.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 4.3. Force the `dbadmin1` user to change its password on the next login.

```
[root@serverb ~]# chage -d 0 dbadmin1
```

- 4.4. Set the password's minimum age of the `dbadmin1` user to 10 days.

```
[root@serverb ~]# chage -m 10 dbadmin1
```

- 4.5. Set the password's maximum age of the `dbadmin1` user to 30 days.

```
[root@serverb ~]# chage -M 30 dbadmin1
```

- 4.6. Enable the `dbadmin1` user to use the `sudo` command to run any command as the superuser. Use the `vim /etc/sudoers.d/dbadmin1` command to create the file and add the following content:

```
[root@serverb ~]# vim /etc/sudoers.d/dbadmin1
dbadmin1 ALL=(ALL) ALL
```

- 4.7. Switch to the `dbadmin1` user. Append the `umask 007` line to the `/home/dbadmin1/.bashrc` file.

```
[root@serverb ~]# su - dbadmin1
[dbadmin1@serverb ~]$ echo "umask 007" >> .bashrc
```

- 4.8. Source the `~/ .bashrc` file to update the `umask`.

```
[dbadmin1@serverb ~]$ source ~/.bashrc
```

5. Create the `/home/dbadmin1/grading/review2` directory with `dbadmin1` as the owning user and the `database` group as the owning group.

- 5.1. Use the `mkdir` command `-p` option to create the `/home/dbadmin1/grading/review2` directory.

```
[dbadmin1@serverb ~]$ mkdir -p /home/dbadmin1/grading/review2
```

- 5.2. Recursively set `dbadmin1` and `database` as the respective owning user and group of the `/home/dbadmin1/` directory and subdirectories.

```
[dbadmin1@serverb ~]$ chown -R dbadmin1:database /home/dbadmin1/
```

- 5.3. Recursively set group `execute` permissions on the `/home/dbadmin1` directory and subdirectories. This permission allow members of the `database` group to traverse the `/home/dbadmin1` directory structure.

```
[dbadmin1@serverb ~]$ chmod -R g+x /home/dbadmin1
```

6. Configure the `/home/dbadmin1/grading/review2` directory to allow members of the `database` group to create contents in it. All other users should have read and execute permissions on the directory.
 - 6.1. Apply the SetGID special permission on the `/home/dbadmin1/grading/review2` directory so that the `database` group owns files that are created in the directory.

```
[dbadmin1@serverb ~]$ chmod g+s /home/dbadmin1/grading/review2
```

- 6.2. Apply the 775 permission mode on the `/home/dbadmin1/grading/review2` directory.

```
[dbadmin1@serverb ~]$ chmod 775 /home/dbadmin1/grading/review2
```

7. Ensure that users are allowed to delete only files that they own from the `/home/dbadmin1/grading/review2` directory.
 - 7.1. Apply the sticky bit special permission on the `/home/dbadmin1/grading/review2` directory.

```
[dbadmin1@serverb ~]$ chmod o+t /home/dbadmin1/grading/review2
```

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

```
[dbadmin1@serverb ~]$ exit
logout
[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-rh124-review2
```

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-rh124-review2
```

► Lab

Configure and Manage a Server



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.

Configure, secure, and use the SSH service to access a remote machine, and manage packages with the `dnf` utility.

Outcomes

- Create a new SSH key pair.
- Disable SSH logins as the `root` user.
- Disable password-based SSH logins.
- Install packages and package modules by using the `dnf` command.

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-rh124-review3
```

Specifications

- Log in to `serverb` as the `student` user.
- Generate SSH keys for the `student` user. Do not protect the private key with a passphrase. Save the private and public keys as the `/home/student/.ssh/review3_key` and `/home/student/.ssh/review3_key.pub` files respectively.
- Configure the `student` user on `servera` to accept logins that are authenticated by the `review3_key` SSH key pair. The `student` user on `serverb` should be able to log in to `servera` by using SSH without entering a password.
- On `serverb`, configure the `sshd` service to prevent the `root` user from logging in.

- On `serverb`, configure the `sshd` service to prevent users from using their passwords to log in. Users should still be able to authenticate logins by using an SSH key pair.
- Install the `zsh` package on 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 rhcsa-rh124-review3
```

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-rh124-review3
```

► Solution

Configure and Manage a Server



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.

Configure, secure, and use the SSH service to access a remote machine, and manage packages with the `dnf` utility.

Outcomes

- Create a new SSH key pair.
- Disable SSH logins as the `root` user.
- Disable password-based SSH logins.
- Install packages and package modules by using the `dnf` command.

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-rh124-review3
```

1. Log in to `serverb` as the `student` user.

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

2. Generate SSH keys for the `student` user. Do not protect the private key with a passphrase. Name the private and public key files `/home/student/.ssh/review3_key` and `/home/student/.ssh/review3_key.pub` respectively.


```
[student@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/
id_rsa): /home/student/.ssh/review3_key
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/review3_key.
Your public key has been saved in /home/student/.ssh/review3_key.pub.
The key fingerprint is:
SHA256:Uqefehw+vRfm94fQZDoz/6IfNYSLK/0piQ4n6lrKIbY student@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]-----+
| .+=oBo+          |
| ...0 * =         |
|.. + % =          |
| .+.B =.          |
|...*..o S         |
|E=. o + .         |
| . = oo o .       |
|   *... .         |
|   .oo.           |
+-----[SHA256]-----+
```

3. Configure the student user on servera to accept logins that are authenticated by the review3_key SSH key pair. The student user on serverb should be able to log in to servera by using SSH without entering a password.

3.1. Export the review3_key public key to servera from serverb.

```
[student@serverb ~]$ ssh-copy-id -i .ssh/review3_key.pub student@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/review3.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
student@servera's password: student
```

Number of key(s) added: 1

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

- 3.2. Verify that you can log in to servera from serverb as the student user by using the review3_key private key without being prompted for the password.

```
[student@serverb ~]$ ssh -i .ssh/review3_key student@servera
...output omitted...
[student@servera ~]$
```

- 3.3. Exit from servera.

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

4. On serverb, configure the sshd service to prevent the root user from logging in.
 - 4.1. Set the PermitRootLogin parameter to no in the /etc/ssh/sshd_config file. Use the `sudo vim /etc/ssh/sshd_config` command to edit the configuration file.
 - 4.2. Reload the sshd service.

```
[student@serverb ~]$ sudo systemctl reload sshd.service
```

5. On serverb, configure the sshd service to prevent users from using their passwords to log in. Users should still be able to authenticate logins by using SSH keys.
 - 5.1. Set the PasswordAuthentication parameter to no in the /etc/ssh/sshd_config file. Use the `sudo vim /etc/ssh/sshd_config` command to edit the configuration file.
 - 5.2. Reload the sshd service.

```
[student@serverb ~]$ sudo systemctl reload sshd.service
```

6. Install the zsh package.

```
[student@serverb ~]$ sudo dnf install zsh
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  zsh-5.8-9.el9.x86_64
Complete!
```

- 6.1. 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 rhcsa-rh124-review3
```

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-rh124-review3
```

► Lab

Manage Networks

**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.

Configure and test network connectivity.

Outcomes

- Configure network settings.
- Test network connectivity.
- Set a static hostname.
- Use locally resolvable canonical hostnames to connect to systems.

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-rh124-review4
```

**Important**

When you use the **ssh** command to adjust networking settings, an incorrect command might hang or lock out your session. You are then disconnected from the server, and thus the server becomes inaccessible. You must adjust the network configuration through the server console, locally or through a remote console.

In this review, open the **serverb** machine console to adjust the networking settings.

Specifications

- On `serverb`, determine the name of the Ethernet interface and its active connection profile.
- On `serverb`, create and activate a `static` connection profile for the available Ethernet interface. The `static` profile statically sets network settings and does not use DHCP. Configure the `static` profile to use the network settings in the following table:

Parameter	Setting
IPv4 address	172.25.250.111
Netmask	255.255.255.0
Gateway	172.25.250.254
DNS Server	172.25.250.254

- Set the `serverb` hostname to `server-review4.lab4.example.com`.
- On `serverb`, set `client-review4` as the canonical hostname for the `servera` `172.25.250.10` IPv4 address.
- Configure the `static` connection profile with an additional IPv4 address of `172.25.250.211` with a netmask of `255.255.255.0`. Do not remove the existing IPv4 address. Ensure that `serverb` responds to all addresses when the `static` connection is active.
- On `serverb`, restore the original network settings by activating the original network connection profile.

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-rh124-review4
```

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-rh124-review4
```

► Solution

Manage Networks



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.

Configure and test network connectivity.

Outcomes

- Configure network settings.
- Test network connectivity.
- Set a static hostname.
- Use locally resolvable canonical hostnames to connect to systems.

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-rh124-review4
```



Important

When you use the **ssh** command to adjust networking settings, an incorrect command might hang or lock out your session. You are then disconnected from the server, and thus the server becomes inaccessible. You must adjust the network configuration through the server console, locally or through a remote console.

In this review, open the **serverb** machine console to adjust the networking settings.

1. Use the `serverb` console to log in as the `student` user on the `serverb` machine, and switch to the `root` user.

```
serverb login: student
Password: student
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```
2. On `serverb`, determine the Ethernet interface name and the connection profile name that it uses.
 - 2.1. Display the network connection information.

```
[root@serverb ~]# nmcli device status
```

DEVICE	TYPE	STATE	CONNECTION
eth0	ethernet	connected	System eth0
eth1	ethernet	connected	System eth1
lo	loopback	connected (externally)	lo

In this example, `eth0` is the Ethernet interface name. The connection profile name is `System eth0`. Create the static connection profile for this interface.



Note The network interface and connection profile names might differ from the previous output. Use the name that your system shows to replace the `ethX` placeholder name in subsequent steps.

3. On `serverb`, create the static connection profile for the `ethX` interface. Set the network settings statically so that it does not use DHCP. When done, activate that connection profile. Base the settings on the following table:

IPv4 address	172.25.250.111
Netmask	255.255.255.0
Gateway	172.25.250.254
DNS server	172.25.250.254

- 3.1. Create the static connection profile with the provided network settings.

```
[root@serverb ~]# nmcli connection add con-name static type ethernet \
ifname ethX ipv4.addresses '172.25.250.111/24' ipv4.gateway '172.25.250.254' \
ipv4.dns '172.25.250.254' ipv4.method manual
Connection 'static' (ac8620e6-b77e-499f-9931-118b8b015807) successfully added.
```

- 3.2. Activate the new connection profile.

```
[root@serverb ~]# nmcli connection up static
```

4. Set the `serverb` hostname to `server-review4.lab4.example.com`. Verify the new hostname.

4.1. Configure `server-review4.lab4.example.com` as the new hostname.

```
[root@serverb ~]# hostnamectl hostname server-review4.lab4.example.com
[root@serverb ~]# hostname
server-review4.lab4.example.com
```

5. On `serverb`, set `client-review4` as the canonical hostname for the `servera` `172.25.250.10` IPv4 address.

5.1. Edit the `/etc/hosts` file and add `client-review4` as a name for the `172.25.250.10` IPv4 address.

```
172.25.250.10 client-review4
```

5.2. Verify that you can reach the `servera` `172.25.250.10` IPv4 address by using the canonical `client-review4` hostname.

```
[root@serverb ~]# ping -c2 client-review4
PING client-review4 (172.25.250.10) 56(84) bytes of data.
64 bytes from client-review4 (172.25.250.10): icmp_seq=1 ttl=64 time=0.259 ms
64 bytes from client-review4 (172.25.250.10): icmp_seq=2 ttl=64 time=0.391 ms

--- client-review4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 33ms
rtt min/avg/max/mdev = 0.259/0.325/0.391/0.066 ms
```

6. Modify the `static` connection profile to configure the additional `172.25.250.211` IPv4 address with the `255.255.255.0` netmask. Do not remove the existing IPv4 address. Verify that `serverb` responds to all addresses when the `static` connection profile is active.

6.1. Add the `172.25.250.211` IP address to the `static` connection.

```
[root@serverb ~]# nmcli connection modify static \
+ipv4.addresses '172.25.250.211/24'
```

6.2. Activate the new IP address.

```
[root@serverb ~]# nmcli connection up static
...output omitted...
```

6.3. From `workstation`, use the `ping` command to verify that the `172.25.250.211` IPv4 address is reachable.


```
[student@workstation ~]$ ping -c2 172.25.250.211
PING 172.25.250.211 (172.25.250.211) 56(84) bytes of data.
64 bytes from 172.25.250.211: icmp_seq=1 ttl=64 time=0.246 ms
64 bytes from 172.25.250.211: icmp_seq=2 ttl=64 time=0.296 ms

--- 172.25.250.211 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 50ms
rtt min/avg/max/mdev = 0.246/0.271/0.296/0.025 ms
```

7. On `serverb`, restore the original settings by activating the original network profile.

- 7.1. Return to the console and use the `nmcli` command to activate the original network profile.

```
[root@serverb ~]# nmcli connection up "System eth0"
...output omitted...
```

The original connection profile name might differ on `serverb`. Replace the name in this solution with the name from your system. Find the profile name with the `nmcli connection show` command.

- 7.2. From `workstation`, log in to `serverb` as the `student` user to verify that the original network settings are active.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@server-review4 ~]$
```

- 7.3. Exit any extra terminals. Return to the `workstation` system as the `student` user.

```
[student@server-review4 ~]$ 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-rh124-review4
```

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-rh124-review4
```

► Lab

Mount File Systems and Find Files



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.

Mount a file system and locate files based on different criteria.

Outcomes

- Mount an existing file system.
- Find files based on their file name, permissions, and size.

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-rh124-review5
```

Specifications

- Log in to the **serverb** machine as the **student** user and switch to the **root** user.
- Identify the unmounted block device that contains an XFS file system on the **serverb** machine. Mount the block device on the **/review5-disk** directory.
- Find the **review5-path** file. Create the **/review5-disk/review5-path.txt** file that contains a single line with the absolute path to the **review5-path** file.
- Find all the files that the **contractor1** user and the **contractor** group own. The files must also have the **640** octal permissions. Save the list of these files in the **/review5-disk/review5-perms.txt** file.
- Find all files with a size of 100 bytes. Save the absolute paths of these files in the **/review5-disk/review5-size.txt** file.

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-rh124-review5
```

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-rh124-review5
```

► Solution

Mount File Systems and Find Files



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.

Mount a file system and locate files based on different criteria.

Outcomes

- Mount an existing file system.
- Find files based on their file name, permissions, and size.

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-rh124-review5
```

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. Identify the unmounted block device that contains an XFS file system on the **serverb** machine. Mount the block device on the **/review5-disk** directory.
 - 2.1. Identify the unmounted block device that contains the XFS file system.

```
[root@serverb ~]# lsblk -fs
NAME FSTYPE LABEL UUID MOUNTPOINT
...output omitted...
vdb1 xfs 7694653c-45f6-4749-bd87-f2f69c37daa7
└─vdb
...output omitted...
```

From the preceding output, the `vdb1` block device contains the XFS file system, which is not mounted on the system.

- 2.2. Create the `/review5-disk` directory.

```
[root@serverb ~]# mkdir /review5-disk
```

- 2.3. Mount the `vdb1` block device on the `/review5-disk` directory.

```
[root@serverb ~]# mount /dev/vdb1 /review5-disk
```

- 2.4. Verify that the `vdb1` block device is mounted on the `/review5-disk` directory.

```
[root@serverb ~]# df -Th
Filesystem Type Size Used Avail Use% Mounted on
...output omitted...
/dev/vdb1 xfs 2.0G 47M 2.0G 3% /review5-disk
...output omitted...
```

3. Find the `review5-path` file. Save its absolute path in the `/review5-disk/review5-path.txt` file.

- 3.1. Find the `review5-path` file. Redirect all error messages to the `/dev/null` special file.

```
[root@serverb ~]# find / -iname review5-path 2>/dev/null
/var/tmp/review5-path
```

Note the absolute path to the `review5-path` file from the preceding output.

- 3.2. Use the `vim /review5-disk/review5-path.txt` command and save the absolute path to the `review5-path` file. The following example shows the expected content of the `/review5-disk/review5-path.txt` file.

```
[root@serverb ~]# cat /review5-disk/review5-path.txt
/var/tmp/review5-path
```

4. Find all files that the `contractor1` user and the `contractor` group own. The files must have `640` octal permissions. Save the absolute paths of these files in the `/review5-disk/review5-perms.txt` file.

- 4.1. Find all the files that the `contractor1` user and the `contractor` group own and that have `640` octal permissions. Redirect all the errors to the `/dev/null` special file.

```
[root@serverb ~]# find / -user contractor1 \
-group contractor -perm 640 2>/dev/null
/usr/share/review5-perms
```

Only the `/usr/share/review5-perms` file meets the criteria of the preceding `find` command. Note the absolute path to the `review5-perms` file.

- 4.2. Use the `vim /review5-disk/review5-perms.txt` command and save the absolute path of the `review5-perms` file. The following example shows the expected content of the `/review5-disk/review5-perms.txt` file.

```
[root@serverb ~]# cat /review5-disk/review5-perms.txt
/usr/share/review5-perms
```

5. Find all the files with a size of 100 bytes. Save the absolute paths of these files in the `/review5-disk/review5-size.txt` file.

- 5.1. Find all the files with a size of exactly 100 bytes. Redirect all the errors to the `/dev/null` special file.

```
[root@serverb ~]# find / -type f -size 100c 2>/dev/null
/usr/share/licenses/ethtool/LICENSE
/usr/share/doc/libuser
/usr/share/doc/plymouth/AUTHORS
...output omitted...
/opt/review5-size
...output omitted...
```

The preceding output might vary depending on the number of files that match the size criteria in your system. Note the absolute paths to all the files from the preceding output.

- 5.2. Use the `vim /review5-disk/review5-size.txt` command and save the absolute path of the files from the preceding output. The following example shows the expected content of the `/review5-disk/review5-size.txt` file.

```
[root@serverb ~]# cat /review5-disk/review5-size.txt
...output omitted...
/opt/review5-size
...output omitted...
```

- 5.3. 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 rhcsa-rh124-review5
```

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-rh124-review5
```

