Join the explorers, builders, and individuals who boldly offer new solutions to old problems. For open source, innovation is only possible because of the people behind it.

Red Hat Training and Certification

STUDENT WORKBOOK

Red Hat Enterprise Linux 8.0 RH199

RHCSA RAPID TRACK

Edition 1

Red Hat

RH199-RHEL8.0-en-1-20190531

Copyright © 2019 Red Hat, Inc.

Red Hat Enterprise Linux 8.0 RH199
RHCSA Rapid Track
Edition 1 20190531
Publication date 20190531

Authors: Adrian Andrade, Fiona Allen, Herve Quatremain, Marc Kesler, Saumik Paul, Snehangshu Karmakar, Victor Costea
Editor: Steven Bonneville, Philip Sweany, Ralph Rodriguez, David Sacco, Heather Charles, David O'Brien, Seth Kenlon

Copyright © 2019 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2019 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 e-mail training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered 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 and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: Achyut Madhusudan, Rudolf Kastl, Rob Locke, Heider Souza, Michael Phillips, Dallas Spohn

Document Conventions ix

Introduction xi
RHCSA Rapid Track ... xi
Orientation to the Classroom Environment ... xii
Controlling Your Systems ... xiii
Internationalization ... xiv

1. Accessing Systems and Obtaining Support 1
Accessing the Command Line ... 2
Quiz: Accessing the Command Line ... 7
Configuring SSH Key-based Authentication ... 11
Guided Exercise: Configuring SSH Key-based Authentication ... 15
Getting Help From Red Hat Customer Portal ... 21
Guided Exercise: Getting Help from Red Hat Customer Portal ... 30
Detecting and Resolving Issues with Red Hat Insights ... 32
Quiz: Detecting and Resolving Issues with Red Hat Insights ... 40
Summary ... 42

2. Navigating File Systems 43
Describing Linux File System Hierarchy Concepts ... 44
Quiz: Describing Linux File System Hierarchy Concepts ... 47
Managing Files Using Command-line Tools ... 51
Guided Exercise: Managing Files Using Command-line Tools ... 57
Making Links Between Files ... 62
Guided Exercise: Making Links Between Files ... 66
Summary ... 68

3. Managing Local Users and Groups 69
Describing User and Group Concepts ... 70
Quiz: Describing User and Group Concepts ... 73
Gaining Superuser Access ... 77
Guided Exercise: Gaining Superuser Access ... 82
Managing Local User Accounts ... 88
Guided Exercise: Managing Local User Accounts ... 91
Managing Local Group Accounts ... 94
Guided Exercise: Managing Local Group Accounts ... 97
Managing User Passwords ... 101
Guided Exercise: Managing User Passwords ... 105
Lab: Managing Local Users and Groups ... 109
Summary ... 114

4. Controlling Access to Files 115
Managing File System Permissions from the Command Line ... 116
Guided Exercise: Managing File System Permissions from the Command Line ... 120
Managing Default Permissions and File Access ... 124
Guided Exercise: Managing Default Permissions and File Access ... 128
Lab: Controlling Access to Files ... 133
Summary ... 140

5. Managing SELinux Security 141
Changing the SELinux Enforcement Mode ... 142
Guided Exercise: Changing the SELinux Enforcement Mode ... 146
Controlling SELinux File Contexts ... 149
Guided Exercise: Controlling SELinux File Contexts ... 153
Adjusting SELinux Policy with Booleans ... 156
Guided Exercise: Adjusting SELinux Policy with Booleans ... 158
Investigating and Resolving SELinux Issues ... 161

Guided Exercise: Investigating and Resolving SELinux Issues 165
Lab: Managing SELinux Security 169
Summary 175

6. Tuning System Performance **177**
Killing Processes 178
Guided Exercise: Killing Processes 184
Monitoring Process Activity 189
Guided Exercise: Monitoring Process Activity 193
Adjusting Tuning Profiles 198
Guided Exercise: Adjusting Tuning Profiles 203
Influencing Process Scheduling 205
Guided Exercise: Influencing Process Scheduling 209
Lab: Tuning System Performance 213
Summary 218

7. Installing and Updating Software Packages **219**
Registering Systems for Red Hat Support 220
Quiz: Registering Systems for Red Hat Support 224
Installing and Updating Software Packages with Yum 226
Guided Exercise: Installing and Updating Software Packages with Yum 233
Enabling Yum Software Repositories 239
Guided Exercise: Enabling Yum Software Repositories 243
Managing Package Module Streams 247
Guided Exercise: Managing Package Module Streams 254
Lab: Installing and Updating Software Packages 260
Summary 267

8. Managing Basic Storage **269**
Mounting and Unmounting File Systems 270
Guided Exercise: Mounting and Unmounting File Systems 274
Adding Partitions, File Systems, and Persistent Mounts 277
Guided Exercise: Adding Partitions, File Systems, and Persistent Mounts 287
Managing Swap Space 291
Guided Exercise: Managing Swap Space 295
Lab: Managing Basic Storage 300
Summary 310

9. Controlling Services and the Boot Process **311**
Identifying Automatically Started System Processes 312
Guided Exercise: Identifying Automatically Started System Processes 317
Controlling System Services 321
Guided Exercise: Controlling System Services 325
Selecting the Boot Target 329
Guided Exercise: Selecting the Boot Target 334
Resetting the Root Password 337
Guided Exercise: Resetting the Root Password 341
Repairing File System Issues at Boot 343
Guided Exercise: Repairing File System Issues at Boot 345
Lab: Controlling Services and Daemons 348
Summary 352

10. Managing Networking **353**
Validating Network Configuration 354
Guided Exercise: Validating Network Configuration 360
Configuring Networking from the Command Line 363
Guided Exercise: Configuring Networking from the Command Line 369

Editing Network Configuration Files 376
Guided Exercise: Editing Network Configuration Files 379
Configuring Host Names and Name Resolution 384
Guided Exercise: Configuring Host Names and Name Resolution 387
Lab: Managing Networking 392
Summary 397

11. Analyzing and Storing Logs **399**
Describing System Log Architecture 400
Quiz: Describing System Log Architecture 402
Reviewing Syslog Files 406
Guided Exercise: Reviewing Syslog Files 410
Reviewing System Journal Entries 412
Guided Exercise: Reviewing System Journal Entries 417
Preserving the System Journal 421
Guided Exercise: Preserving the System Journal 424
Maintaining Accurate Time 426
Guided Exercise: Maintaining Accurate Time 430
Lab: Analyzing and Storing Logs 435
Summary 440

12. Implementing Advanced Storage Features **441**
Creating Logical Volumes 442
Guided Exercise: Creating Logical Volumes 449
Extending Logical Volumes 455
Guided Exercise: Extending Logical Volumes 460
Managing Layered Storage with Stratis 464
Guided Exercise: Managing Layered Storage with Stratis 469
Compressing and Deduplicating Storage with VDO 475
Guided Exercise: Compressing and Deduplicating Storage with VDO 478
Lab: Implementing Advanced Storage Features 483
Summary 493

13. Scheduling Future Tasks **495**
Scheduling Recurring System Jobs 496
Guided Exercise: Scheduling Recurring System Jobs 499
Managing Temporary Files 503
Guided Exercise: Managing Temporary Files 506
Quiz: Scheduling Future Tasks 510
Summary 514

14. Accessing Network-Attached Storage **515**
Mounting Network-Attached Storage with NFS 516
Guided Exercise: Managing Network-Attached Storage with NFS 521
Automounting Network-Attached Storage 526
Guided Exercise: Automounting Network-Attached Storage 529
Lab: Accessing Network-Attached Storage 535
Summary 542

15. Managing Network Security **543**
Managing Server Firewalls 544
Guided Exercise: Managing Server Firewalls 552
Lab: Managing Network Security 557
Summary 566

16. Comprehensive Review **567**
Comprehensive Review 568
Lab: Fixing Boot Issues and Maintaining Servers 572

Lab: Configuring and Managing File Systems and Storage 580
Lab: Configuring and Managing Server Security 587

DOCUMENT CONVENTIONS

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

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

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

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

INTRODUCTION

RHCSA RAPID TRACK

The RHCSA Rapid Track course (RH199) is designed for students who already have significant experience with Linux administration. The RHCSA Rapid Track reviews the tasks covered in Red Hat System Administration I & II at an accelerated pace. The course builds on a student's foundational understanding of command-line based Linux administration. Students should be able to execute common commands such as cp, grep, sort, mkdir, tar, mkfs, ssh, and yum from the command prompt. Students should also be familiar working with common command options and accessing man pages for help. Students lacking this knowledge are strongly encouraged to take Red Hat System Administration I (RH124).

COURSE OBJECTIVES

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

AUDIENCE

- The RHCSA Rapid Track course (RH199/RH200) is among the most important courses in the Linux portfolio. It is designed to be a curriculum entry point for IT professionals with significant exposure to Linux administration. This course covers 9 days of rather dense System Administration I & II content in 4 days. To do so requires that elemental concepts are either not covered or only done so in a cursory fashion. Students who are unsure whether they are qualified to take RH199 are strongly encouraged to take the online pre-assessment.

PREREQUISITES

- Students for this class should have one to three years of full time Linux administration experience.

ORIENTATION TO THE CLASSROOM ENVIRONMENT

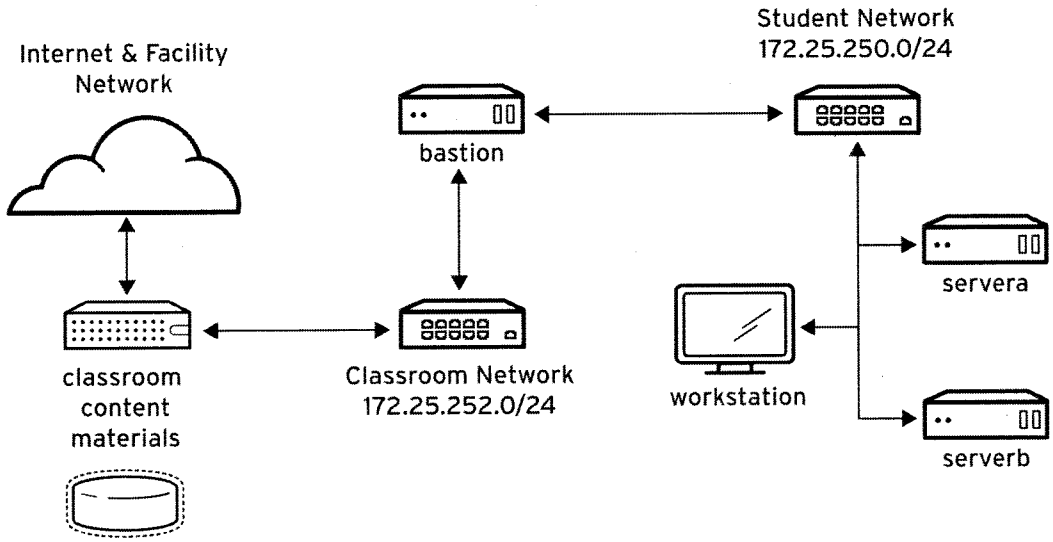


Figure 0.1: Classroom environment

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

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

Classroom Machines

MACHINE NAME	IP ADDRESSES	ROLE
bastion.lab.example.com	172.25.250.254	Gateway system to connect student private network to classroom server (must always be running)
workstation.lab.example.com	172.25.250.9	Graphical workstation used for system administration
servera.lab.example.com	172.25.250.10	First server
serverb.lab.example.com	172.25.250.11	Second server

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

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

CONTROLLING YOUR SYSTEMS

In an instructor-led training classroom, students are assigned a physical computer (`foundationX.ilt.example.com`), which is used to access their virtual machines running on that host. Students are automatically logged in to the physical machine as the `kiosk` user, with the password `redhat`. The classroom systems with which you work are virtual machines running on that host.

On `foundationX`, a special command called **`rht-vmctl`** is used to work with the virtual machines. The commands in the following table should be run as the `kiosk` user on `foundationX`, and can be used with any of the virtual machines.

`rht-vmctl` Commands

ACTION	COMMAND
Start `server` ***machine***	**`rht-vmctl start server`**
View "physical console" to log in and work with the `server` ***machine***	**`rht-vmview view server`**
Reset `server` ***machine to its previous state and restart the virtual machine***	**`rht-vmctl reset server`**

At the start of a lab exercise, if the instruction "reset your servera" appears, you should run the **`rht-vmctl reset servera`** command on the `foundationX` system as the `kiosk` user. Likewise, if the instruction "reset your workstation" appears, you should run the **`rht-vmctl reset workstation`** command on `foundationX` as the `kiosk` user.

INTERNATIONALIZATION

PER-USER LANGUAGE SELECTION

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

Language Settings

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

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

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

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

NOTE

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

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

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

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

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

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

Input Method Settings

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

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

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

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

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

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

NOTE

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

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

SYSTEM-WIDE DEFAULT LANGUAGE SETTINGS

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

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

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

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

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

IMPORTANT

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

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

LANGUAGE PACKS

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

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

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

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

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

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

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

IMPORTANT

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

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

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

REFERENCES

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

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

LANGUAGE CODES REFERENCE

NOTE

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

Language Codes

LANGUAGE	LANGPACKS SUFFIX	$LANG VALUE
English (US)	en	en_US.utf8

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

CHAPTER 1

ACCESSING SYSTEMS AND OBTAINING SUPPORT

GOAL

Log in to local and remote Linux system, and investigate problem resolution methods provided through Red Hat Support and Red Hat Insights.

OBJECTIVES

- Log in to a Linux system on a local text console and run simple commands using the shell.
- Configure key-based authentication for a user account to log in to remote systems securely without a password.
- Describe key resources available through the Red Hat Customer Portal, and find information from Red Hat documentation and the Knowledgebase.
- Analyze servers for issues, remediate or resolve them, and confirm the solution with Red Hat Insights.

SECTIONS

- Accessing the Command Line (and Quiz)
- Configuring SSH Key-Based Authentication (and Guided Exercise)
- Getting Help from Red Hat Customer Portal (and Guided Exercise)
- Detecting and Resolving Issues with Red Hat Insights (and Quiz)

ACCESSING THE COMMAND LINE

OBJECTIVES

After completing this section, you should be able to log in to a Linux system and run simple commands using the shell.

INTRODUCTION TO THE BASH SHELL

A *command line* is a text-based interface which can be used to input instructions to a computer system. The Linux command line is provided by a program called the *shell*. Various options for the shell program have been developed over the years, and different users can be configured to use different shells. Most users, however, stick with the current default.

The default shell for users in Red Hat Enterprise Linux is the ***GNU Bourne-Again Shell*** (**bash**). Bash is an improved version of one of the most successful shells used on UNIX-like systems, the ***Bourne Shell*** (**sh**).

When a shell is used interactively, it displays a string when it is waiting for a command from the user. This is called the *shell prompt*. When a regular user starts a shell, the default prompt ends with a **$** character, as shown below.

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

The **$** character is replaced by a **#** character if the shell is running as the superuser, `root`. This makes it more obvious that it is a superuser shell, which helps to avoid accidents and mistakes which can affect the whole system. The superuser shell prompt is shown below.

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

Using **bash** to execute commands can be powerful. The **bash** shell provides a scripting language that can support automation of tasks. The shell has additional capabilities that can simplify or make possible operations that are hard to accomplish efficiently with graphical tools.

NOTE

The **bash** shell is similar in concept to the command-line interpreter found in recent versions of Microsoft Windows, **`cmd.exe`**, although **bash** has a more sophisticated scripting language. It is also similar to Windows PowerShell in Windows 7 and Windows Server 2008 R2 and later. Administrators using the Apple Mac who use the ***Terminal*** utility may be pleased to note that **bash** is the default shell in macOS.

SHELL BASICS

Commands 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 may be followed by one or more *options*, which adjust the behavior of the command or what it will do. Options normally start with one or two dashes (**-a** or **--all**, for example) to distinguish them from arguments. Commands may also be followed by one or more *arguments*, which often indicate a target that the command should operate upon.

For example, the command **usermod -L user01** has a command (**usermod**), an option (**-L**), and an argument (**user01**). The effect of this command is to lock the password of the `user01` user account.

LOGGING IN TO A LOCAL COMPUTER

To run the shell, you need to log in to the computer on a *terminal*. A terminal is a text-based interface used to enter commands into and print output from a computer system. There are several ways to do this.

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

The computer might provide a graphical login prompt on one of the virtual consoles. You can use this 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. This allows resources which would be used by the graphical environment to be used by the server's services instead.

In Red Hat Enterprise Linux 8, if the graphical environment is available, the login screen will run on the first virtual console, called `tty1`. Five additional text login prompts are available on virtual consoles two through six.

If you log in using the graphical login screen, your graphical environment will start on the first virtual console that is not currently being used by a login session. Normally, your graphical session will replace the login prompt on the second virtual console (`tty2`). However, if that console is in use by an active text login session (not just a login prompt), 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 log in as another user on the graphical login screen or use the ***Switch User*** menu item to switch users in the graphical environment without logging out, another graphical environment will be started for that user on the next free virtual console.

When you log out of a graphical environment, it will exit and the physical console will automatically switch 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 5 and earlier, the first six virtual consoles always provided text login prompts. If the graphical environment is running, it is on virtual console *seven* (accessed through **`Ctrl`**+**`Alt`**+**`F7`**).

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

The serial console would normally be used to fix the server if its own network card became misconfigured and logging in over its own network connection became impossible. Most of the time, however, headless servers are accessed by other means over the network.

LOGGING IN OVER THE NETWORK

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 actually 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 machine `host` 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) do not allow users to use a password to log in with **`ssh`** for tighter security. 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 containing 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 and if their matching public key is installed in that account on that remote server, it will log them in without asking for a password.

In the next example, a user with a shell prompt on the machine `host` logs in to `remotehost` as `remoteuser` using **`ssh`**, using public key authentication. The **`-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 this to work, the private key file must be readable only by the user that owns the file. In the preceding example, where the private key is in the **mylab.pem** file, the command **chmod 600 mylab.pem** could be used to ensure this. How to set file permissions is discussed in more detail in a later chapter.

Users might also have private keys configured 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 on this topic.

NOTE

The first time you log in to a new machine, you will be 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 you connect to a remote host with **ssh**, the remote host sends **ssh** its *host key* to authenticate itself and to help set up encrypted communication. The **ssh** command compares that against a list of saved host keys to make sure it has not changed. If the host key has changed, this might indicate that someone is trying to pretend to be that host to hijack the connection which is also known as man-in-the-middle attack. In SSH, host keys protect against man-in-the-middle attacks, these host keys are unique for each server, and they need to be changed periodically and whenever a compromise is suspected.

You will get this warning if your local machine does not have a host key saved for the remote host. If you enter **yes**, the host key that the remote host sent will be accepted and saved for future reference. Login will continue, and you should not see this message again when connecting to this host. If you enter **no**, the host key will be rejected and the connection closed.

If the local machine does have a host key saved and it does not match the one actually sent by the remote host, the connection will automatically be closed with a warning.

LOGGING OUT

When you are finished using 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 is an example of a user logging out of an SSH session:

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

REFERENCES

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

*Note: Some details of the **console**(4) man page, involving **init**(8) and **inittab**(5), are outdated.*

For more information on OpenSSH and public key authentication, refer to the *Using secure communications between two systems with OpenSSH* chapter in the *Red Hat Enterprise Linux 8 Securing networks* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/securing_networks/assembly_using-secure-communications-with-openssh-securing-networks

NOTE

Instructions on how to read **man** pages and other online help documentation is included at the end of the next section.

QUIZ

ACCESSING THE COMMAND LINE

Choose the correct answer to the following questions:

1. **Which term describes the interpreter that executes commands typed as strings?**
   a. Command
   b. Console
   c. Shell
   d. Terminal

2. **Which term describes the visual cue that indicates 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 used 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

ACCESSING THE COMMAND LINE

Choose the correct answer to the following questions:

1. **Which term describes the interpreter that executes commands typed as strings?**
   a. Command
   b. Console
   c. Shell
   d. Terminal

2. **Which term describes the visual cue that indicates 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 used 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

CONFIGURING SSH KEY-BASED AUTHENTICATION

OBJECTIVES

After completing this section, you should be able to 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 an SSH server to allow you to authenticate without a password by using key-based authentication. This is based on a private-public key scheme.

To do this, you generate a matched pair of cryptographic key files. One is a private key, the other a matching public key. The private key file is used as the authentication credential and, like a password, must be kept secret and secure. The public key is copied to systems the user wants to connect to, and is used to verify the private key. The public key does not need to be secret.

You put a copy of the public key in your account on the server. When you try to log in, the SSH server can use the public key to issue a challenge that can only be correctly answered by using the private key. As a result, your **ssh** client can automatically authenticate your login to the server with your unique copy of the private key. This allows you to securely access systems in a way that doesn't require you to enter a password interactively every time.

Generating SSH Keys

To create a private key and matching public key for authentication, use the **ssh-keygen** command. By default, your private and public keys are saved in your **~/.ssh/id_rsa** and **~/.ssh/id_rsa.pub** files, respectively.

```
[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:vxutUNPio3QDCyvkYm1oIx35hmMrHpPKWFdIYu3HV+w user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|   .     .       |
|  o o     o      |
| . = o   o .     |
|  o + = S E .    |
| ..O o + * +     |
|.+% O . + B .    |
```

```
|=*oO . . + *      |
|++.      . +.     |
+----[SHA256]-----+
```

If you do not specify a passphrase when **ssh-keygen** prompts you, the generated private key is not protected. In this case, anyone with your private key file could use it for authentication. If you set a passphrase, then you will need to enter that passphrase when you use the private key for authentication. (Therefore, you would be using the private key's passphrase rather than your password on the remote host to authenticate.)

You can run a helper program called **ssh-agent** which can temporarily cache your private key passphrase in memory at the start of your session to get true passwordless authentication. This will be discussed later in this section.

The following example of the **ssh-keygen** command shows the creation of the passphrase-protected private key alongside the public key.

```
[user@host ~]$ ssh-keygen -f .ssh/key-with-pass
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
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]----+
|     . + =.o ...  |
|      = B XEo o.  |
|  . o O X =....   |
| = = = B = o.     |
|= + * * S .       |
|.+ = o + .        |
|  + .             |
|                  |
|                  |
+----[SHA256]-----+
```

The **-f** option with the **ssh-keygen** command determines the files where the keys are saved. In the preceding example, the private and public keys are saved in the **/home/user/.ssh/key-with-pass** **/home/user/.ssh/key-with-pass.pub** files, respectively.

> **WARNING**
> During further SSH keypair generation, unless you specify a unique file name, you are prompted for permission to overwrite the existing **id_rsa** and **id_rsa.pub** files. If you overwrite the existing **id_rsa** and **id_rsa.pub** files, then you must replace the old public key with the new one on all the SSH servers that have your old public key.

Once the SSH keys have been generated, they are stored by default in the **.ssh/** directory of the user's home directory. The permission modes must be 600 on the private key and 644 on the public key.

Sharing the Public Key

Before key-based authentication can be used, the public key needs to be copied to the destination system. The **ssh-copy-id** command copies the public key of the SSH keypair to the destination system. If you omit the path to the public key file while running **ssh-copy-id**, it uses the default **/home/user/.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 the public key is successfully transferred to a remote system, you can authenticate to the remote system using the corresponding private key while logging in to the remote system over SSH. If you omit the path to the private key file while running the **ssh** command, it uses the default **/home/*user*/.ssh/id_rsa** file.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost
Enter passphrase for key '.ssh/key-with-pass': redhatpass
...output omitted...
[user@remotehost ~]$ exit
logout
Connection to remotehost closed.
[user@host ~]$
```

Using ssh-agent for Non-interactive Authentication

If your SSH private key is protected with a passphrase, you normally have to enter the passphrase to use the private key for authentication. However, you can use a program called **ssh-agent** to temporarily cache the passphrase in memory. Then any time that you use SSH to log in to another system with the private key, **ssh-agent** will automatically provide the passphrase for you. This is convenient, and can improve security by providing fewer opportunities for someone "shoulder surfing" to see you type the passphrase in.

Depending on your local system's configuration, if you initially log in to the GNOME graphical desktop environment, the **ssh-agent** program might automatically be started and configured for you.

If you log in on a text console, log in using **ssh**, or use **sudo** or **su**, you will probably need to start **ssh-agent** manually for that session. You can do this with the following command:

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

NOTE

When you run **ssh-agent**, it prints out some shell commands. You need to run these commands to set environment variables used by programs like **ssh-add** to communicate with it. The **eval $(ssh-agent)** command starts **ssh-agent** and runs those commands to automatically set those environment variables for that shell session. It also displays the PID of the **ssh-agent** process.

Once **ssh-agent** is running, you need to tell it the passphrase for your private key or keys. You can do this with the **ssh-add** command.

The following **ssh-add** commands add the private keys from **/home/*user*/.ssh/id_rsa** (the default) and **/home/*user*/.ssh/key-with-pass** files, respectively.

```
[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: redhatpass
Identity added: .ssh/key-with-pass (user@host.lab.example.com)
```

After successfully adding the private keys to the `ssh-agent` process, you can invoke an SSH connection using the **ssh** command. If you are using any private key file other than the default **/home/*user*/.ssh/id_rsa** file, then you must use the **-i** option with the **ssh** command to specify the path to the private key file.

The following example of the **ssh** command uses the default private key file to authenticate to an SSH server.

```
[user@host ~]$ ssh user@remotehost
Last login: Fri Apr  5 10:53:50 2019 from host.example.com
[user@remotehost ~]$
```

The following example of the **ssh** command uses the **/home/*user*/.ssh/key-with-pass** (non-default) private key file to authenticate to an SSH server. The private key in the following example has already been decrypted and added to its parent **ssh-agent** process, so the **ssh** command does not prompt you to decrypt the private key by interactively entering its passphrase.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost
Last login: Mon Apr  8 09:44:20 2019 from host.example.com
[user@remotehost ~]$
```

When you log out of the session that started **ssh-agent**, the process will exit and your the passphrases for your private keys will be cleared from memory.

REFERENCES

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

GUIDED EXERCISE

CONFIGURING SSH KEY-BASED AUTHENTICATION

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

OUTCOMES

You should be able to:

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

BEFORE YOU BEGIN

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

On `workstation`, run **`lab ssh-configure start`** to start the exercise. This script creates the necessary user accounts.

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

1. From `workstation`, open an SSH session to `serverb` as `student`.

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

2. Use the **su** command to switch to the `operator1` user on `serverb`. Use `redhat` as the password of `operator1`.

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

3. Use the **ssh-keygen** command to generate 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+xXhOqsJQQLzBNUldb+jJbyrCZQBERI
 operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|E+*+ooo .        |
|.= o.o o .       |
|o.. = . . o      |
|+. + * . o .     |
|+ = X . S +      |
| + @ +   = .     |
|. + =   o        |
|.o . . . .       |
|o     o..        |
+----[SHA256]-----+
```

4. Use the **ssh-copy-id** command to send the public key of the SSH key pair to `operator1` on `servera`. Use `redhat` as the password of `operator1` on `servera`.

```
[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.
ECDSA key fingerprint is SHA256:ERTdjooOIrIwVSZQnqD5or+JbXfidg0udb3DXBuHWzA.
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
operator1@servera's password: redhat
Number of key(s) added: 1

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

5. Execute the **hostname** command on `servera` remotely using SSH without accessing the remote interactive shell.

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

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

6. Use the **ssh-keygen** command to 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.

> **WARNING**
> If you do not specify the file where the key gets saved, the default file (**/home/*user*/.ssh/id_rsa**) is used. You have already used the default file name when generating SSH keys in the preceding step, so it is vital that you specify a non-default file, otherwise the existing SSH keys will be overwritten.

```
[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:OCtCjfPm5QrbPBgqbEIWCcw5AI4oSlMEbgLrBQ1HWKI
 operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|O=X*             |
|OB=.             |
|E*o.             |
|Booo   .         |
|..= . o S        |
| +.o   o         |
|+.oo+ o          |
|+o.O.+           |
|+ . =o.          |
+----[SHA256]-----+
```

7. Use the **ssh-copy-id** command to send the public key of the passphrase-protected key pair to `operator1` on `servera`.

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

Number of key(s) added: 1

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

Notice that the preceding **ssh-copy-id** command did not prompt you for a password because it used the public key of the passphrase-less private key that you exported to `servera` in the preceding step.

8. Execute the **hostname** command on `servera` remotely with SSH without accessing the remote interactive shell. Use **/home/operator1/.ssh/key2** as the identity file. Specify `redhatpass` as the passphrase, which you set for the private key in the preceding step.

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

Notice that the preceding **ssh** command prompted you for the passphrase you used to protect the private key of the SSH key pair. This passphrase protects the private key. Should an attacker gain access to the private key, the attacker cannot use it to access other systems because the private key itself is protected with a passphrase. The **ssh** command uses a different passphrase than the one for `operator1` on `servera`, requiring users to know both.

You can use **ssh-agent**, as in the following step, to avoid interactively typing in the passphrase while logging in with SSH. Using **ssh-agent** is both more convenient and more secure in situations where the administrators log in to remote systems regularly.

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

```
[operator1@serverb ~]$ eval $(ssh-agent)
Agent pid 21032
[operator1@serverb ~]$ ssh-add .ssh/key2
Enter passphrase for .ssh/key2: redhatpass
Identity added: .ssh/key2 (operator1@serverb.lab.example.com)
```

The preceding **eval** command started **ssh-agent** and configured this shell session to use it. You then used **ssh-add** to provide the unlocked private key to **ssh-agent**.

10. Execute the **hostname** command on `servera` remotely without accessing a remote interactive shell. Use **/home/operator1/.ssh/key2** as the identity file.

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

Notice that the preceding **ssh** command did not prompt you to enter the passphrase interactively.

11. Open another terminal on `workstation` and open an SSH session to `serverb` as `student`.

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

12. On `serverb`, use the **su** command to switch to `operator1` and invoke an SSH connection to `servera`. Use **/home/operator1/.ssh/key2** as the identity file to authenticate using the SSH keys.

12.1. Use the **su** command to switch to `operator1`. Use `redhat` as the password of `operator1`.

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

12.2. Open an SSH session to `servera` as `operator1`.

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

Notice that the preceding **ssh** command prompted you to enter the passphrase interactively because you did not invoke the SSH connection from the shell that you used to start **ssh-agent**.

13. Exit all the shells you are using in the second terminal.

13.1. Log out of `servera`.

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

13.2. Exit the `operator1` and `student` shells on `serverb` to return to the `student` user's shell on `workstation`.

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

13.3. Close the second terminal on `workstation`.

```
[student@workstation ~]$ exit
```

14. Log out of `serverb` on the first terminal and conclude this exercise.

14.1. From the first terminal, exit the `operator1` user's shell on `serverb`.

```
[operator1@serverb ~]$ exit
logout
[student@serverb ~]$
```

The **`exit`** command caused you to exit the `operator1` user's shell, terminating the shell session where **`ssh-agent`** was active, and return to the `student` user's shell on `serverb`.

14.2. Exit the `student` user's shell on `serverb` to return to the `student` user's shell on `workstation`.

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

Finish

On `workstation`, run **`lab ssh-configure finish`** to complete this exercise.

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

This concludes the guided exercise.

GETTING HELP FROM RED HAT CUSTOMER PORTAL

OBJECTIVES

After completing this section, you should be able to describe key resources available through the Red Hat Customer Portal and use them to find information from Red Hat documentation and the Knowledgebase.

ACCESSING SUPPORT RESOURCES ON THE RED HAT CUSTOMER PORTAL

The Red Hat Customer Portal (https://access.redhat.com) provides customers access to documentation, downloads, tools, and technical expertise. Customers can search for solutions, FAQs, and articles through the Knowledgebase. From the Customer Portal, you can:

- Access official product documentation.
- Submit and manage support tickets.
- Manage software subscriptions and entitlements.
- Obtain software downloads, updates, and evaluations.
- Consult tools that can help you optimize the configuration of your systems.

Parts of the site are accessible to everyone, and other areas are only available to customers with active subscriptions. Get help accessing the Customer Portal at https://access.redhat.com/help/.

GETTING ORIENTED TO THE CUSTOMER PORTAL

You can access the Red Hat Customer Portal through a web browser. This section introduces the Customer Portal Tour. The tour can be found at https://access.redhat.com/start.

The tour is a very useful tool for discovering all the portal has to offer and how to get the most out of your Red Hat subscription. After you have logged in to the Red Hat Customer Portal, click ***Tour the Customer Portal***.

Tour the Customer Portal

Figure 1.1: Tour the Customer Portal

The ***WELCOME TO THE RED HAT CUSTOMER PORTAL*** window opens with two options: ***CLOSE*** and ***NEXT***. Click ***NEXT*** to start the tour. This is the first of a sequence of windows that highlight different parts of the interface.

The Top Navigation Bar

The first three stops on the Customer Portal Tour can be found on the top navigation bar of the Red Hat Customer Portal website:

SUBSCRIPTIONS DOWNLOADS CONTAINERS SUPPORT CASES

Figure 1.2: Top Navigation Bar

Subscriptions opens a new page where you can manage your registered systems and your subscriptions and entitlements usage. It lists information about errata that apply and allows you to create *activation keys* that you can use when registering systems to ensure they get entitlements from the correct subscriptions. Note that if you are part of an organization, your Organization Administrator can limit your access to this page.

Downloads opens a new page which gives you access to your product downloads and to request evaluation entitlements for products for which you do not have entitlements.

Support Cases opens a new page which provides access to create, track, and manage your support cases through the Case Management system, assuming that your organization has authorized that level of access.

Your name is the title for the ***User Menu***, which allows you to manage your account, accounts for which you are Organization Administrator, your personal profile, and options for email notifications of new content that is available.

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

Topics Menus

Underneath the top navigation bar on the Customer Portal's main page are menus that you can use to navigate to four major categories of resources available on the site.

Red Hat CUSTOMER PORTAL Products & Services Tools Security Community

Figure 1.3: Resources Menus

Products & Services provides access to the *Product Hubs*, pages that provide access to product-specific evaluations, overviews, getting started guides, and other product support information. You can also access documentation for Red Hat products, direct links to the Knowledgebase of support articles, and information on support policies and how to contact Red Hat Support.

Infrastructure and Management
- Red Hat Enterprise Linux
- Red Hat Virtualization

Cloud Computing
- Red Hat Identity Management
- Red Hat Directory Server

Storage
- Red Hat Certificate System
- Red Hat Satellite

JBoss Development and Management
- Red Hat Subscription Management
- Red Hat Update Infrastructure

JBoss Integration and Automation
- Red Hat Insights
- Red Hat Ansible Tower

Mobile
- Red Hat Ansible Engine

Figure 1.4: Products and Services

The ***Tools*** menu provides links to tools to help you succeed with Red Hat products. The Solution Engine section provides you with an efficient way to search for solutions to your problems quickly, by product, and opening a support ticket if you do not find a satisfactory solution. The Customer Portal Labs section provides a collection of web-based applications and tools to help you improve performance, diagnose issues, identify security problems, and optimize your configurations. For example, the ***Product Life Cycle Checker*** allows you to select a particular product and view its support life cycle schedule. Another tool, the ***Rescue Mode Assistant***, helps you reset the root password of a system, generate diagnostic reports, or fix boot-time problems with file systems. But there are many other tools available at that site.

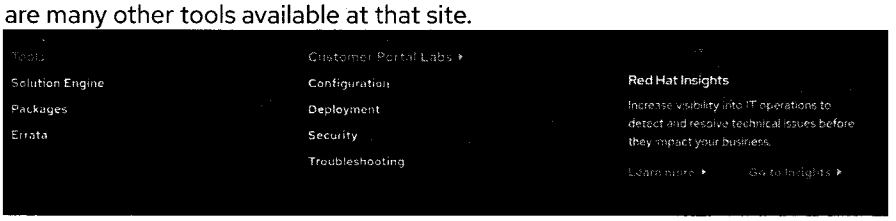


Figure 1.5: Tools menu in Customer Portal

The Security section provides access to the *Red Hat Product Security Center* at https://access.redhat.com/security/. This section also provides information about high-profile security issues, access to the Red Hat CVE Database, the Security channel of the Red Hat Blog, and resources about Red Hat's security response process and how we rate issues and resolve them.

Finally, the Community section is a place where Red Hat experts, customers, and partners can communicate and collaborate. Discussion forums, blogs, and information about upcoming events in your area are available here.

NOTE

You should complete the entire tour at Getting Started with Red Hat [https://access.redhat.com/start], including the sections on how to personalize your Customer Portal experience and exploring the benefits of your Red Hat subscription, to get the full story about the Customer Portal. You will need at least one active subscription on your Customer Portal account to access this page.

SEARCHING THE KNOWLEDGEBASE WITH THE RED HAT SUPPORT TOOL

The Red Hat Support Tool utility, **`redhat-support-tool`**, provides a text-based interface that allows you to search Knowledgebase articles and to file support cases on the Customer Portal from your system's command line. The tool does not have a graphical interface and, because it interacts with the Red Hat Customer Portal, it requires internet access. Run the **`redhat-support-tool`** command using any terminal or SSH connection.

The **`redhat-support-tool`** command may be used in an interactive mode or invoked as a command with options and arguments. The tool's syntax is identical for both methods. By default, the program launches in interactive mode. Use the **`help`** subcommand to see all available commands. Interactive mode supports tab completion and the ability to call programs in the parent shell.

```
[user@host ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help):
```

When first invoked, **redhat-support-tool** prompts for Red Hat Customer Portal subscriber login information. To avoid repetitively supplying this information, the tool asks to store account information in the user's home directory (**~/.redhat-support-tool/redhat-support-tool.conf**). If issues are all filed through a particular Red Hat Customer Portal account, the **--global** option can save account information to **/etc/redhat-support-tool.conf**, along with other system-wide configuration. The tool's **config** command modifies tool configuration settings.

The **redhat-support-tool** command allows subscribers to search and display Knowledgebase content from the Red Hat Customer Portal. The Knowledgebase permits keyword searches, similar to the **man** command. You can enter error codes, syntax from log files, or any mix of keywords to produce a list of relevant solution documents.

The following is an initial configuration and basic search demonstration:

```
[user@host ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): search How to manage system entitlements with subscription-
manager
Please enter your RHN user ID: subscriber
Save the user ID in /home/student/.redhat-support-tool/redhat-support-tool.conf
 (y/n): y
Please enter the password for subscriber: password
Save the password for subscriber in /home/student/.redhat-support-tool/redhat-
support-tool.conf (y/n): y
```

After prompting the user for the required user configuration, the tool continues with the original search request:

```
Type the number of the solution to view or 'e' to return to the previous menu.
  1 [ 253273:VER] How to register and subscribe a system to the Red Hat Customer
    Portal using Red Hat Subscription-Manager
  2 [ 265523:VER] Enabling or disabling a repository using Red Hat Subscription
    Management
  3 [ 100423:VER] Why does subscription-manager list return: "No Installed
    Products found" ?
...output omitted...
Select a Solution: 1
```

Select article number 1 as above and you are prompted to select the section of the document to read. Finally, use the **Q** key to quit the section you are in, or use it repeatedly to quit the **redhat-support-tool** command.

```
Select a Solution: 1

Type the number of the section to view or 'e' to return to the previous menu.
 1 Title
 2 Issue
 3 Environment
 4 Resolution
 5 Display all sections
End of options.
Section: 1
```

```
Title
================================================================================
How to register and subscribe a system to the Red Hat Customer Portal using Red
 Hat Subscription-Manager
URL:        https://access.redhat.com/solutions/253273
Created On:  None
Modified On:  2017-11-29T15:33:51Z

(END) q
Section:
Section: q

Select a Solution: q

Command (? for help): q
[user@hosts ~]#
```

Accessing Knowledgebase Articles by Document ID

Locate online articles directly using the tool's **kb** command with the Knowledgebase document ID. A returned document scrolls on the screen without pagination, but you can redirect it to a file to save it and use **less** to scroll through it a screen at a time.

```
[user@host ~]$ redhat-support-tool kb 253273

Title
================================================================================
How to register and subscribe a system to the Red Hat Customer Portal using Red
 Hat Subscription-Manager
URL:        https://access.redhat.com/solutions/253273
Created On:  None
Modified On:  2017-11-29T15:33:51Z

Issue
================================================================================
*   How to register a new `Red Hat Enterprise Linux` system to the Customer Portal
 using `Red Hat Subscription-Manager`
```

MANAGING SUPPORT CASES WITH RED HAT SUPPORT TOOL

One benefit of a product subscription is access to technical support through the Red Hat Customer Portal. Depending on the system's subscription support level, Red Hat may be contacted through online tools or by phone. See `https://access.redhat.com/site/support/policy/support_process` for detailed information.

Preparing a Bug report

Before contacting Red Hat Support, it is important to gather relevant information for a bug report.

Define the problem. Be able to clearly state the problem and its symptoms. Be as specific as possible. Detail the steps that will reproduce the problem.

Gather background information. Which product and version is affected? Be ready to provide relevant diagnostic information. This can include output of **`sosreport`**, discussed later in this section. For kernel problems, this could include the system's **`kdump`** crash dump or a digital photo of the kernel backtrace 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 should be followed by a phone call to the relevant local support center (see `https://access.redhat.com/site/support/contact/technicalSupport`).

SEVERITY	DESCRIPTION
Urgent ***(Severity 1)***	A problem that severely impacts your use of the software in a production environment. This 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 use in a production environment is severely reduced. The situation is causing a high impact to 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. For production environments, there is a medium to low impact on your business. Business continues to function using a procedural workaround. For development environments, the situation is causing problems 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. For production environments, there is low to no impact on your business or the performance or functionality of your system. For development environments, there is a medium to low impact on your business, but your business continues to function using a procedural workaround.

Managing a Bug Report with `redhat-support-tool`

You can create, view, modify, and close Red Hat Support cases using **`redhat-support-tool`**. When support cases are in an **`opened`** or **`maintained`** status, users may attach files or documentation, such as diagnostic reports (sosreport). The tool uploads and attaches files to cases.

Case details including the product name, version, summary, description, severity, and case group may be assigned with command options or letting the tool prompt for required information. In the following example, a new case is opened. The **`--product`** and **`--version`** options are specified.

```
[user@host ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): opencase --product="Red Hat Enterprise Linux" --
version="7.0"
Please enter a summary (or 'q' to exit): System fails to run without power
Please enter a description (Ctrl-D on an empty line when complete):
When the server is unplugged, the operating system fails to continue.
 1   Urgent
 2   High
```

```
 3   Normal
 4   Low
Please select a severity (or 'q' to exit): 4
Would you like to assign a case group to this case (y/N)? N
Would see if there is a solution to this problem before opening a support case?
 (y/N) N
--------------------------------------------------------------------------------
Support case 01034421 has successfully been opened.
```

If the **--product** and **--version** options are not specified the **redhat-support-tool** provides a list of choices for those options.

```
[user@host ~]$ redhat-support-tool
Welcome to the Red Hat Support Tool.
Command (? for help): opencase
Do you want to use the default product - "Red Hat Enterprise Linux" (y/N)?: y
...output omitted...
29  7.4
30  7.5
31  7.6
32  8.0 Beta
Please select a version (or 'q' to exit): 32
Please enter a summary (or 'q' to exit): yum fails to install apache
Please enter a description (Ctrl-D on an empty line when complete):
yum cannot find correct repo
 1   Urgent
 2   High
 3   Normal
 4   Low
Please select a severity (or 'q' to exit): 4
Would you like to use the default (Ungrouped Case) Case Group (y/N)? : y
Would you like to see if there's a solution to this problem before opening a
 support case? (y/N) N
--------------------------------------------------------------------------------
Support case 010355678 has successfully been opened.
```

Attaching Diagnostic Information to a Support Case

Including diagnostic information can lead to a quicker resolution. Attach the *sosreport* when the case is opened. The **sosreport** command generates a compressed tar archive of diagnostic information gathered from the running system. The **redhat-support-tool** prompts to include one if an archive has been created previously:

```
Please attach a SoS report to support case 01034421. Create a SoS report as
the root user and execute the following command to attach the SoS report
directly to the case:
 redhat-support-tool addattachment -c 01034421 path to sosreport

Would you like to attach a file to 01034421 at this time? (y/N) N
Command (? for help):
```

If a current SoS report does not exist, an administrator can generate and attach one later. Use the **redhat-support-tool addattachment** command to attach the report.

Support cases can also be viewed, modified, and closed by the subscriber:

```
Command (? for help): listcases

Type the number of the case to view or 'e' to return to the previous menu.
 1 [Waiting on Red Hat]  System fails to run without power
No more cases to display
Select a Case: 1

Type the number of the section to view or 'e' to return to the previous menu.
 1 Case Details
 2 Modify Case
 3 Description
 4 Recommendations
 5 Get Attachment
 6 Add Attachment
 7 Add Comment
End of options.
Option: q

Select a Case: q

Command (? for help):q

[user@host ~]$ redhat-support-tool modifycase --status=Closed 01034421
Successfully updated case 01034421
```

The Red Hat Support Tool has advanced application diagnostic and analytic capabilities. Using kernel crash dump core files, **`redhat-support-tool`** can create and extract a *backtrace*. The kernel crash dump core file is created using the **`kdump`** command. A backtrace is a report of the active stack frames at the point of a crash dump and provides onsite diagnostics. One of the options of the **`redhat-support-tool`** is to open a support case.

The tool also provides log file analysis. Using the tool's **`analyze`** command, log files of many types, including operating system, JBoss, Python, Tomcat, and oVirt, can be parsed to recognize problem symptoms. The log files can be viewed and diagnosed individually. Providing preprocessed analysis, as opposed to raw data such as crash dump or log files, allows support cases to be opened and made available to engineers more quickly.

JOINING RED HAT DEVELOPER

One other useful resource available from Red Hat is Red Hat Developer. Hosted at https://developer.redhat.com, this program provides subscription entitlements to Red Hat software for development purposes, documentation, and premium books from our experts on microservices, serverless computing, Kubernetes, and Linux. A blog, links to information about upcoming events and training, and other help resources are also available, as well as links to Red Hat Customer Portal.

Registration is free, and can be completed at https://developer.redhat.com/register.

REFERENCES

`sosreport`(1) man page

Red Hat Access: Red Hat Support Tool
https://access.redhat.com/site/articles/445443

Red Hat Support Tool First Use
https://access.redhat.com/site/videos/534293

Contacting Red Hat Technical Support
https://access.redhat.com/site/support/policy/support_process/

Help - Red Hat Customer Portal
https://access.redhat.com/site/help/

GUIDED EXERCISE

GETTING HELP FROM RED HAT CUSTOMER PORTAL

In this exercise, you will generate a diagnostics report using Web Console.

OUTCOMES

You should be able to generate a diagnostics report using Web Console which could be submitted to Red Hat Customer Portal as part of a support case.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab support-portal start`** command. The command runs a start script that determines if `servera` is reachable on the network. It also starts and enables Web Console on servera.

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

1. From `workstation` use the **ssh** command to log into `servera` as the `student` user.

```
[student@workstation ~]$ ssh student@servera
Web console: https://servera.lab.example.com:9090/ or
 https://172.25.250.10:9090/
[student@servera ~]$
```

2. Use the **`systemctl`** command to confirm that the `cockpit` service is running. Enter `student` as the password when prompted.

```
[student@servera ~]$ sudo systemctl status cockpit.socket
[sudo] password for student: student
● cockpit.socket - Cockpit Web Service Socket
   Loaded: loaded (/usr/lib/systemd/system/cockpit.socket; enabled; vendor
 preset: disabled)
   Active: active (listening) since Thu 2019-05-16 10:32:33 IST; 4min 37s
 ago
     Docs: man:cockpit-ws(8)
   Listen: [::]:9090 (Stream)
  Process: 676 ExecStartPost=/bin/ln -snf active.motd /run/cockpit/motd
 (code=exited, status=0/SUCCESS)
  Process: 668 ExecStartPost=/usr/share/cockpit/motd/update-motd  localhost
 (code=exited, status=0/SUCCESS)
```

```
 Tasks: 0 (limit: 11405)
Memory: 1.5M
CGroup: /system.slice/cockpit.socket
```

3. Log out from `servera`.

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

4. On `workstation`, open *Firefox* and log in to the Web Console interface running on `servera.lab.example.com` as the `root` user with `redhat` as the password.

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

   4.2. If prompted, accept the self-signed certificate by adding it as an exception.

   4.3. Log in as the `root` user with `redhat` as the password. You are now logged in as a privileged user, which is necessary to create a diagnostic report.

   4.4. Click ***Diagnostic Reports*** in the left navigation bar. Click on ***Create Report***. The report takes a few minutes to create.

5. When the report is ready, click on ***Download report***. Save the file.

   5.1. Click the ***Download report*** button, followed by the ***Save File*** button.

   5.2. Click the ***Close*** button.

   5.3. Log out from the Web Console interface.

Finish

On `workstation`, run the **`lab support-portal finish`** script to complete this exercise.

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

This concludes the guided exercise.

DETECTING AND RESOLVING ISSUES WITH RED HAT INSIGHTS

OBJECTIVES

After completing this section, you should be able to use Red Hat Insights to analyze servers for issues, remediate or resolve them, and confirm the solution worked.

INTRODUCING RED HAT INSIGHTS

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

Red Hat regularly updates the knowledge base used by Red Hat Insights, based on common support risks, security vulnerabilities, known-bad configurations, and other issues identified by Red Hat. Actions to mitigate or remediate these issues are validated and verified by Red Hat. This allows you to proactively identify, prioritize, and resolve issues before they become a larger problem.

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

Red Hat Insights recommendations are tailored to each system registered to the service. You install each client system with an agent that collects metadata about the runtime configuration of the system. This data is a subset of what you might provide to Red Hat Support using **`sosreport`** in order to resolve a support ticket. You can limit or obfuscate the data that your clients send. This will block some of the analytic rules from operating, depending on what you limit.

Almost immediately 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 Red Hat Insights console in 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 4 and later
- Red Hat OpenShift Container Platform
- Red Hat OpenStack Platform 7 and later

Describing the Red Hat Insights Architecture

You may register a system to Red Hat Insights through the Red Hat Cloud Portal. When you register the system, it provides Red Hat Insights with metadata about its current configuration. This data is sent to Red Hat Insights using TLS encryption to protect it in transit. It is also anonymized before it sent.

Based on the recommendations provided by the Red Hat Insights rule engine, the results of the analysis are displayed on the Red Hat Insights console in Red Hat Cloud Portal at https://cloud.redhat.com/insights.

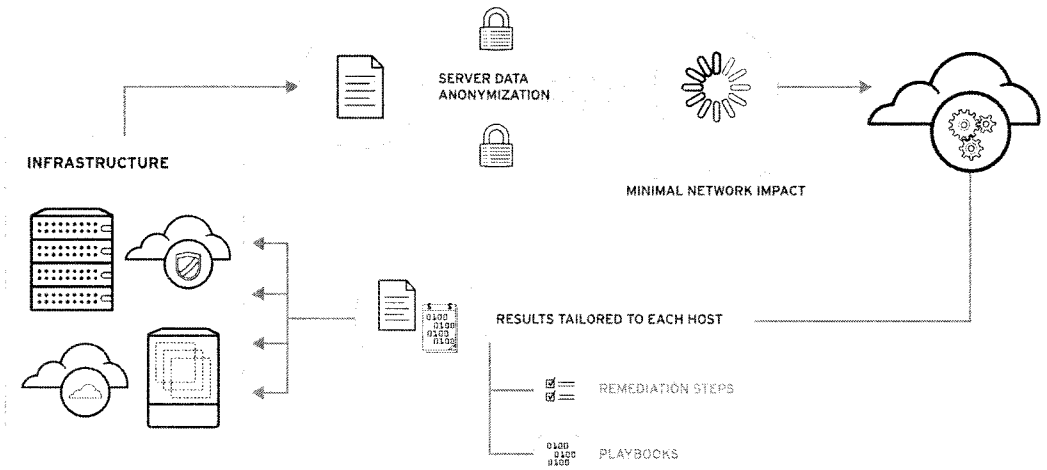


Figure 1.6: Red Hat Insights high-level architecture

INSTALLING RED HAT INSIGHTS CLIENTS

Red Hat Insights is included with Red Hat Enterprise Linux 8 as part of the subscription. Older version of Red Hat Enterprise Linux servers require to install the *insights-client* package on the system.

IMPORTANT

The *insights-client* package replaces the older *redhat-access-insights* package starting with Red Hat Enterprise Linux 7.5.

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

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

The Insights client periodically updates the metadata provided to Red Hat Insights. Use the **`insights-client`** command to refresh the client's metadata at any time.

```
[root@demo ~]# insights-client
Starting to collect Insights data for demo.lab.example.com
Uploading Insights data.
Successfully uploaded report from 773b351b-dfb1-4393-afa8-915cc2875e06 to
 account XXXXX.
```

Registering a RHEL System with Red Hat Insights

To register a RHEL server to Red Hat Insights, the overall process is as follows:

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

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

A valid entitlement for Red Hat Insights must be attached to the system, which you might receive as part of the Red Hat Enterprise Linux subscription.

2. Make sure that the *insights-client* package is installed on the system. In RHEL 7, this package is in the `rhel-7-server-rpms` channel.

NOTE
This step is not required on Red Hat Enterprise Linux 8 systems.

```
[root@demo ~]# yum install insights-client
```

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

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

4. Verify that the system is visible at https://cloud.redhat.com/insights.

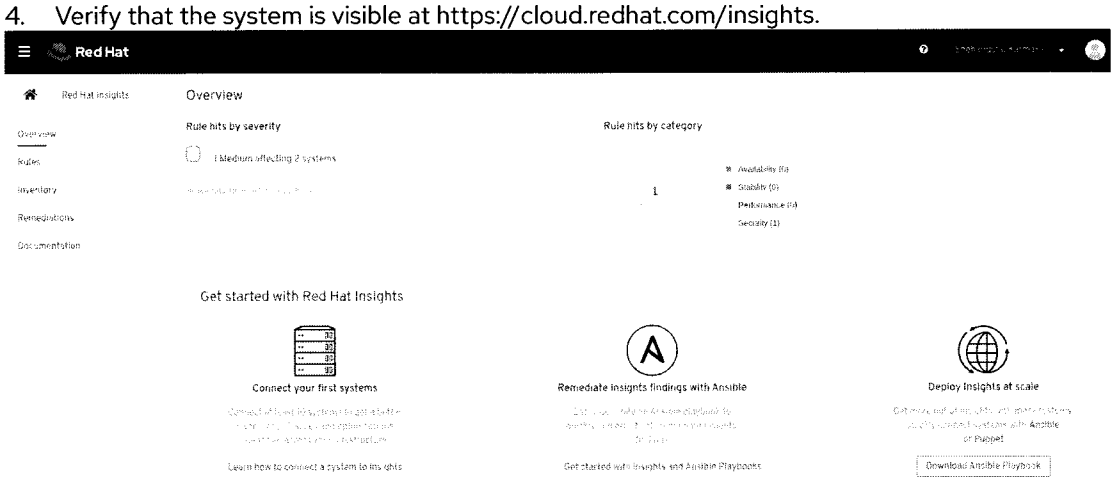


Figure 1.7: Red Hat Insights overview on the Cloud Portal

VIEWING REPORTS PROVIDED BY RED HAT INSIGHTS

A Red Hat Insights report shows the state of a system over time. With these reports, you can easily view current assessments of risk and identify historical trends in order to enhance your decision making.

The Red Hat Insights interface provides you with information that includes:

- A current overall risk score based on your registered systems.
- Recommended actions to take on your systems, which is further broken down into categories and severities.
- Information on when systems last checked in with Red Hat Insights.
- Issues that need to be prioritized based on their impact.

Navigating the Red Hat Insights Console

The Red Hat Insights console on the Cloud Portal provides the following pages:

Overview

The ***Overview*** page provides a view of current risks to the registered infrastructure. The ***Overview*** provides a view to investigate how a specific rule is affecting the registered systems, or see all the rules that pose risk to a selected system.

The page allows you to view rules based on severity and classify the infrastructure risk based on category. Each rule is categorized based on the potential impact on one of the following areas of operations: **`Availabilty`**, **`Stability`**, **`Performance`**, and **`Security`**

Rules

The ***Rules*** page provides list of Insights rules and the affected hosts .

In the ***Rules*** page, you will notice some of these issues are marked with check mark under the ***Ansible*** logo column. This indicates the issue has an Ansible remediation playbook available. The issues with out a check mark, do not have an Ansible remediation playbook, but may have manual mitigation or remediation instructions in the issue details.

You can click on the name of the rule to view all the affected systems. Each issue provides a description of how the issue may be manifested on the system, and ***Remediate with Ansible*** to create a Playbook for remediation.

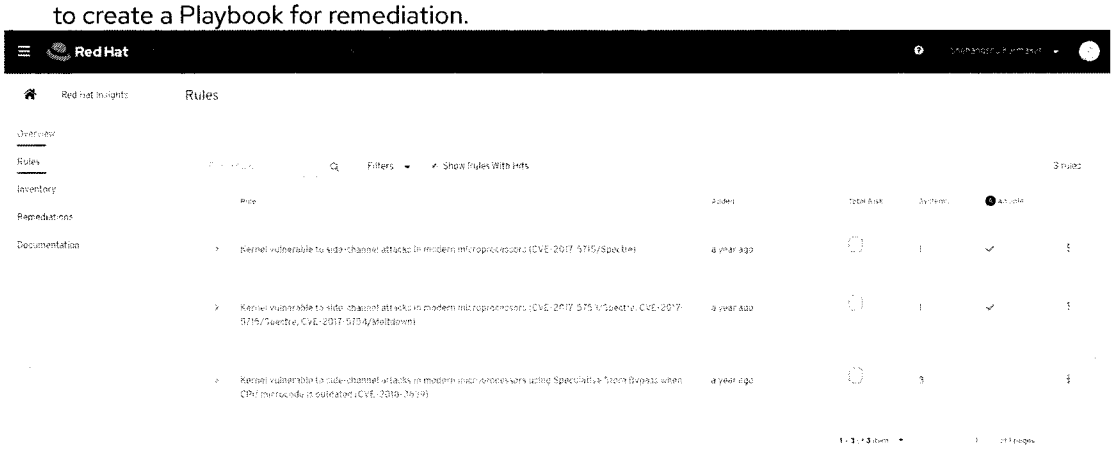


Figure 1.8: Rules page in Red Hat Insights console

Inventory

The ***Inventory*** page provides a list of the systems you have registered with Red Hat Insights.

You can easily filter the inventory for specific systems. The ***Last Sync*** column displays the time of the most recent metadata update for each system.

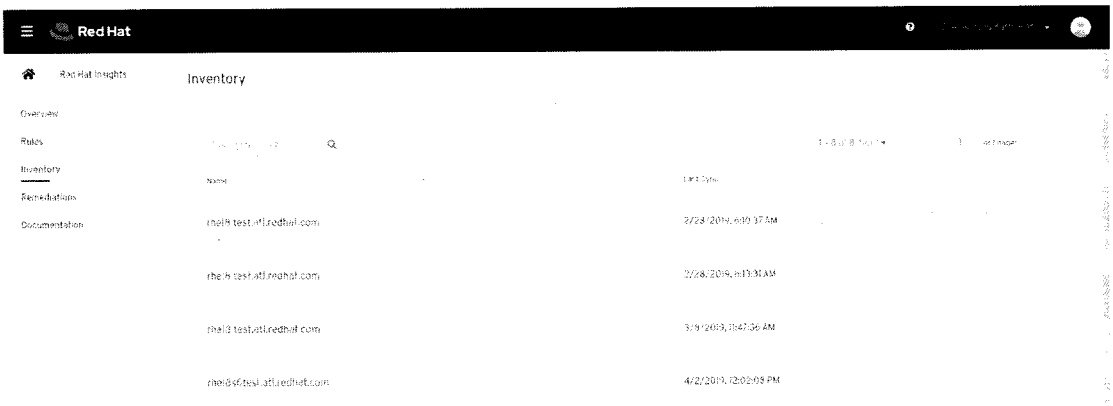


Figure 1.9: Inventory page in Red Hat Insights console

Remediations

The ***Remediations*** page provides a list of Ansible Playbooks created and allows those Playbooks to be downloaded.

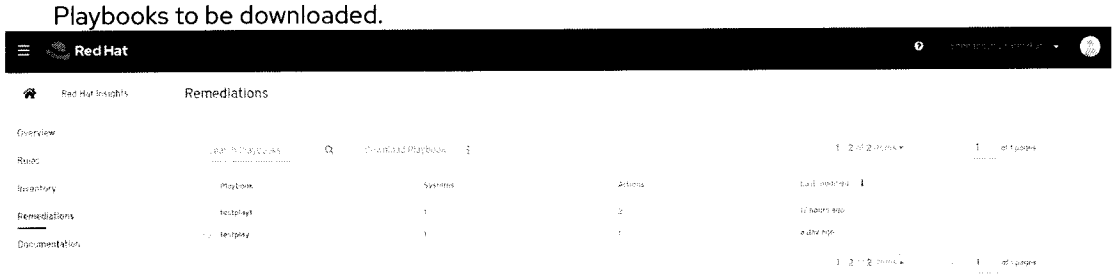


Figure 1.10: Remediations page in Red Hat Insights console

Viewing Issues Reported by Red Hat Insights

To view issues reported by Red Hat Insights, the overall process is as follows:

1. Log in to Red Hat Cloud Portal and access the Red Hat Insights page at https://cloud.redhat.com/insights.
2. On the portal, navigate to the ***Overview*** page.
3. Select ***Rule hits by severity*** to view rules by the ***Total Risk*** they pose to the registered infrastructure. Alternatively, select ***Rule hits by category*** to view the type of risk based on category.
4. Scroll through the list of rules to see high-level information about risk, systems exposed, and availability of Ansible Playbook to automate the remediation.
5. Click on a rule to see the more description of the rule, click the link to read the relevant Knowledgebase articles, and view the list of all the affected hosts.
6. Click on a host to see specific information about detected issues and steps to resolve the issue.

INTERPRETING RED HAT INSIGHTS REPORTS

In Red Hat Insights, its rules determine the issues it looks for on your systems. Red Hat frequently adds new rules to Red Hat Insights to check for newly identified issues. Rules may look for incidents that have occurred on your system that indicate a problem, or they may proactively anticipate problems based on your system's current configuration.

When a rule matches your system, indicating that an issue exists, additional information is provided with the rule to help you understand the issue, prioritize work to address it, determine what mitigation or remediation is available, and to help you automate its resolution.

Each rule is categorized by type, and has a summary name and a longer description to explain what the issue is. Rules are normally linked to Knowledgebase articles on the Customer Portal with additional information. The Knowledgebase article may provide information about different ways to mitigate or remediate a problem, and Ansible Playbooks or other materials may be provided by the rule to help automate mitigation and remediation.

Some problems are complex to address, and a complete fix may require a reboot or downtime. In that case, there may be options to mitigate the issues by reducing its risk as a temporary measure. The rule will provide scores of the risk presented by the issue, in several different categories.

For example, consider a security issue that requires an update to the kernel packages and a reboot to fix, but which also can be made very hard to exploit through certain temporary configuration changes. You might choose to apply the temporary changes immediately, and defer the reboot until you can schedule an emergency maintenance window.

Red Hat Insights categorizes the risk that an issue presents to your system into four categories. It evaluates the level of that risk using these levels: **`Low`**, **`Moderate`**, **`Important`**, and **`Critical`**.

The **`Likelihood`**, **`Impact`**, **`Total Risk`**, and **`Risk of Change`** categories forecast the risk factors of an issue detected on your subscribed systems.

`Impact`
: Indicates the predicted level of impact to the system by this issue.

`Likelihood`
: Indicates the likelihood that a given issue will impact the system.

`Total Risk`
: Indicates the impact of security issues found in Red Hat products using a four-point scale (Low, Moderate, Important, and Critical), as well as Common Vulnerability Scoring System (CVSS) base scores. These provide a prioritized risk assessment to help you to take informed decisions on the risk each issue places on your infrastructure.

`Risk of Change`
: Indicates the risk that the recommended remediation action could cause disruption to a system.

To view the risk factors forecast by various rules in Red Hat Insights, navigate to ***Overview***, or to the ***Rules*** page. Each rule shows the icon for ***Total Risk***, and ***Risk of Change***.

Rules > Kernel Vulnerable To Side-Channel Attacks In Modern Microprocessors (CVE-2017-5715/Spectre)

Kernel vulnerable to side-channel attacks in modern microprocessors (CVE-2017-5715/Spectre)

Publish Date: 1/17/2018

A vulnerability was discovered in modern microprocessors supported by the kernel, whereby an unprivileged attacker can use this flaw to bypass restrictions to gain read access to privileged memory. The issue was reported as CVE-2017-5715 / Spectre.

Knowledgebase Article

Total Risk

The **likelihood** that this will be a problem is Medium. The **impact** of the problem would be High if it occurred.

Risk of Change

These will likely require an outage window.

Reboot Required

Affected systems

Remediate with Ansible

1 - 1 of 1 items 1 of 1 pages

Name	Last Sync
servera.lab.example.com	5/14/2019, 12:01:41 PM

1 - 1 of 1 items 1 of 1 pages

Figure 1.11: Red Hat Insights rules that apply to a host

When you have identified the issues you want to address, you can address them manually or automatically. After the issue has been resolved, and your Red Hat Insights client has uploaded new metadata, the rule should no longer match that system and the issue should disappear from the list of recommended actions.

Manually Remediating an Issue Reported by Red Hat Insights

To manually remediate an issue reported by Red Hat Insights, the overall process is as follows:

1. Log in to Red Hat Cloud Portal and access the Red Hat Insights page at https://cloud.redhat.com/insights.

   Navigate to the ***Rules*** page. Click on the name of the rule to resolve.

2. Scroll down to ***Affected systems*** to see all the affected systems due to the rule.

3. Click on one of the affected systems link under the ***Name*** column. The page shows description of how the issue can affect the system, and the steps to resolve the issue on the system. Follow the instructions in ***Steps to resolve*** to remediate the issue on the system.

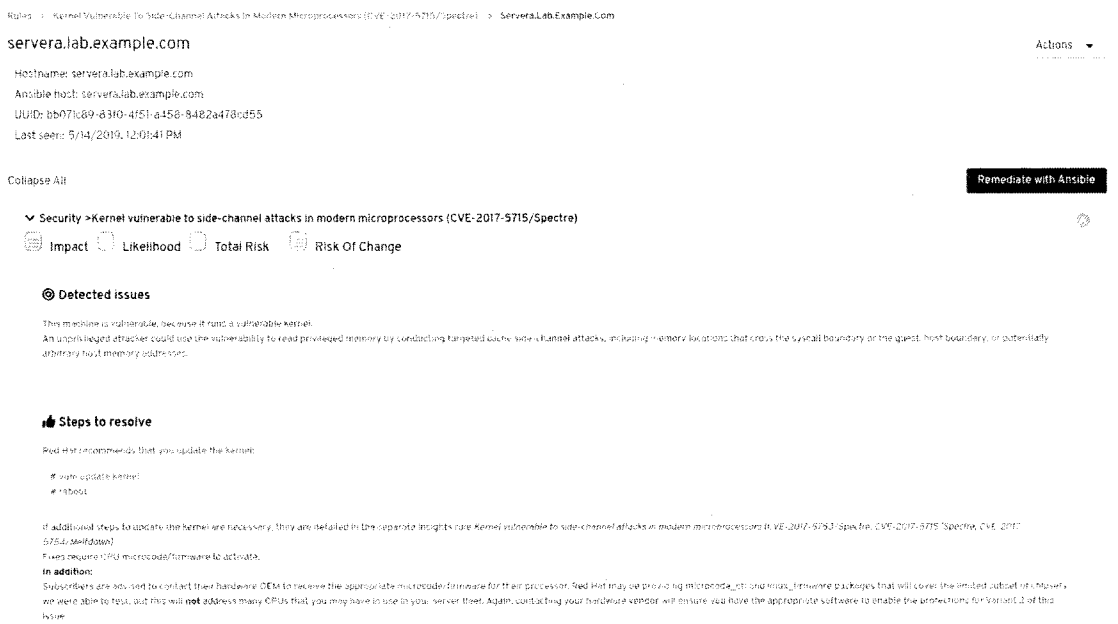


Figure 1.12: Remediating rules on systems manually

4. Perform the steps on the affected system to resolve the issue.

5. After applying the remediation steps, run the following command as `root` on the system to report the changes to Red Hat Insights:

```
[root@demo ~]# insights-client
Starting to collect Insights data for demo.lab.example.com
Uploading Insights data.
Successfully uploaded report from 773b351b-dfb1-4393-afa8-915cc2875e06 to
 account xxxxxx.
```

6. In the Red Hat Insights console, navigate to the ***Rules*** page. Click rule and scroll down to ***Affected systems*** and verify that the issue no longer appears in the list of affected systems.

REFERENCES

`insights-client`(8) and **`insights-client.conf`**(5) man pages.

For more information, refer to the *GET STARTED* chapter in the *Red Hat Insights 1.0 Getting Started Guide* at
https://access.redhat.com/products/red-hat-insights/#getstarted

More information on feature updates for Red Hat Insights is at
https://access.redhat.com/documentation/en-us/red_hat_insights/1.0/html-single/release_notes/#release_information

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

Information on how to exclude data collected by Red Hat Insights is available at
Opting Out of Sending Metadata from Red Hat Insights Client
https://access.redhat.com/articles/2025273

QUIZ

DETECTING AND RESOLVING ISSUES WITH RED HAT INSIGHTS

Choose the correct answers to the following questions:

1. **In what order do the following events occur when managing a Red Hat Enterprise Linux system using 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. Red Hat Insights 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 is used 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. **Which two pages in the Red Hat Insights console allows you to display list of rules, using filters based on the catgeory of risk? (Choose two.)**
   a. ***Overview***
   b. ***Inventory***
   c. ***Rules***
   d. ***Remediation***

SOLUTION

DETECTING AND RESOLVING ISSUES WITH RED HAT INSIGHTS

Choose the correct answers to the following questions:

1. **In what order do the following events occur when managing a Red Hat Enterprise Linux system using 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. Red Hat Insights 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 is used 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. **Which two pages in the Red Hat Insights console allows you to display list of rules, using filters based on the catgeory of risk? (Choose two.)**
   a. ***Overview***
   b. *Inventory*
   c. ***Rules***
   d. *Remediation*

SUMMARY

In this chapter, you learned:

- 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.
- 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.
- Red Hat Customer Portal provides you with access to documentation, downloads, optimization tools, support case management, and subscription and entitlement management for your Red Hat products.
- **`redhat-support-tool`** is a command-line tool to query Knowledgebase and work with support cases from the server's command line.
- Red Hat Insights is a SaaS-based predictive analytics tool to help you identify and remediate threats to your systems' security, performance, availability, and stability.

CHAPTER 2

NAVIGATING FILE SYSTEMS

GOAL	Copy, move, create, delete, and organize files while working from the Bash shell.
OBJECTIVES	• Describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy. • Create, copy, move, and remove files and directories. • Make multiple file names reference the same file using hard links and symbolic (or "soft") links.
SECTIONS	• Describing Linux File-system Hierarchy Concepts (and Quiz) • Managing Files Using Command-line Tools (and Guided Exercise) • Making Links Between Files (and Guided Exercise)

DESCRIBING LINUX FILE SYSTEM HIERARCHY CONCEPTS

OBJECTIVES

After completing this section, you should be able to describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.

THE FILE-SYSTEM HIERARCHY

All files on a Linux system are stored on file systems, which are organized into a single *inverted* tree of directories, known as a *file-system hierarchy*. This tree is inverted because the root of the tree is said to be at the *top* of the hierarchy, and the branches of directories and subdirectories stretch *below* the root.

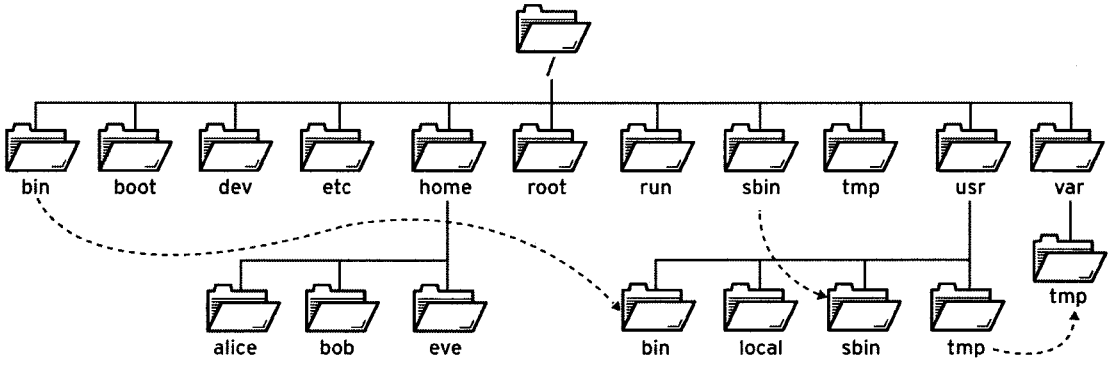


Figure 2.1: Significant file-system directories in Red Hat Enterprise Linux 8

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, you could refer to that directory as **`/etc`**. Likewise, if the **`/etc`** directory contained a file named **`issue`**, you could refer to that file as **`/etc/issue`**.

Subdirectories of **/** are used for standardized purposes to organize files by type and purpose. This makes it easier to find files. For example, in the root directory, the subdirectory **`/boot`** is used for storing files needed 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 may be modified or appended by active processes.
- *persistent* content remains after a reboot, like configuration settings.
- *runtime* content is process- or system-specific content that is deleted by a reboot.

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

Important Red Hat Enterprise Linux Directories

LOCATION	PURPOSE
/usr	Installed software, shared libraries, include files, and read-only program data. Important subdirectories include: • **/usr/bin**: User commands. • **/usr/sbin**: System administration commands. • **/usr/local**: Locally customized software.
/etc	Configuration files specific to this system.
/var	Variable data specific to this system that should persist between boots. Files that dynamically change, such as databases, cache directories, log files, printer-spooled documents, and website content may be found under **/var**.
/run	Runtime data for processes started since the last boot. This includes process ID files and lock files, among other things. The contents of this directory are recreated on reboot. This directory consolidates **/var/run** and **/var/lock** from earlier versions of Red Hat Enterprise Linux.
/home	*Home directories* are where regular users store their personal data and configuration files.
/root	Home directory for the administrative superuser, `root`.
/tmp	A world-writable space for temporary files. Files which have not been accessed, changed, or modified for 10 days are deleted from this directory automatically. Another temporary directory exists, **/var/tmp**, in which files that have not been accessed, changed, or modified in more than 30 days are deleted automatically.
/boot	Files needed in order to start the boot process.
/dev	Contains special *device files* that are used by the system to access hardware.

IMPORTANT

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

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

In earlier versions of Red Hat Enterprise Linux, these were distinct directories containing 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

The UsrMove feature page from Fedora 17
https://fedoraproject.org/wiki/Features/UsrMove

QUIZ

DESCRIBING 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 temporary files?
   a. `/tmp`
   b. `/trash`
   c. `/run`
   d. `/var`

5. Which directory contains dynamic data, such as for databases and websites?
   a. `/etc`
   b. `/run`
   c. `/usr`
   d. `/var`

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

DESCRIBING 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 temporary files?**
   a. **`/tmp`**
   b. `/trash`
   c. `/run`
   d. `/var`

5. **Which directory contains dynamic data, such as for databases and websites?**
   a. `/etc`
   b. `/run`
   c. `/usr`
   d. **`/var`**

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`

MANAGING FILES USING COMMAND-LINE TOOLS

OBJECTIVES

After completing this section, you should be able to create, copy, move, and remove files and directories.

COMMAND-LINE FILE MANAGEMENT

To manage files, you need to be able to create, remove, copy, and move them. You also need to organize them logically into directories, which you also need to be able to create, remove, copy, and move.

The following table summarizes some of the most common file management commands. The remainder of this section will discuss ways to use these commands in more detail.

Common file management commands

ACTIVITY	COMMAND SYNTAX
Create a directory	**`mkdir`** ***`directory`***
Copy a file	**`cp`** ***`file new-file`***
Copy a directory and its contents	**`cp -r`** ***`directory new-directory`***
Move or rename a file or directory	**`mv`** ***`file new-file`***
Remove a file	**`rm`** ***`file`***
Remove a directory containing files	**`rm -r`** ***`directory`***
Remove an empty directory	**`rmdir`** ***`directory`***

Creating Directories

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

The **`mkdir`** command will fail with an error if the directory already exists, or if you are trying to create a subdirectory in a directory that does not exist. The **`-p`** (*parent*) option creates missing parent directories for the requested destination. Use the **`mkdir -p`** command with caution, because spelling mistakes can create unintended directories without generating error messages.

In the following example, pretend that you are trying to create a directory in the **`Videos`** directory named **`Watched`**, but you accidentally left off 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 failed because **Videos** was misspelled and the directory **Video** does not exist. If you had used the **mkdir** command with the **-p** option, the directory **Video** would be created, which was not what you had intended, and the subdirectory **Watched** would be created in that incorrect directory.

After correctly spelling the **Videos** parent directory, creating the **Watched** subdirectory will succeed.

```
[user@host ~]$ mkdir Videos/Watched
[user@host ~]$ ls -R Videos
Videos/:
blockbuster1.ogg  blockbuster2.ogg  Watched

Videos/Watched:
```

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
[user@host Documents]$ ls
ProjectX  ProjectY
```

Use the **mkdir -p** command and space-delimited relative paths for each of the subdirectory names to create multiple parent directories with subdirectories.

```
[user@host Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapter2 Thesis/Chapter3
[user@host Documents]$ cd
[user@host ~]$ ls -R Videos Documents
Documents:
ProjectX  ProjectY  Thesis

Documents/ProjectX:

Documents/ProjectY:

Documents/Thesis:
Chapter1  Chapter2  Chapter3

Documents/Thesis/Chapter1:

Documents/Thesis/Chapter2:

Documents/Thesis/Chapter3:

Videos:
blockbuster1.ogg  blockbuster2.ogg  Watched

Videos/Watched:
```

The last **mkdir** command created three Chapter*N* subdirectories with one command. The **-p** option created the missing parent directory **Thesis**.

Copying Files

The **cp** command copies a file, creating a new file either in the current directory or in a specified directory. It can also copy multiple files to a directory.

WARNING
If the destination file already exists, the **cp** command overwrites the file.

```
[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
drwxrwxr-x. 2 user user 4096 Feb  8 16:05 Watched
[user@host Videos]$
```

When copying multiple files with one command, the last argument must be a directory. Copied files retain their original names in the new directory. If a file with the same name exists in the target directory, the existing file is overwritten. By default, the **cp** does not copy directories; it ignores them.

In the following example, two directories are listed, **Thesis** and **ProjectX**. Only the last argument, **ProjectX** is valid as a destination. The **Thesis** directory is ignored.

```
[user@host Videos]$ cd ../Documents
[user@host Documents]$ cp thesis_chapter1.odf thesis_chapter2.odf Thesis ProjectX
cp: omitting directory `Thesis'
[user@host Documents]$ ls Thesis ProjectX
ProjectX:
thesis_chapter1.odf  thesis_chapter2.odf

Thesis:
Chapter1  Chapter2  Chapter3
```

In the first **cp** command, the **Thesis** directory failed to copy, but the **thesis_chapter1.odf** and **thesis_chapter2.odf** files succeeded.

If you want to copy a file to the current working directory, you can use the special **.** directory:

```
[user@host ~]$ cp /etc/hostname .
[user@host ~]$ cat hostname
host.example.com
[user@host ~]$
```

Use the copy command with the **-r** (*recursive*) option, to copy the **Thesis** directory and its contents to the **ProjectX** directory.

```
[user@host Documents]$ cp -r Thesis ProjectX
[user@host Documents]$ ls -R ProjectX
ProjectX:
Thesis  thesis_chapter1.odf  thesis_chapter2.odf

ProjectX/Thesis:
Chapter1  Chapter2  Chapter3

ProjectX/Thesis/Chapter1:

ProjectX/Thesis/Chapter2:
thesis_chapter2.odf

ProjectX/Thesis/Chapter3:
```

Moving Files

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

Use the **mv** command to *rename* a file.

```
[user@host Videos]$ cd ../Documents
[user@host Documents]$ ls -l thesis*
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter1.odf
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter2.odf
[user@host Documents]$ mv thesis_chapter2.odf thesis_chapter2_reviewed.odf
[user@host Documents]$ ls -l thesis*
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter1.odf
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter2_reviewed.odf
```

Use the **mv** command to *move* a file to a different directory.

```
[user@host Documents]$ ls Thesis/Chapter1
[user@host Documents]$
[user@host Documents]$ mv thesis_chapter1.odf Thesis/Chapter1
[user@host Documents]$ ls Thesis/Chapter1
thesis_chapter1.odf
[user@host Documents]$ ls -l thesis*
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter2_reviewed.odf
```

Removing Files and Directories

The **rm** command removes files. By default, **rm** will not remove directories that contain files, unless you add the **-r** or **--recursive** option.

IMPORTANT
There is no command-line undelete feature, nor a "trash bin" from which you can restore files staged for deletion.

It is a good idea to verify your current working directory before removing a file or directory.

```
[user@host Documents]$ pwd
/home/student/Documents
```

Use the **rm** command to remove a single file from your working directory.

```
[user@host Documents]$ ls -l thesis*
-rw-rw-r--. 1 user user 0 Feb  6 21:16 thesis_chapter2_reviewed.odf
[user@host Documents]$ rm thesis_chapter2_reviewed.odf
[user@host Documents]$ ls -l thesis*
ls: cannot access 'thesis*': No such file or directory
```

If you attempt to use the **rm** command to remove a directory without using the **-r** option, the command will fail.

```
[user@host Documents]$ rm Thesis/Chapter1
rm: cannot remove `Thesis/Chapter1': Is a directory
```

Use the **rm -r** command to remove a subdirectory and its contents.

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

Thesis/Chapter1:
thesis_chapter1.odf

Thesis/Chapter2:
thesis_chapter2.odf

Thesis/Chapter3:
[user@host Documents]$ rm -r Thesis/Chapter1
[user@host Documents]$ ls -l Thesis
total 8
drwxrwxr-x. 2 user user 4096 Feb 11 12:47 Chapter2
drwxrwxr-x. 2 user user 4096 Feb 11 12:48 Chapter3
```

The **rm -r** command traverses each subdirectory first, individually removing their files before removing each directory. You can use the **rm -ri** command to interactively prompt for confirmation before deleting. This is essentially the opposite of using the **-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.odf'? y
rm: remove directory `Thesis/Chapter2'? y
rm: remove directory `Thesis/Chapter3'? y
rm: remove directory `Thesis'? y
[user@host Documents]$
```

WARNING

If you specify both the **-i** and **-f** options, the **-f** option takes priority and you will not be prompted for confirmation before **rm** deletes files.

In the following example, the **rmdir** command only removes the directory that is empty. Just like the earlier example, you must use the **rm -r** command to remove a directory that contains content.

```
[user@host Documents]$ pwd
/home/student/Documents
[user@host Documents]$ rmdir ProjectY
[user@host Documents]$ rmdir ProjectX
rmdir: failed to remove `ProjectX': Directory not empty
[user@host Documents]$ rm -r ProjectX
[user@host Documents]$ ls -lR
.:
total 0
```

NOTE

The **rm** command with no options cannot remove an empty directory. You must use the **rmdir** command, **rm -d** (which is equivalent to **rmdir**), or **rm -r**.

REFERENCES

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

GUIDED EXERCISE

MANAGING FILES USING COMMAND-LINE TOOLS

In this exercise you will create, organize, copy, and remove files and directories.

OUTCOMES

You should be able to create, organize, copy, and remove files and directories.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab files-manage start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network.

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

1. Use the **`ssh`** command to log in to `servera` 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 ~]$
```

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
```

3. Continuing in the `student` user's home directory, use the **`touch`** command to create sets of empty practice files to use during this lab.

   - 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`**.

   In each set, replace X with the numbers 1 through 6.

```
[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-rw-r--. 1 student student 0 Feb  4 18:23 film1.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film2.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film3.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film4.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film5.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film6.avi
drwxrwxr-x. 2 student student 6 Feb  4 18:23 Music
drwxrwxr-x. 2 student student 6 Feb  4 18:23 Pictures
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap1.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap2.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap3.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap4.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap5.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap6.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song1.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song2.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song3.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song4.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song5.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song6.mp3
drwxrwxr-x. 2 student student 6 Feb  4 18:23 Videos
```

4. Continuing in the `student` user's home directory, move the song files to the **`Music`** subdirectory, the snapshot files to the **`Pictures`** subdirectory, and the movie files to the **`Videos`** subdirectory.

   When distributing files from one location to many locations, first change to the directory containing the *source* files. Use the simplest path syntax, absolute or relative, to reach the destination for each file management task.

```
[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-rw-r--. 1 student student 0 Feb  4 18:23 song1.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song2.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song3.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song4.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song5.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:23 song6.mp3

Pictures:
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap1.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap2.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap3.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap4.jpg
```

```
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap5.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:23 snap6.jpg

Videos:
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film1.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film2.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film3.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film4.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film5.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:23 film6.avi
```

5. Continuing in the `student` user's home directory, create three subdirectories for organizing your files into projects. Name the subdirectories **`friends`**, **`family`**, and **`work`**. Use a single command to create all three subdirectories at the same time.

   You will use these directories to rearrange your files into projects.

```
[student@servera ~]$ mkdir friends family work
[student@servera ~]$ ls -l
total 0
drwxrwxr-x. 2 student student   6 Feb  4 18:38 family
drwxrwxr-x. 2 student student   6 Feb  4 18:38 friends
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos
drwxrwxr-x. 2 student student   6 Feb  4 18:38 work
```

6. Copy a selection of new files to the project directories **`family`** and **`friends`**. Use as many commands as needed. You do not have to use only one command as in the example. For each project, first change to the project directory, then copy the source files to this directory. Keep in mind that you are making copies, therefore the original files will remain in their original locations after the files are copied to the project directories.

   - Copy files (all types) containing the numbers 1 and 2 in to the **`friends`** subdirectory.
   - Copy files (all types) containing the numbers 3 and 4 in to the **`family`** subdirectory.

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

```
[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-rw-r--. 1 student student 0 Feb  4 18:42 film1.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:42 film2.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:42 snap1.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:42 snap2.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:42 song1.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:42 song2.mp3
[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
-rw-rw-r--. 1 student student 0 Feb  4 18:44 film3.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:44 film4.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:44 snap3.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:44 snap4.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:44 song3.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:44 song4.mp3
```

7. For your work project, create additional copies.

```
[student@servera family]$ cd ../work
[student@servera work]$ cp ~/Music/song5.mp3 ~/Music/song6.mp3 \
~/Pictures/snap5.jpg ~/Pictures/snap6.jpg \
~/Videos/film5.avi ~/Videos/film6.avi .
[student@servera work]$ ls -l
total 0
-rw-rw-r--. 1 student student 0 Feb  4 18:48 film5.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:48 film6.avi
-rw-rw-r--. 1 student student 0 Feb  4 18:48 snap5.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:48 snap6.jpg
-rw-rw-r--. 1 student student 0 Feb  4 18:48 song5.mp3
-rw-rw-r--. 1 student student 0 Feb  4 18:48 song6.mp3
```

8. Your project tasks are now complete, and it is time to clean up the projects.

   Change to the `student` user's home directory. Attempt to delete both the **`family`** and **`friends`** project directories with a single **`rmdir`** command.

```
[student@servera work]$ cd
[student@servera ~]$ rmdir family friends
rmdir: failed to remove 'family': Directory not empty
rmdir: failed to remove 'friends': Directory not empty
```

   Using the **`rmdir`** command should fail because both subdirectories contain files.

9. Use the **`rm -r`** command to recursively delete both the **`family`** and **`friends`** subdirectories and their contents.

```
[student@servera ~]$ rm -r family friends
[student@servera ~]$ ls -l
total 0
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos
drwxrwxr-x. 2 student student 108 Feb  4 18:48 work
```

10. Delete all the files in the work project, but do not delete the work directory.

```
[student@servera ~]$ cd work
[student@servera work]$ rm song5.mp3 song6.mp3 snap5.jpg snap6.jpg \
film5.avi film6.avi
[student@servera work]$ ls -l
total 0
```

11. Finally, from the `student` user's home directory, use the **`rmdir`** command to delete the **`work`** directory. The command should succeed now that it is empty.

```
[student@servera work]$ cd
[student@servera ~]$ rmdir work
[student@servera ~]$ ls -l
total 0
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Music
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Pictures
drwxrwxr-x. 2 student student 108 Feb  4 18:36 Videos
```

12. Exit from `servera`.

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

Finish

On `workstation`, run the **`lab files-manage finish`** script to finish this exercise. The script will remove all directories and files created during this exercise.

```
[student@workstation ~]$ lab files-manage finish
```

This concludes the guided exercise.

MAKING LINKS BETWEEN FILES

OBJECTIVES

After completing this section, you should be able to make multiple file names reference the same file using hard links and symbolic (or "soft") links.

MANAGING LINKS BETWEEN FILES

Hard Links and Soft Links

It is possible to create multiple names that point to the same file. There are two ways to do this: by creating a *hard link* to the file, or by creating a *soft link* (sometimes called a *symbolic link*) to the file. Each has its advantages and disadvantages.

Creating 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 new 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. Once created, you cannot tell the difference between the new hard link and the original name of the file.

You can find out if a file has multiple hard links with the **`ls -l`** command. One of the things it reports is each file's *link count*, the number of hard links the file has.

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

In the preceding example, the link count of **`newfile.txt`** is 1. It has exactly one absolute path, which is **`/home/user/newfile.txt`**.

You can use the **`ln`** command to create a new hard link (another 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 named **`newfile-link2.txt`** for the existing file **`newfile.txt`** 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
```

If you want to find out whether two files are hard links of each other, one way is to use the **`-i`** option with the **`ls`** command to list the files' *inode number*. If the files are on the same file system (discussed in a moment) and their inode numbers are the same, the files are hard links pointing to the same data.

```
[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

All hard links that reference the same file will have the same link count, access permissions, user and group ownerships, time stamps, and file content. If any of that information is changed using one hard link, all other hard links pointing to the same file will show the new information as well. This is because each hard link points to the same data on the storage device.

Even if the original file gets deleted, the contents of the file are still available as long as at least one hard link exists. Data is only deleted from storage when the last hard link is deleted.

```
[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. Firstly, hard links can only be used with regular files. You cannot use **ln** to create a hard link to a directory or special file.

Secondly, hard links can only be used if both files are on the same *file system*. The file-system hierarchy can be made up of multiple storage devices. Depending on the configuration of your system, when you change into a new directory, that directory and its contents may 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 output like the following:

```
[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_rhel8--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. (The most specific match wins.) So, the system in this example, you can create a hard link between **/var/tmp/link1** and **/home/user/file** because they are both subdirectories of **/** but not any other directory on the list. But you cannot create a hard link between **/boot/test/badlink** and **/home/user/file** because the first file is in a subdirectory of **/boot** (on the "Mounted on" list) and the second file is not.

Creating Soft Links

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

Soft links have some advantages over hard links:

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

In the following example, the **ln -s** command is used to create a new soft link for the existing file **/home/user/newfile-link2.txt** that will be named **/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
Soft Hello World
```

In the preceding example, the first character of the long listing for **/tmp/newfile-symlink.txt** is **l** instead of **-**. This indicates that the file is a soft link and not a regular file. (A **d** would indicate that the file is a directory.)

When the original regular file gets deleted, the soft link will still point to the file but the target is gone. A soft link pointing to a missing file is called a "dangling soft 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 soft link in the preceding example is that if you later create a new file with the same name as the deleted file (**/home/user/newfile-link2.txt**), the soft link will no longer be "dangling" and will point to the new file.

Hard links do not work like this. If you delete a hard link and then use normal tools (rather than **ln**) to create a new file with the same name, the new file will not be linked to the old file.

One way to compare hard links and soft links that might help you understand how they work:

- A hard link points a name to data on a storage device
- A soft link points a name to another name, that points to data on a storage device

A soft link can point to a directory. The soft link then acts like a directory. Changing to the soft link with **cd** will make the current working directory the linked directory. Some tools may keep track of the fact that you followed a soft link to get there. For example, by default **cd** will update

your current working directory using the name of the soft link rather than the name of the actual directory. (There is an option, **-P**, that will update it with the name of the actual directory instead.)

In the following example, a soft link named **/home/user/configfiles** is created 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
```

REFERENCES

ln(1) man page

info ln (*'ln': Make links between files*)

GUIDED EXERCISE

MAKING LINKS BETWEEN FILES

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

OUTCOMES

You should be able to create hard links and soft links between files.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab files-make start`** command. This command runs a start script that determines if the `servera` host is reachable on the network and creates the files and working directories on `servera`.

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

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 therefore a password is not required.

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

2. Create a hard link named **`/home/student/backups/source.backup`** for the existing file, **`/home/student/files/source.file`**.

   2.1. View the link count for the file, **`/home/student/files/source.file`**.

```
[student@servera ~]$ ls -l files/source.file
total 4
-rw-r--r--. 1 student student 11 Mar  5 21:19 source.file
```

   2.2. Create a hard link named **`/home/student/backups/source.backup`**. Link it to the file, **`/home/student/files/source.file`**.

```
[student@servera ~]$ ln /home/student/files/source.file \
/home/student/backups/source.backup
```

   2.3. Verify the link count for the original **`/home/student/files/source.file`** and the new linked file, **`/home/student/backups/source.backup`**. The link count should be **2** for both files.

```
[student@servera ~]$ ls -l /home/student/files/
-rw-r--r--. 2 student student 11 Mar  5 21:19 source.file
[student@servera ~]$ ls -l /home/student/backups/
-rw-r--r--. 2 student student 11 Mar  5 21:19 source.backup
```

3. Create a soft link named **/home/student/tempdir** that points to the **/tmp** directory on servera.

   3.1. Create a soft link named **/home/student/tempdir** and link it to **/tmp**.

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

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

```
[student@servera ~]$ ls -l /home/student/tempdir
lrwxrwxrwx. 1 student student  4 Mar  5 22:04 /home/student/tempdir -
> /tmp
```

4. Exit from servera.

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

Finish

On workstation, run the **lab files-make finish** script to finish this exercise. This script removes all files and directories created on servera during the exercise.

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

This concludes the guided exercise.

SUMMARY

In this chapter, you learned:

- Files on a Linux system are organized into a single inverted tree of directories, known as a file-system hierarchy.
- Absolute paths start with a / and specify the location of a file in the file-system hierarchy.
- Relative paths do not start with a / and specify the location of a file relative to the current working directory.
- Five key commands are used to manage files: **`mkdir`**, **`rmdir`**, **`cp`**, **`mv`**, and **`rm`**.
- Hard links and soft links are different ways to have multiple file names point to the same data.

CHAPTER 3

MANAGING 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 using the **sudo** command.
- Create, modify, and delete locally defined user accounts.
- Create, modify, and delete locally defined group accounts.
- Set a password management policy for users, and manually lock and unlock user accounts.

SECTIONS

- Describing Users and Groups Concepts (and Quiz)
- Gaining Superuser Access (and Guided Exercise)
- Managing Local User Accounts (and Guided Exercise)
- Managing Local Group Accounts (and Guided Exercise)
- Managing User Passwords (and Guided Exercise)

LAB

Managing Local Linux Users and Groups

DESCRIBING USER AND GROUP CONCEPTS

OBJECTIVES

After completing this section, you should be able to describe the purpose of users and groups on a Linux system.

WHAT IS A USER?

A *user* account is used to provide security boundaries between different people and programs that can run commands.

Users have *user names* to identify them to human users and make them easier to work with. Internally, the system distinguishes user accounts by the unique identification number assigned to them, the *user ID* or *UID*. If a user account is used by humans, it will generally be assigned a secret *password* that the user will use to prove that they are the actual authorized user when logging 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. File ownership helps the system enforce access control for users of the files. The user associated with a running process determines the files and directories accessible to that process.

There are three main types of user account: the *superuser*, *system users*, and *regular users*.

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

You can use the **`id`** command to show information about the currently logged-in user.

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

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

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

To view the owner of a file use the **`ls -l`** command. To view the owner of a directory use the **`ls -ld`** command. In the following output, the third column shows the username.

```
[user01@host ~]$ ls -l file1
-rw-rw-r--. 1 user01 user01 0 Feb  5 11:10 file1
[user01@host]$ ls -ld dir1
drwxrwxr-x. 2 user01 user01 6 Feb  5 11:10 dir1
```

To view process information, use the **ps** command. The default is to show only processes in the current shell. Add the **a** option to view all processes with a terminal. To view the user associated with a process, include the **u** option. 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       777  0.0  0.0 225752  1496 tty1     Ss+  11:03   0:00 /sbin/agetty -o -
p -- \u --noclear tty1 linux
root       780  0.0  0.1 225392  2064 ttyS0    Ss+  11:03   0:00 /sbin/agetty -o -
p -- \u --keep-baud 115200,38400,9600
user01      1207  0.0  0.2 234044  5104 pts/0    Ss   11:09   0:00 -bash
user01      1319  0.0  0.2 266904  3876 pts/0    R+   11:33   0:00 ps au
```

The output of the preceding command displays users by name, but internally the operating system uses the 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. It is divided up into seven colon-separated fields. Here is an example of a line from **/etc/passwd**:

```
❶user01:❷x:❸1000:❹1000:❺User One:❻/home/user01:❼/bin/bash
```

❶ Username for this user (`user01`).
❷ The user's password used to be stored here in encrypted format. That has been moved to the **/etc/shadow** file, which will be covered later. This field should always be `x`.
❸ The UID number for this user account (`1000`).
❹ The GID number for this user account's primary group (`1000`). Groups will be discussed later in this section.
❺ The real name for this user (`User One`).
❻ The home directory for this user (**/home/user01**). This is the initial working directory when the shell starts and contains the user's data and configuration settings.
❼ The default shell program for this user, which runs on login (**/bin/bash**). For a regular user, this is normally the program that provides the user's command-line prompt. A system user might use **/sbin/nologin** if interactive logins are not allowed for that user.

WHAT IS A GROUP?

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

Like users, groups have *group names* to make them easier to work with. Internally, the system distinguishes groups by the unique identification number assigned to them, the *group ID* or *GID*.

The mapping of group names to GIDs is defined in 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. Here is an example of a line from **/etc/group**:

```
❶group01:❷x:❸10000:❹user01,user02,user03
```

❶ Group name for this group (`group01`).
❷ Obsolete group password field. This field should always be `x`.
❸ The GID number for this group (`10000`).
❹ A list of users who are members of this group as a supplementary group (`user01`, `user02`, `user03`). Primary (or default) and supplementary groups are discussed later in this section.

Primary Groups and Supplementary Groups

Every user has exactly one primary group. For local users, this is the group listed by GID number in the **/etc/passwd** file. By default, this is the group that will own new files created by the user.

Normally, when you create a new regular user, a new group with the same name as that user is created. That group is used as the primary group for the new user, and that user is the only member of this *User Private Group*. It turns out that this helps make management of file permissions simpler, which will be discussed later in this course.

Users may also have *supplementary groups*. Membership in supplementary groups is determined by the **/etc/group** file. Users are granted access to files based on whether any of their groups have access. It doesn't matter if the group or groups that have access are primary or supplementary for the user.

For example, if the user `user01` has a primary group `user01` and supplementary groups `wheel` and `webadmin`, then that user can read files readable by any of those three groups.

The **id** command can also be used to find out about group membership for a user.

```
[user03@host ~]$ id
uid=1003(user03) gid=1003(user03) groups=1003(user03),10(wheel),10000(group01)
 context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

In the preceding example, `user03` has the group `user03` as their primary group (**gid**). The **groups** item lists all groups for this user, and other than the primary group `user03`, the user has groups `wheel` and `group01` as supplementary groups.

REFERENCES

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

info libc (*GNU C Library Reference Manual*)

- Section 30: Users and groups

(Note that the *glibc-devel* package must be installed for this info node to be available.)

QUIZ

DESCRIBING 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

DESCRIBING 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

GAINING SUPERUSER ACCESS

OBJECTIVES

After completing this section, you will be able to 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 some sort of *superuser*, a user that has all power over the system. In Red Hat Enterprise Linux this is the `root` user. This user has the power to override normal privileges on the file system, and is used to manage and administer the system. To perform tasks such as installing or removing software and to manage system files and directories, users must escalate their privileges to the `root` user.

The `root` user only among normal users can control most devices, but there are a few exceptions. For example, normal users can control removable devices, such as USB devices. Thus, normal users can add and remove files and otherwise manage a removable device, but only `root` can manage "fixed" 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's account is compromised, someone else would have administrative control of the system. Throughout this course, administrators are encouraged to log in as a normal user and escalate privileges to `root` only when needed.

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

WARNING

One common practice on Microsoft Windows in the past was for the local `Administrator` user to log in directly to perform system administrator duties. Although this is possible on Linux, Red Hat recommends that system administrators do not log in directly as `root`. Instead, system administrators should log in as a normal user and use other mechanisms (**su**, **sudo**, or ***PolicyKit***, for example) to temporarily gain superuser privileges.

By logging in as the superuser, the entire desktop environment unnecessarily runs with administrative privileges. In that situation, any security vulnerability which would normally only compromise the user account has the potential to compromise the entire system.

SWITCHING USERS

The **su** command allows users to switch to a different user account. If you run **su** from a regular user account, you will be prompted for the password of the account to which you want to switch. When `root` runs **su**, you do not need to enter the user's password.

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

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

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

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

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

NOTE

The **su** command is most frequently used to get a command-line interface (shell prompt) which is running as another user, typically `root`. However, with the **-c** option, it can be used like the Windows utility **runas** to run an arbitrary program as another user. Run **info su** to view more details.

RUNNING COMMANDS WITH SUDO

In some cases, the `root` user's account may not have a valid password at all for security reasons. In this case, users cannot log in to the system as `root` directly with a password, and **su** cannot be used to get an interactive shell. One tool that can be used to get `root` access in this case is **sudo**.

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

Additionally, **sudo** can be configured to allow specific users to run any command as some other user, or only some commands as that user.

For example, when **sudo** is configured to allow the `user01` user to run the command **usermod** as `root`, `user01` could run the following command to lock or unlock a user account:

```
[user01@host ~]$ sudo usermod -L user02
[sudo] password for user01:
[user01@host ~]$ su - 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, the command will be blocked, the attempt will be logged, and by default an email will be sent to the `root` user.

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

One additional benefit to using **sudo** is that all commands executed are logged by default to **/var/log/secure**.

```
[user01@host ~]$ sudo tail /var/log/secure
...output omitted...
Feb  6 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 Red Hat Enterprise Linux 8, all members of the `wheel` group can use **sudo** to run commands as any user, including `root`. The user is prompted for their own password. This is a change from Red Hat Enterprise Linux 6 and earlier, where users who were members of the `wheel` group did not get this administrative access by default.

WARNING

RHEL 6 did not grant the `wheel` group any special privileges by default. Sites that have been using this group for a non-standard purpose might be surprised when RHEL 7 and RHEL 8 automatically grants all members of `wheel` full **sudo** privileges. This could lead to unauthorized users getting administrative access to RHEL 7 and RHEL 8 systems.

Historically, UNIX-like systems use membership in the `wheel` group to grant or control superuser access.

Getting an Interactive Root Shell with Sudo

If there is a nonadministrative user account on the system that can use **sudo** to run the **su** command, you can run **sudo su -** from that account to get an interactive `root` user shell. This works because **sudo** will run **su -** as `root`, and `root` does not need to enter a password to use **su**.

Another way to access the `root` account with **sudo** is to use the **sudo -i** command. This will switch to the `root` account and run that user's default shell (usually **bash**) and associated shell login scripts. If you just want to run the shell, you can use the **sudo -s** command.

For example, an administrator might get an interactive shell as `root` on an AWS EC2 instance by using SSH public-key authentication to log in as the normal user `ec2-user`, and then by running **sudo -i** to get the `root` user's shell.

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

The **sudo su -** command and **sudo -i** do not behave exactly the same. This will be discussed briefly at the end of the section.

Configuring Sudo

The main configuration file for **sudo** is **/etc/sudoers**. To avoid problems if multiple administrators try to edit it at the same time, it should only be edited with the special **visudo** command.

For example, the following line from the **/etc/sudoers** file enables **sudo** access for members of group wheel.

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

In this line, **%wheel** is the user or group to whom the rule applies. A % specifies that this is a group, group wheel. The **ALL=(ALL)** specifies that on any host that might have this file, wheel can run any command. The final **ALL** specifies that wheel can run those commands as any user on the system.

By default, **/etc/sudoers** also includes the contents of any files in the **/etc/sudoers.d** directory as part of the configuration file. This allows an administrator to add **sudo** access for a user simply by putting an appropriate file in that directory.

> **NOTE**
> Using supplementary files under the **/etc/sudoers.d** directory is convenient and simple. You can enable or disable **sudo** access simply by copying a file into the directory or removing it from the directory.
>
> In this course, you will 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 user user01, you could create **/etc/sudoers.d/user01** with the following content:

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

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

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

It is also possible to set up **sudo** to allow a user to run commands as another user without entering their password:

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

While there are obvious security risks to granting this level of access to a user or group, it is frequently used with cloud instances, virtual machines, and provisioning systems to help configure servers. The account with this access must be carefully protected and might require SSH public-key authentication in order for a user on a remote system to access it at all.

For example, the official AMI for Red Hat Enterprise Linux in the Amazon Web Services Marketplace ships with the root and the ec2-user users' passwords locked. The ec2-user user account is set up to allow remote interactive access through SSH public-key authentication. The

user `ec2-user` can also run any command as `root` without a password because the last line of the AMI's **/etc/sudoers** file is set up as follows:

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

The requirement to enter a password for **sudo** can be re-enabled or other changes may be made to tighten security as part of the process of configuring the system.

NOTE

In this course, you will frequently see **sudo su -** used instead of **sudo -i**. Both commands work, but there are some subtle differences between them.

The **sudo su -** command sets up the `root` environment exactly like a normal login because the **su -** command ignores the settings made by **sudo** and sets up the environment from scratch.

The default configuration of the **sudo -i** command actually sets up some details of the `root` user's environment differently than a normal login. For example, it sets the PATH environment variable slightly differently. This affects where the shell will look to find commands.

You can make **sudo -i** behave more like **su -** by editing **/etc/sudoers** with **visudo**. Find the line

```
Defaults    secure_path = /sbin:/bin:/usr/sbin:/usr/bin
```

and replace it with the following two lines:

```
Defaults      secure_path = /usr/local/bin:/usr/bin
Defaults>root secure_path = /usr/local/sbin:/usr/local/bin:/usr/sbin:/
usr/bin
```

For most purposes, this is not a major difference. However, for consistency of PATH settings on systems with the default **/etc/sudoers** file, the authors of this course mostly use **sudo su -** in examples.

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

(Note that the *glibc-devel* package must be installed for this info node to be available.)

GUIDED EXERCISE

GAINING SUPERUSER ACCESS

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

OUTCOMES

You should be able to:

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

BEFORE YOU BEGIN

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

On `workstation`, run **`lab users-sudo start`** to start the exercise. This script creates the necessary user accounts and files to set up the environment correctly.

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

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

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

2. Explore the shell environment of `student`. 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 executables.

   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 `root` in a non-login shell and explore the new shell environment.

3.1. Run **sudo su** 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
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

If you already have some experience with Linux and the **su** command, you may have expected that using **su** without the dash (`-`) option to become `root` would cause you to keep the current PATH of `student`. That did not happen. As you will see in the next step, this is not the usual PATH for `root` either.

What happened? The difference is that you did not run **su** directly. Instead, you ran **su** as `root` using **sudo** because you did not possess the password of the superuser. The **sudo** command initially overrides the PATH variable from the initial environment

for security reasons. Any command that runs after the initial override can still update the PATH variable, as you will 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 `root` in a login shell and explore the new shell environment.

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

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

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

sudo may or may 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 have authenticated to **sudo** within the last five minutes, **sudo** will not prompt you for the password. If it has been more than five minutes since you authenticated to **sudo**, you need to enter `student` as the password to get authenticated to **sudo**.

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
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
```

As in the preceding step, after **sudo** reset the PATH variable from the settings in the `student` user's shell environment, the **su -** command ran the shell login scripts for

root and set the PATH variable to yet another value. The **su** command without the dash (-) option did not do that.

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 is configured as to run any command as any user using **sudo**.

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

6. Become operator1 and view the contents of **/var/log/messages**. Copy **/etc/motd** to **/etc/motdOLD** and remove it (/etc/motdOLD). These operations require administrative

rights and so use **sudo** to run those commands as the superuser. Do not switch to root using **sudo su** or **sudo su -**. Use `redhat` as the password of `operator1`.

6.1. Switch to `operator1`.

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

6.2. Attempt to view the last five lines of **/var/log/messages** without using **sudo**. This should fail.

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

6.3. Attempt to view the last five lines of **/var/log/messages** with **sudo**. This should succeed.

```
[operator1@servera ~]$ sudo tail -5 /var/log/messages
[sudo] password for operator1: redhat
Jan 23 15:53:36 servera su[2304]: FAILED SU (to operator1) student on
 pts/1
Jan 23 15:53:51 servera su[2307]: FAILED SU (to operator1) student on
 pts/1
Jan 23 15:53:58 servera su[2310]: FAILED SU (to operator1) student on
 pts/1
Jan 23 15:54:12 servera su[2322]: (to operator1) student on pts/1
Jan 23 15:54:25 servera su[2353]: (to operator1) student on pts/1
```

NOTE
The preceding output may differ on your system.

6.4. Attempt to make a copy of **/etc/motd** as **/etc/motdOLD** without using **sudo**. This should fail.

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

6.5. Attempt to make a copy of **/etc/motd** as **/etc/motdOLD** with **sudo**. This should succeed.

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

6.6. Attempt to delete **/etc/motdOLD** without using **sudo**. This should fail.

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

6.7. Attempt to delete **/etc/motdOLD** with **sudo**. This should succeed.

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

6.8. Exit the `operator1` user's shell to return to the `student` user's shell.

```
[operator1@servera ~]$ exit
logout
[student@servera ~]$
```

6.9. Log off from `servera`.

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

Finish

On `workstation`, run **lab users-sudo finish** to complete this exercise. This script deletes the user accounts and files created at the start of the exercise to ensure that the environment is clean.

```
[student@workstation ~]$ lab users-sudo finish
```

This concludes the guided exercise.

MANAGING LOCAL USER ACCOUNTS

OBJECTIVES

After completing this section, you should be able to create, modify, and delete local user accounts.

MANAGING LOCAL USERS

A number of command-line tools can be used to manage local user accounts.

Creating Users from the Command Line

- The **`useradd`** ***`username`*** command creates a new user named `username`. It sets up the user's home directory and account information, and creates a private group for the user named `username`. At this point the account does not have a valid password set, and the user cannot log in until a password is set.
- The **`useradd --help`** command displays the basic options that can be used to override the defaults. In most cases, the same options can be used with the **`usermod`** command to modify an existing user.
- Some defaults, such as the range of valid UID numbers and default password aging rules, are read from the **`/etc/login.defs`** file. Values in this file are only used when creating new users. A change to this file does not affect existing users.

Modifying Existing Users from the Command Line

- The **`usermod --help`** command displays the basic options that can be used to modify an account. Some common options include:

USERMOD OPTIONS:	USAGE
`-c, --comment COMMENT`	Add the user's real name to the comment field.
`-g, --gid GROUP`	Specify the primary group for the user account.
`-G, --groups GROUPS`	Specify a comma-separated list of supplementary groups for the user account.
`-a, --append`	Used with the **`-G`** 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.
`-d, --home HOME_DIR`	Specify a particular home directory for the user account.
`-m, --move-home`	Move the user's home directory to a new location. Must be used with the **`-d`** option.
`-s, --shell SHELL`	Specify a particular login shell for the user account.
`-L, --lock`	Lock the user account.

USERMOD OPTIONS:	USAGE
-U, --unlock	Unlock the user account.

Deleting Users from the Command Line

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

WARNING

When a user is removed with **userdel** without the **-r** option specified, the system will have files that are owned by an unassigned UID. This can also happen when a file, having a deleted user as its owner, exists outside that user's home directory. This situation can lead to information leakage and other security issues.

In Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux 8, the **useradd** command assigns new users the first free UID greater than or equal to 1000, unless you explicitly specify one using the **-u** option.

This is how information leakage can occur. If the first free UID had been previously assigned to a user account which has since been removed from the system, the old user's UID will get reassigned to the new user, giving the new user ownership of the old user's remaining files.

The following scenario demonstrates this situation.

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

Notice that `user02` now owns all files that `user01` previously owned.

Depending on the situation, one solution to this problem is to remove all unowned files from the system when the user that created them is deleted. Another solution is to manually assign the unowned files to a different user. The `root` user can use the **find / -nouser -o -nogroup** command to find all unowned files and directories.

Setting Passwords from the Command Line

- The **passwd *username*** command sets the initial password or changes the existing password of `username`.

- The `root` user can set a password to any value. A message is displayed if the password does not meet the minimum recommended criteria, but is followed by a prompt to retype the new password and all tokens are updated successfully.

```
[root@host ~]# passwd user01
Changing password for user user01.
New password: redhat
BAD PASSWORD: The password fails the dictionary check - it is based on a
 dictionary word
Retype new password: redhat
passwd: all authentication tokens updated successfully.
[root@host ~]#
```

- A regular user must choose a password at least eight characters long and is also not based on a dictionary word, the username, or the previous password.

UID Ranges

Specific UID numbers and ranges of numbers are used for specific purposes by Red Hat Enterprise Linux.

- *UID 0* is always assigned to the superuser account, **`root`**.
- *UID 1-200* is a range of "system users" assigned statically to system processes by Red Hat.
- *UID 201-999* is a range of "system users" used by system processes that do not own files on the file system. They are typically assigned dynamically from the available pool when the software that needs them is installed. Programs run as these "unprivileged" system users in order to limit their access to only the resources they need to function.
- *UID 1000+* is the range available for assignment to regular users.

NOTE
Prior to RHEL 7, the convention was that UID 1-499 was used for system users and UID 500+ for regular users. Default ranges used by **`useradd`** and **`groupadd`** can be changed in the **`/etc/login.defs`** file.

REFERENCES
`useradd`(8), **`usermod`**(8), **`userdel`**(8) man pages

GUIDED EXERCISE

MANAGING LOCAL USER ACCOUNTS

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

OUTCOMES

You should be able to configure a Linux system with additional user accounts.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab users-manage start`** to start the exercise. This script ensures that the environment is set up correctly.

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

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

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

2. On `servera`, switch to `root` using **sudo**, converting to the `root` user's shell environment.

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

3. 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:1001:1001::/home/operator1:/bin/bash
```

4. 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.
```

5. Create the additional users called `operator2` and `operator3`. Set their passwords to **`redhat`**.

   5.1. 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.
```

   5.2. 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.
```

6. Update the `operator1` and `operator2` user accounts to include the **`Operator One`** and **`Operator Two`** comments, respectively. Verify that the comments are successfully added.

   6.1. Run **`usermod -c`** to update the comments of the `operator1` user account.

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

   6.2. Run **`usermod -c`** to update the comments of the `operator2` user account.

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

   6.3. Confirm that the comments for each of the `operator1` and `operator2` users are reflected in the user records.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1001:1001:Operator One:/home/operator1:/bin/bash
operator2:x:1002:1002:Operator Two:/home/operator2:/bin/bash
operator3:x:1003:1003::/home/operator3:/bin/bash
```

7. Delete the `operator3` user along with any personal data of the user. Confirm that the user is successfully deleted.

   7.1. Remove the `operator3` user from the system.

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

   7.2. Confirm that `operator3` is successfully deleted.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1001:1001:Operator One:/home/operator1:/bin/bash
operator2:x:1002:1002:Operator Two:/home/operator2:/bin/bash
```

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

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

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

   7.4. Log off from `servera`.

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

Finish

On `workstation`, run **`lab users-manage finish`** to complete this exercise. This script ensures that the environment is clean.

```
[student@workstation ~]$ lab users-manage finish
```

This concludes the guided exercise.

MANAGING LOCAL GROUP ACCOUNTS

OBJECTIVES

After completing this section, students should be able to create, modify, and delete local group accounts.

MANAGING LOCAL GROUPS

A group must exist before a user can be added to that group. Several command-line tools are used to manage local group accounts.

Creating Groups from the Command Line

- The **groupadd** command creates groups. Without options the **groupadd** command uses the next available GID from the range specified in the **/etc/login.defs** file while creating the groups.
- The **-g** option specifies a particular GID for the group to use.

```
[user01@host ~]$ sudo groupadd -g 10000 group01
[user01@host ~]$ tail /etc/group
...output omitted...
group01:x:10000:
```

> **NOTE**
> Given the automatic creation of user private groups (GID 1000+), it is generally recommended to set aside a range of GIDs to be used for supplementary groups. A higher range will avoid a collision with a system group (GID 0-999).

- The **-r** option creates a system group using a GID from the range of valid system GIDs listed in the **/etc/login.defs** file. The SYS_GID_MIN and SYS_GID_MAX configuration items in **/etc/login.defs** define the range of system GIDs.

```
[user01@host ~]$ sudo groupadd -r group02
[user01@host ~]$ tail /etc/group
...output omitted...
group01:x:10000:
group02:x:988:
```

Modifying Existing Groups from the Command Line

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

```
[user01@host ~]$ sudo groupmod -n group0022 group02
[user01@host ~]$ tail /etc/group
...output omitted...
group0022:x:988:
```

Notice that the group name is updated to `group0022` from `group02`.

- The **-g** option specifies a new GID.

```
[user01@host ~]$ sudo groupmod -g 20000 group0022
[user01@host ~]$ tail /etc/group
...output omitted...
group0022:x:20000:
```

Notice that the GID is updated to **20000** from **988**.

Deleting Groups from the Command Line

- The **groupdel** command removes groups.

```
[user01@host ~]$ sudo groupdel group0022
```


NOTE
You cannot remove a group if it is the primary group of any existing user. As with **userdel**, check all file systems to ensure that no files remain on the system that are owned by the group.

Changing 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.

```
[user01@host ~]$ id user02
uid=1006(user02) gid=1008(user02) groups=1008(user02)
[user01@host ~]$ sudo usermod -g group01 user02
[user01@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.

```
[user01@host ~]$ id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03)
[user01@host ~]$ sudo usermod -aG group01 user03
[user01@host ~]$ id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
```

IMPORTANT

The use of the **-a** option makes **usermod** function in *append* mode. Without **-a**, the user will be removed from any of their current supplementary groups that are not included in the **-G** option's list.

REFERENCES

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

GUIDED EXERCISE

MANAGING LOCAL GROUP ACCOUNTS

In this exercise, you will 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

You should be able to:

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

BEFORE YOU BEGIN

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

On `workstation`, run **`lab users-group-manage start`** to start the exercise. This script creates the necessary user accounts to set up the environment correctly.

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

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

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

2. On `servera`, switch to `root` using **sudo**, inheriting the full environment of the `root` user.

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

3. Create the `operators` supplementary group with the GID of 30000.

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

4. Create `admin` as an additional supplementary group.

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

5. Verify that both the `operators` and `admin` supplementary groups exist.

```
[root@servera ~]# tail /etc/group
...output omitted...
operators:x:30000:
admin:x:30001:
```

6. Ensure that the users `operator1`, `operator2` and `operator3` belong to the group `operators`.

   6.1. Add `operator1`, `operator2`, and `operator3` to `operators`.

```
[root@servera ~]# usermod -aG operators operator1
[root@servera ~]# usermod -aG operators operator2
[root@servera ~]# usermod -aG operators operator3
```

   6.2. Confirm that the users are successfully added to the group.

```
[root@servera ~]# id operator1
uid=1001(operator1) gid=1001(operator1)
 groups=1001(operator1),30000(operators)
[root@servera ~]# id operator2
uid=1002(operator2) gid=1002(operator2)
 groups=1002(operator2),30000(operators)
[root@servera ~]# id operator3
uid=1003(operator3) gid=1003(operator3)
 groups=1003(operator3),30000(operators)
```

7. Ensure that the users sysadmin1, sysadmin2 and sysadmin3 belong to the group admin. Enable administrative rights for all the group members of admin. Verify that any member of admin can run administrative commands.

7.1. Add sysadmin1, sysadmin2, and sysadmin3 to admin.

```
[root@servera ~]# usermod -aG admin sysadmin1
[root@servera ~]# usermod -aG admin sysadmin2
[root@servera ~]# usermod -aG admin sysadmin3
```

7.2. Confirm that the users are successfully added to the group.

```
[root@servera ~]# id sysadmin1
uid=1004(sysadmin1) gid=1004(sysadmin1)
 groups=1004(sysadmin1),30001(admin)
[root@servera ~]# id sysadmin2
uid=1005(sysadmin2) gid=1005(sysadmin2)
 groups=1005(sysadmin2),30001(admin)
[root@servera ~]# id sysadmin3
uid=1006(sysadmin3) gid=1006(sysadmin3)
 groups=1006(sysadmin3),30001(admin)
```

7.3. Examine **/etc/group** to verify the supplemental group memberships.

```
[root@servera ~]# tail /etc/group
...output omitted...
operators:x:30000:operator1,operator2,operator3
admin:x:30001:sysadmin1,sysadmin2,sysadmin3
```

7.4. Create the **/etc/sudoers.d/admin** file such that the members of admin have full administrative privileges.

```
[root@servera ~]# echo "%admin ALL=(ALL) ALL" >> /etc/sudoers.d/admin
```

7.5. Switch to sysadmin1 (a member of admin) and verify that you can run a **sudo** command as sysadmin1.

```
[root@servera ~]# su - sysadmin1
[sysadmin1@servera ~]$ sudo cat /etc/sudoers.d/admin
[sudo] password for sysadmin1: redhat
%admin ALL=(ALL) ALL
```

7.6. Exit the sysadmin1 user's shell to return to the root user's shell.

```
[sysadmin1@servera ~]$ exit
logout
[root@servera ~]#
```

7.7. Exit the root user's shell to return to the student user's shell.

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

7.8. Log off from servera.

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

Finish

On workstation, run **lab users-group-manage finish** to complete this exercise. This script deletes the user accounts created at the start of the exercise.

```
[student@workstation ~]$ lab users-group-manage finish
```

This concludes the guided exercise.

MANAGING USER PASSWORDS

OBJECTIVES

After completing this section, you should be able to set a password management policy for users, and manually lock and unlock user accounts.

SHADOW PASSWORDS AND PASSWORD POLICY

At one time, encrypted passwords were stored in the world-readable **/etc/passwd** file. This was thought to be reasonably secure until dictionary attacks on encrypted passwords became common. At that point, the encrypted passwords were moved to a separate **/etc/shadow** file which is readable only by `root`. This new file also allowed password aging and expiration features to be implemented.

Like **/etc/passwd**, each user has a line in the **/etc/shadow** file. A sample line from **/etc/shadow** with its nine colon-separated fields is shown below.

```
❶user03:❷$6$CSsX...output omitted...:❸17933:❹0:❺99999:❻7:❼2:❽18113:❾
```

❶ Username of the account this password belongs to.
❷ The *encrypted password* of the user. The format of encrypted passwords is discussed later in this section.
❸ The day on which the password was last changed. This is set in days since 1970-01-01, and is calculated in the UTC time zone.
❹ The minimum number of days that have to elapse since the last password change before the user can change it again.
❺ The maximum number of days that can pass without a password change before the password expires. An empty field means it does not expire based on time since the last change.
❻ Warning period. The user will be warned about an expiring password when they login for this number of days before the deadline.
❼ Inactivity period. Once the password has expired, it will still be accepted for login for this many days. After this period has elapsed, the account will be locked.
❽ The day on which the password expires. This is set in days since 1970-01-01, and is calculated in the UTC time zone. An empty field means it does not expire on a particular date.
❾ The last field is usually empty and is reserved for future use.

Format of an Encrypted Password

The encrypted password field stores three pieces of information: the *hashing algorithm* used, the *salt*, and the *encrypted hash*. Each piece of information is delimited by the **$** sign.

```
$❶6$❷CSsXcYG1L/4ZfHr/$❸2W6evvJahUfzfHpc9X.45Jc6H30E...output omitted...
```

❶ The hashing algorithm used for this password. The number **6** indicates it is a SHA-512 hash, which is the default in Red Hat Enterprise Linux 8. A **1** would indicate MD5, a **5** SHA-256.
❷ The salt used to encrypt the password. This is originally chosen at random.
❸ The encrypted hash of the user's password. The salt and the unencrypted password are combined and encrypted to generate the encrypted hash of the password.

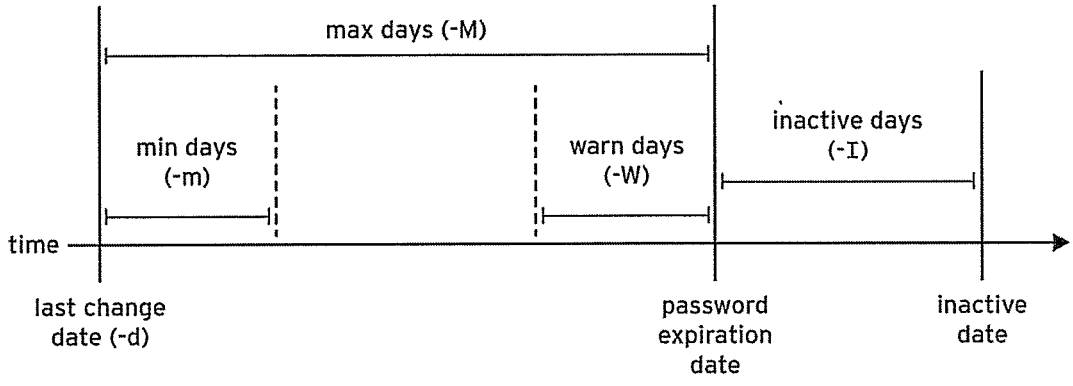
The use of a salt prevents two users with the same password from having identical entries in the **/etc/shadow** file. For example, even if `user01` and `user02` both use `redhat` as their passwords, their encrypted passwords in **/etc/shadow** will be different if their salts are different.

Password Verification

When a user tries to log in, the system looks up the entry for the user in **/etc/shadow**, combines the salt for the user with the unencrypted password that was typed in, and encrypts them using the hashing algorithm specified. If the result matches the encrypted hash, the user typed in the right password. If the result does not match the encrypted hash, the user typed in the wrong password and the login attempt fails. This method allows the system to determine if the user typed in the correct password without storing that password in a form usable for logging in.

CONFIGURING PASSWORD AGING

The following diagram relates the relevant password aging parameters, which can be adjusted using the **chage** command to implement a password aging policy.




```
[user01@host ~]$ sudo chage -m 0 -M 90 -W 7 -I 14 user03
```

The preceding **chage** command uses the **-m**, **-M**, **-W**, and **-I** options to set the minimum age, maximum age, warning period, and inactivity period of the user's password, respectively.

The **chage -d 0 user03** command forces the `user03` user to update its password on the next login.

The **chage -l user03** command displays the password aging details of `user03`.

The **chage -E 2019-08-05 user03** command causes the `user03` user's account to expire on 2019-08-05 (in YYYY-MM-DD format).

NOTE

The **date** command can be used to calculate a date in the future. The **-u** option reports the time in UTC.

```
[user01@host ~]$ date -d "+45 days" -u
Thu May 23 17:01:20 UTC 2019
```

Edit the password aging configuration items in the **/etc/login.defs** file to set the default password aging policies. The PASS_MAX_DAYS sets the default maximum age of the password. The PASS_MIN_DAYS sets the default minimum age of the password. The PASS_WARN_AGE sets the default warning period of the password. Any change in the default password aging policies will be effective for new users only. The existing users will continue to use the old password aging settings rather than the new ones.

RESTRICTING ACCESS

You can use the **chage** command to set account expiration dates. When that date is reached, the user cannot log in to the system interactively. The **usermod** command can lock an account with the **-L** option.

```
[user01@host ~]$ sudo usermod -L user03
[user01@host ~]$ su - user03
Password: redhat
su: Authentication failure
```

If a user leaves the company, the administrator may lock and expire an account with a single **usermod** command. The date must be given as the number of days since 1970-01-01, or in the *YYYY-MM-DD* format.

```
[user01@host ~]$ sudo usermod -L -e 2019-10-05 user03
```

The preceding **usermod** command uses the **-e** option to set the account expiry date for the given user account. The **-L** option locks the user's password.

Locking the account prevents the user from authenticating with a password to the system. It is the recommended method of preventing access to an account by an employee who has left the company. If the employee returns, the account can later be unlocked with **usermod -U**. If the account was also expired, be sure to also change the expiration date.

The nologin Shell

The **nologin** shell acts as a replacement shell for the user accounts not intended to interactively log into the system. It is wise from the security standpoint to disable the user account from logging into the system when the user acount serves a responsibility that does not require the user to log into the system. For example, a mail server may require an account to store mail and a password for the user to authenticate with a mail client used to retrieve mail. That user does not need to log directly into 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, the **nologin** shell closes the connection.

```
[user01@host ~]$ usermod -s /sbin/nologin user03
[user01@host ~]$ su - user03
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) man pages

GUIDED EXERCISE

MANAGING USER PASSWORDS

In this exercise, you will set password policies for several users.

OUTCOMES

You should be able to:

- 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

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

On `workstation`, run **`lab users-pw-manage start`** to start the exercise. This script creates the necessary user accounts and files to ensure that the environment is set up correctly.

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

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

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

2. On `servera`, explore locking and unlocking user accounts as `student`.

   2.1. As `student`, lock the `operator1` account using administrative rights.

```
[student@servera ~]$ sudo usermod -L operator1
[sudo] password for student: student
```

   2.2. Attempt to log in as `operator1`. This should fail.

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

   2.3. Unlock the `operator1` account.

```
[student@servera ~]$ sudo usermod -U operator1
```

2.4. Attempt to log in as `operator1` again. This should succeed.

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

2.5. Exit out of the `operator1` user's shell to return to the `student` user's shell.

```
[operator1@servera ~]$ exit
logout
```

3. Change the password policy for `operator1` to require a new password every 90 days. Confirm that the password age is successfully set.

3.1. Set the maximum age of the `operator1` user's password to 90 days.

```
[student@servera ~]$ sudo chage -M 90 operator1
```

3.2. Verify that the `operator1` user's password expires 90 days after it is changed.

```
[student@servera ~]$ sudo chage -l operator1
Last password change      : Jan 25, 2019
Password expires          : Apr 25, 2019
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.

```
[student@servera ~]$ sudo chage -d 0 operator1
```

5. Log in as `operator1` and change the password to `forsooth123`. After setting the password, return to the `student` user's shell.

5.1. 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 ~]$
```

5.2. Exit the `operator1` user's shell to return to the `student` user's shell.

```
[operator1@servera ~]$ exit
logout
```

6. Set the `operator1` account to expire 180 days from the current day. Hint: The **`date -d "+180 days"`** gives you the date and time 180 days from the current date and time.

6.1. Determine a date 180 days in the future. Use the format **%F** with the **`date`** command to get the exact value.

```
[student@servera ~]$ date -d "+180 days" +%F
2019-07-24
```

You may get a different value to use in the following step based on the current date and time in your system.

6.2. Set the account to expire on the date displayed in the preceding step.

```
[student@servera ~]$ sudo chage -E 2019-07-24 operator1
```

6.3. Verify that the account expiry date is successfully set.

```
[student@servera ~]$ sudo chage -l operator1
Last password change      : Jan 25, 2019
Password expires          : Apr 25, 2019
Password inactive         : never
Account expires           : Jul 24, 2019
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
```

7. Set the passwords to expire 180 days from the current date for all users. Use administrative rights to edit the configuration file.

7.1. Set PASS_MAX_DAYS to **180** in **/etc/login.defs**. Use administrative rights when opening the file with the text editor. You can use the **sudo 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_MIN_LEN    5
PASS_WARN_AGE   7
```

IMPORTANT
The default password and account expiry settings will be effective for new users but not for existing users.

7.2. Log off from servera.

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

Finish

On workstation, run **lab users-pw-manage finish** to complete this exercise. This script deletes the user accounts and files created at the start of the exercise to ensure that the environment is clean.

```
[student@workstation ~]$ lab users-pw-manage finish
```

This concludes the guided exercise.

▷ LAB

MANAGING LOCAL USERS AND GROUPS

PERFORMANCE CHECKLIST

In this lab you will 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

You should be able to:

- Set a default password aging policy of the local user's password.
- Create a group and use the group as a supplementary group for new users.
- Create three new users with the new group as their supplementary group.
- Configure the group members of the supplementary group to run any command as any user using **sudo**.
- Set a user-specific password aging policy.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab users-review start`** to start the exercise. This script creates the necessary files to ensure that the environment is set up correctly.

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

1. From `workstation`, open an SSH session to `serverb` as `student`.
2. On `serverb`, ensure that newly created users have passwords that must be changed every 30 days.
3. Create the new group called `consultants` with a GID of **`35000`**.
4. Configure administrative rights for all members of `consultants` to be able to execute any command as any user.
5. Create the `consultant1`, `consultant2`, and `consultant3` users with `consultants` as their supplementary group.
6. Set the `consultant1`, `consultant2`, and `consultant3` accounts to expire in 90 days from the current day.
7. Change the password policy for the `consultant2` account to require a new password every 15 days.

8. Additionally, force the `consultant1`, `consultant2`, and `consultant3` users to change their passwords on the first login.

Evaluation

On `workstation`, run the **`lab users-review grade`** command to confirm success of this exercise.

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

Finish

On `workstation`, run **`lab users-review finish`** to complete this lab. This script deletes the user accounts and files created throughout the lab to ensure that the environment is clean.

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

This concludes the lab.

▷ SOLUTION

MANAGING LOCAL USERS AND GROUPS

PERFORMANCE CHECKLIST

In this lab you will 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

You should be able to:

- Set a default password aging policy of the local user's password.
- Create a group and use the group as a supplementary group for new users.
- Create three new users with the new group as their supplementary group.
- Configure the group members of the supplementary group to run any command as any user using **sudo**.
- Set a user-specific password aging policy.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab users-review start`** to start the exercise. This script creates the necessary files to ensure that the environment is set up correctly.

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

1. From `workstation`, open an SSH session to `serverb` as `student`.

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

2. On `serverb`, ensure that newly created users have passwords that must be changed every 30 days.

   2.1. Set PASS_MAX_DAYS to **`30`** in **`/etc/login.defs`**. Use administrative rights while opening the file with the text editor. You can use the **`sudo vim /etc/login.defs`** command to perform this step. Use `student` as the password when **sudo** prompts you to enter the `student` user's password.

```
...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_MIN_LEN    5
PASS_WARN_AGE   7
```

3. Create the new group called `consultants` with a GID of **35000**.

```
[student@serverb ~]$ sudo groupadd -g 35000 consultants
```

4. Configure administrative rights for all members of `consultants` to be able to execute any command as any user.

   4.1. Create the new file **/etc/sudoers.d/consultants** and add the following content to it. You can use the **sudo vim /etc/sudoers.d/consultants** command to perform this step.

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

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

```
[student@serverb ~]$ sudo useradd -G consultants consultant1
[student@serverb ~]$ sudo useradd -G consultants consultant2
[student@serverb ~]$ sudo useradd -G consultants consultant3
```

6. Set the `consultant1`, `consultant2`, and `consultant3` accounts to expire in 90 days from the current day.

   6.1. Determine the date 90 days in the future. You may get a different value as compared to the following output based on the current date and time of your system.

```
[student@serverb ~]$ date -d "+90 days" +%F
2019-04-28
```

   6.2. Set the account expiry date of the `consultant1`, `consultant2`, and `consultant3` accounts to the same value as determined in the preceding step.

```
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant1
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant2
[student@serverb ~]$ sudo chage -E 2019-04-28 consultant3
```

7. Change the password policy for the `consultant2` account to require a new password every 15 days.

```
[student@serverb ~]$ sudo chage -M 15 consultant2
```

8. Additionally, force the `consultant1`, `consultant2`, and `consultant3` users to change their passwords on the first login.

   8.1. Set the last day of the password change to **0** so that the users are forced to change the password whenever they log in to the system for the first time.

```
[student@serverb ~]$ sudo chage -d 0 consultant1
[student@serverb ~]$ sudo chage -d 0 consultant2
[student@serverb ~]$ sudo chage -d 0 consultant3
```

   8.2. Log off from `serverb`.

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

Evaluation

On `workstation`, run the **`lab users-review grade`** command to confirm success of this exercise.

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

Finish

On `workstation`, run **`lab users-review finish`** to complete this lab. This script deletes the user accounts and files created throughout the lab to ensure that the environment is clean.

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

This concludes the lab.

SUMMARY

In this chapter, you learned:

- There are three main types of user account: the superuser, system users, and regular users.
- A user must have a primary group and may be a member of one or more supplementary groups.
- The three critical files containing user and group information are **`/etc/passwd`**, **`/etc/group`**, and **`/etc/shadow`**.
- The **su** and **sudo** commands can be used to run commands as the superuser.
- The **`useradd`**, **`usermod`**, and **`userdel`** commands can be used to manage users.
- The **`groupadd`**, **`groupmod`**, and **`groupdel`** commands can be used to manage groups.
- The **chage** command can be used to configure and view password expiration settings for users.

CHAPTER 4

CONTROLLING ACCESS TO FILES

GOAL	Set Linux file-system permissions on files and to interpret the security effects of different permission settings.
OBJECTIVES	• Change the permissions and ownership of files using command-line tools. • Control the default permissions of new files created by users, explain the effect of special permissions, and use special permissions and default permissions to set the group owner of files created in a particular directory.
SECTIONS	• Managing File System Permissions from the Command Line (and Guided Exercise) • Managing Default Permissions and File Access (and Guided Exercise)
LAB	Controlling Access to Files

MANAGING FILE SYSTEM PERMISSIONS FROM THE COMMAND LINE

OBJECTIVES

After completing this section, you should be able to change the permissions and ownership of files using command-line tools.

CHANGING FILE AND DIRECTORY PERMISSIONS

The command used to change permissions from the command line is **chmod**, which means "change mode" (permissions are also called the *mode* of a file). The **chmod** command takes a permission instruction followed by a list of files or directories to change. The permission instruction can be issued either symbolically (the symbolic method) or numerically (the numeric method).

Changing Permissions with the Symbolic Method

```
chmod WhoWhatWhich file|directory
```

- *Who* is u, g, o, a *(for user, group, other, all)*
- *What* is +, -, = *(for add, remove, set exactly)*
- *Which* is r, w, x *(for read, write, execute)*

The *symbolic* method of changing file permissions uses letters to represent the different groups of permissions: **u** for user, **g** for group, **o** for other, and **a** for all.

With the symbolic method, it is not necessary to set a complete new group of permissions. Instead, you can change one or more of the existing permissions. Use **+** or **-** to add or remove permissions, respectively, or use **=** to replace the entire set for a group of permissions.

The permissions themselves are represented by a single letter: **r** for read, **w** for write, and **x** for execute. When using **chmod** to change permissions with the symbolic method, using a capital X as the permission flag will add execute permission only if the file is a directory or already has execute set for user, group, or other.

NOTE

The **chmod** command supports the **-R** option to recursively set permissions on the files in an entire directory tree. When using the **-R** option, it can be useful to set permissions symbolically using the **X** option. This allows the execute (search) permission to be set on directories so that their contents can be accessed, without changing permissions on most files. Be cautious with the **X** option, however, because if a file has any execute permission set, **X** will set the specified execute permission on that file as well. For example, the following command recursively sets read and write access on **demodir** and all its children for their group owner, but only applies group execute permissions to directories and files that already have execute set for user, group, or other.

```
[root@host opt]# chmod -R g+rwX demodir
```

Examples

- Remove read and write permission for group and other on **file1**:

```
[user@host ~]$ chmod go-rw file1
```

- Add execute permission for everyone on **file2**:

```
[user@host ~]$ chmod a+x file2
```

Changing Permissions with the Numeric Method

In the example below the # character represents a digit.

```
chmod ### file|directory
```

- Each digit represents permissions for an access level: user, group, other.
- The digit is calculated by adding together numbers for each permission you want to add, 4 for read, 2 for write, and 1 for execute.

Using the *numeric* method, permissions are represented by 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 (numeric) representation of permissions, each digit stands for one access level, from left to right: user, group, and other. To determine each digit:

1. Start with 0.
2. If the read permission should be present for this access level, add 4.
3. If the write permission should be present, add 2.
4. If the execute permission should be present, add 1.

Examine the permissions **-rwxr-x---**. For the user, **rwx** is calculated as 4+2+1=7. For the group, **r-x** is calculated as 4+0+1=5, and for other users, **---** is represented with 0. Putting these three together, the numeric representation of those permissions is 750.

This calculation can also be performed in the opposite direction. Look at the permissions 640. For the user permissions, 6 represents read (4) and write (2), which displays as **rw-**. For the group

part, 4 only includes read (4) and displays as **r--**. The 0 for other provides no permissions (**---**) and the final set of symbolic permissions for this file is **-rw-r-----**.

Experienced administrators often use numeric permissions because they are shorter to type and pronounce, while still giving full control over all permissions.

Examples

- Set read and write permissions for user, read permission for group and other, on **samplefile**:

```
[user@host ~]$ chmod 644 samplefile
```

- Set read, write, and execute permissions for user, read and execute permissions for group, and no permission for other on **sampledir**:

```
[user@host ~]$ chmod 750 sampledir
```

CHANGING FILE AND DIRECTORY USER OR GROUP OWNERSHIP

A newly created file is owned by the user who creates that file. By default, new files have a group ownership that is the primary group of the user creating 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, the group that owns the file may need to be changed.

Only `root` can change the user that owns a file. Group ownership, however, can be set by `root` or by the file's owner. `root` can grant file ownership to any group, but regular users can make a group the owner of a file only if they are a member of that group.

File ownership can be changed with the **chown** (change owner) command. For example, to grant ownership of the **test_file** file to the `student` user, use the following command:

```
[root@host ~]# chown student test_file
```

chown can be used with the **-R** option to recursively change the ownership of an entire directory tree. The following command grants ownership of **test_dir** and all files and subdirectories within it to `student`:

```
[root@host ~]# chown -R student test_dir
```

The **chown** command can also be used to change group ownership of a file by preceding the group name with a colon (:). For example, the following command changes the group **test_dir** to `admins`:

```
[root@host ~]# chown :admins test_dir
```

The **chown** command can also be used to change both owner and group at the same time by using the *owner:group* syntax. For example, to change the ownership of **test_dir** to `visitor` and the group to `guests`, use the following command:

```
[root@host ~]# chown visitor:guests test_dir
```

Instead of using **chown**, some users change the group ownership by using the **chgrp** command. This command works just like **chown**, except that it is only used to change group ownership and the colon (:) before the group name is not required.

IMPORTANT

You may encounter examples of **chown** commands using an alternative syntax that separates owner and group with a period instead of a colon:

```
[root@host ~]# chown owner.group filename
```

You should not use this syntax. Always use a colon.

A period is a valid character in a user name, but a colon is not. If the user `enoch.root`, the user `enoch`, and the group `root` exist on the system, the result of **`chown enoch.root filename`** will be to have **`filename`** owned by the user `enoch.root`. You may have been trying to set the file ownership to the user `enoch` and group `root`. This can be confusing.

If you always use the **chown** colon syntax when setting the user and group at the same time, the results are always easy to predict.

REFERENCES

`ls`(1), **chmod**(1), **chown**(1), and **`chgrp`**(1) man pages

GUIDED EXERCISE

MANAGING FILE SYSTEM PERMISSIONS FROM THE COMMAND LINE

In this exercise, you will use file system permissions to create a directory in which all members of a particular group can add and delete files.

OUTCOMES

You should be able to create a collaborative directory that is accessible by all members of a particular group.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab perms-cli start`** command. The start script creates a group called `consultants` and two users called `consultant1` and `consultant2`.

```
[student@workstation ~]$ lab perms-cli start
```

1. From `workstation`, use the **`ssh`** command to log in to `servera` as the `student` user.

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

2. Switch to the `root` user using `redhat` as the password.

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

3. Use the **`mkdir`** command to create the **`/home/consultants`** directory.

```
[root@servera ~]# mkdir /home/consultants
```

4. Use the **`chown`** command to change the group ownership of the **`consultants`** directory to `consultants`.

```
[root@servera ~]# chown :consultants /home/consultants
```

5. Ensure that the permissions of the `consultants` group allow its group members to create files in, and delete files from the **`/home/consultants`** directory. The permissions should forbid others from accessing the files.

   5.1. Use the **`ls`** command to confirm that the permissions of the `consultants` group allow its group members to create files in, and delete files from the **`/home/consultants`** directory.

```
[root@servera ~]# ls -ld /home/consultants
drwxr-xr-x.  2 root    consultants       6 Feb  1 12:08 /home/
consultants
```

   Note that the `consultants` group currently does not have write permission.

   5.2. Use the **chmod** command to add write permission to the `consultants` group.

```
[root@servera ~]# chmod g+w /home/consultants
[root@servera ~]# ls -ld /home/consultants
drwxrwxr-x. 2 root consultants 6 Feb  1 13:21 /home/consultants
```

   5.3. Use the **chmod** command to forbid others from accessing files in the **`/home/consultants`** directory.

```
[root@servera ~]# chmod 770 /home/consultants
[root@servera ~]# ls -ld /home/consultants
drwxrwx---. 2 root consultants 6 Feb  1 12:08 /home/consultants/
```

6. Exit the root shell and switch to the `consultant1` user. The password is `redhat`.

```
[root@servera ~]# exit
logout
[student@servera ~]$
[student@servera ~]$ su - consultant1
Password: redhat
```

7. Navigate to the **`/home/consultants`** directory and create a file called **`consultant1.txt`**.

   7.1. Use the **cd** command to change to the **`/home/consultants`** directory.

```
[consultant1@servera ~]$ cd /home/consultants
```

   7.2. Use the **touch** command to create an empty file called **`consultant1.txt`**.

```
[consultant1@servera consultants]$ touch consultant1.txt
```

8. Use the **ls -l** command to list the default user and group ownership of the new file and its permissions.

```
[consultant1@servera consultants]$ ls -l consultant1.txt
-rw-rw-r--. 1 consultant1 consultant1 0 Feb  1 12:53 consultant1.txt
```

9. Ensure that all members of the consultants group can edit the **consultant1.txt** file. Change the group ownership of the **consultant1.txt** file to consultants.

   9.1. Use the **chown** command to change the group ownership of the **consultant1.txt** file to consultants.

```
[consultant1@servera consultants]$ chown :consultants consultant1.txt
```

   9.2. Use the **ls** command with the **-l** option to list the new ownership of the **consultant1.txt** file.

```
[consultant1@servera consultants]$ ls -l consultant1.txt
-rw-rw-r--. 1 consultant1 consultants 0 Feb  1 12:53 consultant1.txt
```

10. 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 ~]$
```

11. Navigate to the **/home/consultants** directory. Ensure that the consultant2 user can add content to the **consultant1.txt** file. Exit from the shell.

    11.1. Use the **cd** command to change to the **/home/consultants** directory. Use the **echo** command to add **text** to the **consultant1.txt** file.

    ```
    [consultant2@servera ~]$ cd /home/consultants/
    [consultant2@servera consultants]$ echo "text" >> consultant1.txt
    [consultant2@servera consultants]$
    ```

    11.2. Use the **cat** command to verify that the text was added to the **consultant1.txt** file.

    ```
    [consultant2@servera consultants]$ cat consultant1.txt
    text
    [consultant2@servera consultants]$
    ```

    11.3. Exit the shell.

    ```
    [consultant2@servera consultants]$ exit
    logout
    [student@servera ~]$
    ```

12. Log off from servera.

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

Finish

On workstation, run the **lab perms-cli finish** script to complete this exercise.

```
[student@workstation ~]$ lab perms-cli finish
```

This concludes the guided exercise.

MANAGING DEFAULT PERMISSIONS AND FILE ACCESS

OBJECTIVES

After completing this section, students should be able to:

- Control the default permissions of new files created by users.
- Explain the effect of special permissions.
- Use special permissions and default permissions to set the group owner of files created in a particular directory.

SPECIAL PERMISSIONS

Special permissions constitute a fourth permission type in addition to the basic user, group, and other types. As the name implies, these permissions provide additional access-related features over and above what the basic permission types allow. This section details the impact of special permissions, summarized in the table below.

Effects of Special Permissions on Files and Directories

SPECIAL PERMISSION	EFFECT ON FILES	EFFECT ON DIRECTORIES
u+s *(suid)*	File executes as the user that owns the file, not the user that ran the file.	No effect.
g+s *(sgid)*	File executes as the group that owns the file.	Files newly created in the directory have their group owner set to match the group owner of the directory.
o+t *(sticky)*	No effect.	Users with write access to the directory can only remove files that they own; they cannot remove or force saves to files owned by other users.

The *setuid* permission on an executable file means that commands run as the user owning the file, not 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** where you would normally expect the **x** (owner execute permissions) to be. If the owner does not have execute permissions, this is replaced by an uppercase **S**.

The special permission *setgid* on a directory means that files created in the directory inherit their group ownership from the directory, rather than inheriting it from the creating user. This is commonly used on group collaborative directories to automatically change a file from the default

private group to the shared group, or if files in a directory should be always owned by a specific group. An example of this 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, commands run as the group that owns that file, not as the user that ran the command, in a similar way to 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** where you would normally expect the **x** (group execute permissions) to be. If the group does not have execute permissions, this is replaced by an uppercase **S**.

Lastly, the *sticky bit* for a directory sets a special restriction on deletion of files. Only the owner of the file (and `root`) can delete files within the directory. An example is **/tmp**:

```
[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** where you would normally expect the **x** (other execute permissions) to be. If other does not have execute permissions, this is replaced by an uppercase **T**.

Setting Special Permissions

- Symbolically: setuid = **u+s**; setgid = **g+s**; sticky = **o+t**
- Numerically (fourth preceding digit): setuid = 4; setgid = 2; sticky = 1

Examples

- Add the setgid bit on **directory**:

```
[user@host ~]# chmod g+s directory
```

- Set the setgid bit and add read/write/execute permissions for user and group, with no access for others, on **directory**:

```
[user@host ~]# chmod 2770 directory
```

DEFAULT FILE PERMISSIONS

When you create a new file or directory, it is assigned initial permissions. There are two things that affect these initial permissions. The first is whether you are creating a regular file or a directory. The second is the current *umask*.

If you create a new directory, the operating system starts by assigning it octal permissions 0777 (**drwxrwxrwx**). If you create a new regular file, the operating system assignes it octal permissions 0666 (**-rw-rw-rw-**). You always have to explicitly add execute permission to a regular file. This

makes it harder for an attacker to compromise a network service so that it creates a new file and immediately executes it as a program.

However, the shell session will also set a umask to further restrict the permissions that are initially set. This is an octal bitmask used to clear the permissions of new files and directories created by a process. 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 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 will display the current value of the shell's umask:

```
[user@host ~]$ umask
0002
```

Use the **umask** command with a single numeric argument to change the umask of the current shell. The numeric argument should be an octal value corresponding to the new umask value. You can omit any leading zeros in the umask.

The system's default umask values for Bash shell users are defined in the **/etc/profile** and **/etc/bashrc** files. Users can override the system defaults in the **.bash_profile** and **.bashrc** files in their home directories.

umask Example

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. The owner and group both have read and write permission on files, and other is set to read. The owner and group both have read, write, and execute permissions on directories. The only permission for other is read.

```
[user@host ~]$ umask
0002
[user@host ~]$ touch default
[user@host ~]$ ls -l default.txt
-rw-rw-r--. 1 user user 0 May  9 01:54 default.txt
[user@host ~]$ mkdir default
[user@host ~]$ ls -ld default
drwxrwxr-x. 2 user user 0 May  9 01:54 default
```

By setting the umask value to 0, the file permissions for other change from read to read and write. The directory permissions for other changes from read and execute to read, write, and execute.

```
[user@host ~]$ umask 0
[user@host ~]$ touch zero.txt
[user@host ~]$ ls -l zero.txt
-rw-rw-rw-. 1 user user 0 May  9 01:54 zero.txt
[user@host ~]$ mkdir zero
[user@host ~]$ ls -ld zero
drwxrwxrwx. 2 user user 0 May  9 01:54 zero
```

To mask all file and directory permissions for other, set the umask value to 007.

```
[user@host ~]$ umask 007
[user@host ~]$ touch seven.txt
[user@host ~]$ ls -l seven.txt
-rw-rw----. 1 user user 0 May  9 01:55 seven.txt
[user@host ~]$ mkdir seven
[user@host ~]$ ls -ld seven
drwxrwx---. 2 user user 0 May  9 01:54 seven
```

A umask of 027 ensures that new files have read and write permissions for user and read permission for group. New directories have read and write access for group and no permissions for other.

```
[user@host ~]$ umask 027
[user@host ~]$ touch two-seven.txt
[user@host ~]$ ls -l two-seven.txt
-rw-r-----. 1 user user 0 May  9 01:55 two-seven.txt
[user@host ~]$ mkdir two-seven
[user@host ~]$ ls -ld two-seven
drwxr-x---. 2 user user 0 May  9 01:54 two-seven
```

The default umask for users is set by the shell startup scripts. By default, if your account's UID is 200 or more and your username and primary group name are the same, you will be assigned a umask of 002. Otherwise, your umask will be 022.

As `root`, you can change this by adding a shell startup script named **`/etc/profile.d/local-umask.sh`** that looks something like the output in this example:

```
[root@host ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

The preceding example will set the umask to 007 for users with a UID greater than 199 and with a username and primary group name that match, and to 022 for everyone else. If you just wanted to set the umask for everyone to 022, you could create that file with just the following content:

```
# Overrides default umask configuration
umask 022
```

To ensure that global umask changes take effect you must log out of the shell and log back in. Until that time the umask configured in the current shell is still in effect.

REFERENCES

bash(1), **ls**(1), **chmod**(1), and **umask**(1) man pages

GUIDED EXERCISE

MANAGING DEFAULT PERMISSIONS AND FILE ACCESS

In this exercise, you will control the permissions on new files created in a directory by using umask settings and the setgid permission.

OUTCOMES

You should be able to:

- Create a shared directory where new files are automatically owned by the `operators` group.
- Experiment with various umask settings.
- Adjust default permissions for specific users.
- Confirm your adjustment is correct.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab perms-default start`** command. The command runs a start script that determines if `servera` is reachable on the network. The script also creates the `operators` group and the `operator1` user on `servera`.

```
[student@workstation ~]$ lab perms-default start
```

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

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

2. Use the **su** command to switch to the `operator1` user using `redhat` as the password.

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

3. Use the **umask** command to list the `operator1` user's default umask value.

```
[operator1@servera ~]$ umask
0002
```

4. Create a new directory named **/tmp/shared**. In the **/tmp/shared** directory, create a file named **defaults**. Look at the default permissions.

4.1. Use the **mkdir** command to create the **/tmp/shared** directory. Use the **ls -ld** command to list the permissions of the new directory.

```
[operator1@servera ~]$ mkdir /tmp/shared
[operator1@servera ~]$ ls -ld /tmp/shared
drwxrwxr-x. 2 operator1 operator1 6 Feb  4 14:06 /tmp/shared
```

4.2. Use the **touch** command to create a file named **defaults** in the **/tmp/shared** directory.

```
[operator1@servera ~]$ touch /tmp/shared/defaults
```

4.3. Use the **ls -l** command to list the permissions of the new file.

```
[operator1@servera ~]$ ls -l /tmp/shared/defaults
-rw-rw-r--. 1 operator1 operator1 0 Feb  4 14:09 /tmp/shared/defaults
```

5. Change the group ownership of **/tmp/shared** to `operators`. Confirm the new ownership and permissions.

5.1. Use the **chown** command to change the group ownership of the **/tmp/shared** directory to `operators`.

```
[operator1@servera ~]$ chown :operators /tmp/shared
```

5.2. Use the **ls -ld** command to list the permissions of the **/tmp/shared** directory.

```
[operator1@servera ~]$ ls -ld /tmp/shared
drwxrwxr-x. 2 operator1 operators 22 Feb  4 14:09 /tmp/shared
```

5.3. Use the **touch** command to create a file named **group** in the **/tmp/shared** directory. Use the **ls -l** command to list the file permissions.

```
[operator1@servera ~]$ touch /tmp/shared/group
[operator1@servera ~]$ ls -l /tmp/shared/group
-rw-rw-r--. 1 operator1 operator1 0 Feb  4 17:00 /tmp/shared/group
```

NOTE

The group owner of the **/tmp/shared/group** file is not `operators` but `operator1`.

6. Ensure that files created in the **/tmp/shared** directory are owned by the operators group.

   6.1. Use the **chmod** command to set the group ID to the operators group for the **/tmp/shared** directory.

```
[operator1@servera ~]$ chmod g+s /tmp/shared
```

   6.2. Use the **touch** command to create a new file named **operations_database.txt** in the **/tmp/shared** directory.

```
[operator1@servera ~]$ touch /tmp/shared/operations_database.txt
```

   6.3. Use the **ls -l** command to verify that the operators group is the group owner for the new file.

```
[operator1@servera ~]$ ls -l /tmp/shared/operations_database.txt
-rw-rw-r--. 1 operator1 operators 0 Feb  4 16:11 /tmp/shared/
operations_database.txt
```

7. Create a new file in the **/tmp/shared** directory named **operations_network.txt**. Record the ownership and permissions. Change the umask for operator1. Create a new file called **operations_production.txt**. Record the ownership and permissions of the **operations_production.txt** file.

   7.1. Use the **echo** command to create a file called **operations_network.txt** in the **/tmp/shared** directory.

```
[operator1@servera ~]$ echo text >> /tmp/shared/operations_network.txt
```

   7.2. Use the **ls -l** command to list the permissions of the **operations_network.txt** file.

```
[operator1@servera ~]$ ls -l /tmp/shared/operations_network.txt
-rw-rw-r--. 1 operator1 operators 5 Feb  4 15:43 /tmp/shared/
operations_network.txt
```

   7.3. Use the **umask** command to change the umask for the operator1 user to 027. Use the **umask** command to confirm the change.

```
[operator1@servera ~]$ umask 027
[operator1@servera ~]$ umask
0027
```

   7.4. Use the **touch** command to create a new file named **operations_production.txt** in the **/tmp/shared/** directory. Use the **ls -l**

command to ensure that newly created files are created with read-only access for the `operators` group and no access for other users.

```
[operator1@servera ~]$ touch /tmp/shared/operations_production.txt
[operator1@servera ~]$ ls -l /tmp/shared/operations_production.txt
-rw-r-----. 1 operator1 operators 0 Feb  4 15:56 /tmp/shared/
operations_production.txt
```

8. Open a new terminal window and log in to `servera` as `operator1`.

```
[student@workstation ~]$ ssh operator1@servera
...output omitted...
[operator1@servera ~]$
```

9. List the umask value for `operator1`.

```
[operator1@servera ~]$ umask
0002
```

10. Change the default umask for the `operator1` user. The new umask prohibits all access for users not in their group. Confirm that the umask has been changed.

10.1. Use the **echo** command to 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

# Uncomment the following line if you don't like systemctl's auto-
paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
```

10.2. Log out and log in again as the `operator1` user. Use the **umask** command to confirm that the change is permanent.

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh operator1@servera
...output omitted...
[operator1@servera ~]$ umask
0007
```

11. On `servera`, exit from all the `operator1` and the `student` user shells.

> **WARNING**
> Exit from all shells opened by `operator1`. Failure to exit from all shells will cause the finish script to fail.

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

Finish

On `workstation`, run the **`lab perms-default finish`** script to complete this exercise.

```
[student@workstation ~]$ lab perms-default finish
```

This concludes the guided exercise.

LAB

CONTROLLING ACCESS TO FILES

PERFORMANCE CHECKLIST

In this lab, you will configure permissions on files and set up a directory that users in a particular group can use to conveniently share files on the local file system.

OUTCOMES

You should be able to:

- 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 of the group.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab perms-review start`** command. The command runs a start script that determines if `serverb` is reachable on the network. The script also creates the `techdocs` group and three users named `tech1`, `tech2`, and `database1`.

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

1. Use the **ssh** command to log in to `serverb` as the `student` user. Switch to `root` on `serverb` using `redhat` as the password.
2. Create a directory called **`/home/techdocs`**.
3. Change the group ownership of the **`/home/techdocs`** directory to the `techdocs` group.
4. Verify that users in the `techdocs` group can create and edit files in the `/home/techdocs` directory.
5. Set permissions on the **`/home/techdocs`** directory. On the **`/home/techdocs`** directory, configure setgid (2), read/write/execute permissions (7) for the owner/user and group, and no permissions (0) for other users.
6. Verify that the permissions are set properly.
7. Confirm that users in the `techdocs` group can now create and edit files in the `/home/techdocs` directory. Users not in the `techdocs` group cannot edit or create files in the `/home/techdocs` directory. Users `tech1` and `tech2` are in the `techdocs` group. User `database1` is not in that group.
8. Modify the global login scripts. Normal users should have a umask setting that prevents others from viewing or modifying new files and directories.
9. Log off from `serverb`.

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

Evaluation

On `workstation`, run the **`lab perms-review grade`** script to confirm success on this exercise.

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

Finish

On `workstation`, run the **`lab perms-review finish`** script to complete the lab.

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

This concludes the lab.

▸ SOLUTION

CONTROLLING ACCESS TO FILES

PERFORMANCE CHECKLIST

In this lab, you will configure permissions on files and set up a directory that users in a particular group can use to conveniently share files on the local file system.

OUTCOMES

You should be able to:

- 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 of the group.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab perms-review start`** command. The command runs a start script that determines if `serverb` is reachable on the network. The script also creates the `techdocs` group and three users named `tech1`, `tech2`, and `database1`.

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

1. Use the **ssh** command to log in to `serverb` as the `student` user. Switch to `root` on `serverb` using `redhat` as the password.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ su -
Password: redhat
[root@serverb ~]#
```

2. Create a directory called **`/home/techdocs`**.

   2.1. Use the **`mkdir`** command to create a directory called **`/home/techdocs`**.

```
[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 can create and edit 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. Use **touch** to create a file named **`techdoc1.txt`** in the **`/home/techdocs`** directory.

```
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt
touch: cannot touch '/home/techdocs/techdoc1.txt': Permission denied
```

NOTE

Note that even though the **`/home/techdocs`** directory is owned by `techdocs` and `tech1` is part of the `techdocs` group, it is not possible to create a new file in that directory. This is because the `techdocs` group does not have write permission. Use the **`ls -ld`** command to show the 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/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. Use the **chmod** command to set the group permission for the **`/home/techdocs`** directory. On the **`/home/techdocs`** directory, configure setgid (2), read/write/execute permissions (7) for the owner/user and group, and no permissions (0) for other users.

```
[root@serverb ~]# chmod 2770 /home/techdocs
```

6. Verify that the permissions are set properly.

```
[root@serverb ~]# ls -ld /home/techdocs
drwxrws---. 2 root techdocs 6 Feb 4 18:12 /home/techdocs/
```

   Note that 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 not in the `techdocs` group cannot edit or create files in the

/home/techdocs directory. Users tech1 and tech2 are in the techdocs group. User database1 is not in that group.

7.1. Switch to the tech1 user. Use **touch** to create a file called techdoc1.txt in the /home/techdocs directory. Exit from the tech1 user shell.

```
[root@serverb ~]# su - tech1
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
-rw-rw-r--. 1 tech1 techdocs 0 Feb  5 16:42 /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ exit
logout
[root@serverb ~]#
```

7.2. Switch to the tech2 user. Use the **echo** command to add some content to the **/home/techdocs/techdoc1.txt** file. Exit from the tech2 user shell.

```
[root@serverb ~]# su - tech2
[tech2@serverb ~]$ cd /home/techdocs
[tech2@serverb techdocs]$ echo "This is the first tech doc." >
 techdoc1.txt
[tech2@serverb techdocs]$ exit
logout
[root@serverb ~]#
```

7.3. Switch to the database1 user. Use the **echo** command to append some content to the **/home/techdocs/techdoc1.txt** file. Notice that you will get a **Permission Denied** message. Use the **ls -l** command to confirm that database1 does not have access to the file. Exit from the database1 user shell.

```
[root@serverb ~]# su - database1
[database1@serverb ~]$ echo "This is the first tech doc." \
>> /home/techdocs/techdoc1.txt
-bash: /home/techdocs/techdoc1.txt: Permission denied
[database1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
ls: cannot access '/home/techdocs/techdoc1.txt': Permission denied
[database1@serverb ~]$ exit
logout
[root@serverb ~]#
```

8. Modify the global login scripts. Normal users should have a umask setting that prevents others from viewing or modifying new files and directories.

8.1. Determine the umask of the `student` user. Use the **`su - student`** command to switch to `student` login shell. When done exit from the shell.

```
[root@serverb ~]# su - student
[student@serverb ~]$ umask
0002
[student@serverb ~]$ exit
logout
[root@serverb ~]#
```

8.2. Create the **`/etc/profile.d/local-umask.sh`** file with the following content to set the umask to **`007`** for users with a UID greater than **`199`** and with a username and primary group name that match, and to **`022`** for everyone else:

```
[root@serverb ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

8.3. Log out of the shell and log back in as `student` to verify that 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
```

9. Log off from `serverb`.

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

Evaluation

On `workstation`, run the **`lab perms-review grade`** script to confirm success on this exercise.

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

Finish

On `workstation`, run the **`lab perms-review finish`** script to complete the lab.

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

SUMMARY

In this chapter, you learned:

- Files have three categories to which permissions apply. 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 with the **`-l`** option expands the file listing to include both the file permissions and ownership.
- The **chmod** command changes file permissions from the command line. There are two methods to represent permissions, symbolic (letters) and numeric (digits).
- The **chown** command changes file ownership. The **`-R`** option recursively changes the ownership of a directory tree.
- The **umask** command without arguments displays the current umask value of the shell. Every process on the system has a umask. The default umask values for Bash are defined in the **`/etc/profile`** and **`/etc/bashrc`** files.

CHAPTER 5

MANAGING SELINUX SECURITY

GOAL	Protect and manage the security of a server by using SELinux.
OBJECTIVES	• Describe how SELinux works and how to switch a server between its various enforcment modes. • Adjust the SELinux type of a file in order to control which processes can access it. • Change the accesses allowed by the SELinux policy by setting tunable parameters called SELinux booleans. • Perform basic investigation and troubleshooting of accesses blocked by SELinux.
SECTIONS	• Changing the SELinux Enforcement Mode (and Guided Exercise) • Controlling SELinux File Contexts (and Guided Exercise) • Adjusting SELinux Policy with Booleans (and Guided Exercise) • Investigating and Resolving SELinux Issues (and Guided Exercise)
LAB	Managing SELinux Security

CHANGING THE SELINUX ENFORCEMENT MODE

OBJECTIVES

After completing this section, you should be able to:

- Explain how SELinux protects resources.
- Change the current SELinux mode of a system.
- Set the default SELinux mode of a system.

HOW SELINUX PROTECTS RESOURCES

SELinux provides a critical security purpose in Linux, permitting or denying access to files and other resources that are significantly more precise than user permissions.

File permissions control which users or groups of users can access which specific files. However, a user given read or write access to any specific file can use that file in any way that user chooses, even if that use is not how the file should be used.

For example, with write access to a file, should a structured data file designed to be written to using only a particular program, be allowed to be opened and modified by other editors that could result in corruption?

File permissions cannot stop such undesired access. They were never designed to control *how* a file is used, but only *who* is allowed to read, write, or run a file.

SELinux consists of sets of policies, defined by the application developers, that declare exactly what actions and accesses are proper and allowed for each binary executable, configuration file, and data file used by an application. This is known as a *targeted policy* because one policy is written to cover the activities of a single application. Policies declare predefined labels that are placed on individual programs, files, and network ports.

WHY USE SECURITY ENHANCED LINUX?

Not all security issues can be predicted in advance. SELinux enforces a set of access rules preventing a weakness in one application from affecting other applications or the underlying system. SELinux provides an extra layer of security; it also adds a layer of complexity which can be off-putting to people new to this subsystem. Learning to work with SELinux may take time but the enforcement policy means that a weakness in one part of the system does not spread to other parts. If SELinux works poorly with a particular subsystem, you can turn off enforcement for that specific service until you find a solution to the underlying problem.

SELinux has three modes:

- Enforcing: SELinux is enforcing access control rules. Computers generally run in this mode.
- Permissive: SELinux is active but instead of enforcing access control rules, it records warnings of rules that have been violated. This mode is used primarily for testing and troubleshooting.
- Disabled: SELinux is turned off entirely: no SELinux violations are denied, nor even recorded. Discouraged!

BASIC SELINUX SECURITY CONCEPTS

Security Enhanced Linux (SELinux) is an additional layer of system security. The primary goal of SELinux is to protect user data from system services that have been compromised. Most Linux administrators are familiar with the standard user/group/other permission security model. This is a user and group based model known as discretionary access control. SELinux provides an additional layer of security that is object-based and controlled by more sophisticated rules, known as mandatory access control.

To allow remote anonymous access to a web server, firewall ports must be opened. However, this gives malicious people an opportunity to crack the system through a security exploit. If they succeed in compromising the web server process they gain its permissions. Specifically, the permissions of the `apache` user and the `apache` group. That user and group has read access to the document root, **`/var/www/html`**. It also has access to **`/tmp`**, and **`/var/tmp`**, and any other files and directories that are world writable.

SELinux is a set of security rules that determine which process can access which files, directories, and ports. Every file, process, directory, and port has a special security label called an SELinux *context*. A context is a name used by the SELinux policy to determine whether a process can access a file, directory, or port. By default, the policy does not allow any interaction unless an explicit rule grants access. If there is no allow rule, no access is allowed.

SELinux labels have several contexts: `user`, `role`, `type`, and `sensitivity`. The targeted policy, which is the default policy enabled in Red Hat Enterprise Linux, bases its rules on the third context: the type context. Type context names usually end with **`_t`**.

```
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
|            |         |                  |    |
SELinux User Role      Type               Level File
```

Figure 5.1: SELinux File Context

The type context for a web server is **`httpd_t`**. The type context for files and directories normally found in **`/var/www/html`** is **`httpd_sys_content_t`**. The contexts for files and directories normally found in **`/tmp`** and **`/var/tmp`** is **`tmp_t`**. The type context for web server ports is **`http_port_t`**.

Apache has a type context of **`httpd_t`**. There is a policy rule that permits Apache access to files and directories with the **`httpd_sys_content_t`** type context. By default files found in **`/var/www/html`** and other web server directories have the **`httpd_sys_content_t`** type context. There is no **`allow`** rule in the policy for files normally found in **`/tmp`** and **`/var/tmp`**, so access is not permitted. With SELinux enabled, a malicious user who had compromised the web server process could not access the **`/tmp`** directory.

The `MariaDB` server has a type context of **`mysqld_t`**. By default, files found in **`/data/mysql`** have the **`mysqld_db_t`** type context. This type context allows `MariaDB` access to those files but disables access by other services, such as the Apache web service.

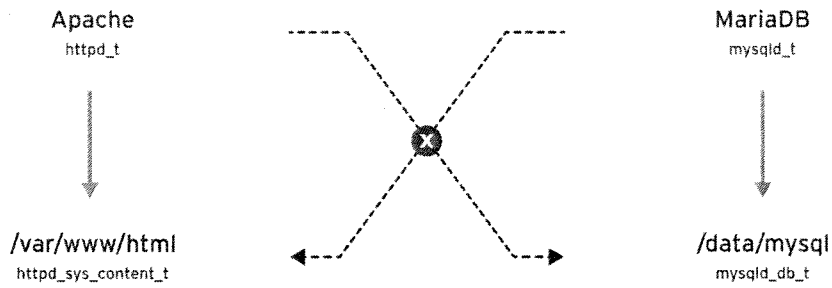


Figure 5.2: SELinux access

Many commands that deal with files use the **-Z** option to display or set SELinux contexts. For instance, **ps**, **ls**, **cp**, and **mkdir** all use the **-Z** option to display or set SELinux contexts.

```
[root@host ~]# ps axZ
LABEL                              PID TTY      STAT   TIME COMMAND
system_u:system_r:init_t:s0          1 ?        Ss     0:09 /usr/lib/systemd/...
system_u:system_r:kernel_t:s0        2 ?        S      0:00 [kthreadd]
system_u:system_r:kernel_t:s0        3 ?        S      0:00 [ksoftirqd/0]
...output omitted...
[root@host ~]# systemctl start httpd
[root@host ~]# ps -ZC httpd
LABEL                              PID TTY          TIME CMD
system_u:system_r:httpd_t:s0      1608 ?        00:00:05 httpd
system_u:system_r:httpd_t:s0      1609 ?        00:00:00 httpd
...output omitted...
[root@host ~]# ls -Z /home
drwx------. root    root    system_u:object_r:lost_found_t:s0 lost+found
drwx------. student student unconfined_u:object_r:user_home_dir_t:s0 student
drwx------. visitor visitor unconfined_u:object_r:user_home_dir_t:s0 visitor
[root@host ~]# ls -Z /var/www
drwxr-xr-x. root root system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 error
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 html
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 icons
```

CHANGING THE CURRENT SELINUX MODE

The SELinux subsystem provides tools to display and change modes. To determine the current SELinux mode, run the **getenforce** command. To set SELinux to a different mode, use the **setenforce** command:

```
[user@host ~]# getenforce
Enforcing
[user@host ~]# setenforce
usage:  setenforce [ Enforcing | Permissive | 1 | 0 ]
[user@host ~]# setenforce 0
[user@host ~]# getenforce
Permissive
```

```
[user@host ~]# setenforce Enforcing
[user@host ~]# getenforce
Enforcing
```

Alternatively, you can set the SELinux mode at boot time by passing a parameter to the kernel: the kernel argument of **enforcing=0** boots the system into permissive mode; a value of **enforcing=1** sets enforcing mode. You can also disable SELinux completely by passing on the kernel parameter **selinux=0**. A value of **selinux=1** enables SELinux.

SETTING THE DEFAULT SELINUX MODE

You can also configure SELinux persistently using the **/etc/selinux/config** file. In the example below (the default configuration), the configuration file sets SELinux to **enforcing**. The comments also show the other valid values: **permissive** and **disabled**.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes
#               are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

The system reads this file at boot time and configures SELinux as shown. Kernel arguments (**selinux=0|1** and **enforcing=0|1**) override this configuration.

REFERENCES

getenforce(8), **setenforce**(8), and **selinux_config**(5) man pages

GUIDED EXERCISE

CHANGING THE SELINUX ENFORCEMENT MODE

In this lab, you will manage SELinux modes, both temporarily and persistently.

OUTCOMES

You should be able to view and set the current SELinux mode.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-opsmode start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network.

```
[student@workstation ~]$ lab selinux-opsmode start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

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

3. Change the default SELinux mode to permissive and reboot.

   3.1. Use the **`getenforce`** command to verify that `servera` is in enforcing mode.

```
[root@servera ~]# getenforce
Enforcing
```

   3.2. Use the **`vim`** command to open the **`/etc/selinux/config`** configuration file. Change the `SELINUX` parameter from `enforcing` to `permissive`.

```
[root@servera ~]# vim /etc/selinux/config
```

3.3. Use the **grep** command to confirm that the SELINUX parameter is set to permissive.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=permissive
SELINUXTYPE=targeted
```

3.4. Use the **systemctl reboot** command to reboot servera.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

4. servera takes a few minutes to reboot. After a few minutes, log in to servera as the student user. Use the **sudo -i** command to become root. Display the current SELinux mode using the **getenforce** command.

4.1. From workstation using the **ssh** command log in to servera as the student user.

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

4.2. Use the **sudo -i** command to become root.

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

4.3. Display the current SELinux mode using the **getenforce** command.

```
[root@servera ~]# getenforce
Permissive
```

5. In the **/etc/selinux/config** file, change the default SELinux mode to enforcing. This change only takes effect on next reboot.

   5.1. Use the **vim** command to open the **/etc/selinux/config** configuration file. Change the SELINUX back to enforcing.

```
[root@servera ~]# vim /etc/selinux/config
```

   5.2. Use the **grep** command to confirm that the SELINUX parameter is set to enforcing.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

6. Use the **setenforce** command to set the current SELinux mode to enforcing without rebooting. Confirm that the mode has been set to enforcing using the **getenforce** command.

```
[root@servera ~]# setenforce 1
[root@servera ~]# getenforce
Enforcing
```

7. Exit from servera.

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

Finish

On workstation, run the **lab selinux-opsmode finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-opsmode finish
```

This concludes the guided exercise.

CONTROLLING SELINUX FILE CONTEXTS

OBJECTIVES

After completing this section, you should be able to:

- Manage the SELinux policy rules that determine the default context for files and directories using the **semanage fcontext** command.
- Apply the context defined by the SELinux policy to files and directories using the **restorecon** command.

INITIAL SELINUX CONTEXT

On systems running SELinux, all processes and files are labeled. The label represents the security relevant information, known as the SELinux context.

New files typically inherit their SELinux context from the parent directory, thus ensuring that they have the proper context.

But this inheritance procedure can be undermined in two different ways. First, if you create a file in a different location from the ultimate intended location and then move the file, the file still has the SELinux context of the directory where it was created, not the destination directory. Second, if you copy a file preserving the SELinux context, as with the **cp -a** command, the SELinux context reflects the location of the original file.

The following example demonstrates inheritance and its pitfalls. Consider these two files created in **/tmp**, one moved to **/var/www/html** and the second one copied to the same directory. Note the SELinux contexts on the files. The file that was moved to the **/var/www/html** directory retains the file context for the **/tmp** directory. The file that was copied to the **/var/www/html** directory inherited SELinux context from the **/var/www/html** directory.

The **ls -Z** command displays the SELinux context of a file. Note the label of the file.

```
[root@host ~]# ls -Z /var/www/html/index.html
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/
index.html
```

And the **ls -Zd** command displays the SELinux context of a directory:

```
[root@host ~]# ls -Zd /var/www/html/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

Note that the **/var/www/html/index.html** has the same label as the parent directory **/var/www/html/**. Now, create files outside of the **/var/www/html** directory and note their file context:

```
[root@host ~]# touch /tmp/file1 /tmp/file2
[root@host ~]# ls -Z /tmp/file*
unconfined_u:object_r:user_tmp_t:s0 /tmp/file1
unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
```

Move one of these files to the **/var/www/html** directory, copy another, and note the label of each:

```
[root@host ~]# mv /tmp/file1 /var/www/html/
[root@host ~]# cp /tmp/file2 /var/www/html/


[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The moved file maintains its original label while the copied file inherits the label from the **/var/www/html** directory. **unconfined_u:** is the user, **object_r:** denotes the role, and **s0** is the level. A sensitivity level of 0 is the lowest possible sensitivity level.

CHANGING THE SELINUX CONTEXT OF A FILE

Commands to change the SELinux context on files include **semanage fcontext**, **restorecon**, and **chcon**.

The preferred method to set the SELinux context for a file is to declare the default labeling for a file using the **semanage fcontext** command and then applying that context to the file using the **restorecon** command. This ensures that the labeling will be as desired even after a complete relabeling of the file system.

The **chcon** command changes SELinux contexts. **chcon** sets the security context on the file, stored in the file system. It is useful for testing and experimenting. However, it does not save context changes in the SELinux context database. When a **restorecon** command runs, changes made by the **chcon** command also do not survive. Also, if the entire file system is relabeled, the SELinux context for files changed using **chcon** are reverted.

The following screen shows a directory being created. The directory has a type value of **default_t**.

```
[root@host ~]# mkdir /virtual
[root@host ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual
```

The **chcon** command changes the file context of the **/virtual** directory: the type value changes to **httpd_sys_content_t**.

```
[root@host ~]# chcon -t httpd_sys_content_t /virtual
[root@host ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:httpd_sys_content_t:s0 /virtual
```

The **restorecon** command runs and the type value returns to the value of **default_t**. Note the **Relabeled** message.

```
[root@host ~]# restorecon -v /virtual
Relabeled /virtual from unconfined_u:object_r:httpd_sys_content_t:s0 to
 unconfined_u:object_r:default_t:s0
[root@host ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual
```

DEFINING SELINUX DEFAULT FILE CONTEXT RULES

The **semanage fcontext** command displays and modifies the rules that **restorecon** uses to set default file contexts. It uses extended regular expressions to specify the path and file names. The most common extended regular expression used in **fcontext** rules is **(/.*)?**, which means "optionally, match a / followed by any number of characters". It matches the directory listed before the expression and everything in that directory recursively.

Basic File Context Operations

The following table is a reference for **semanage fcontext** options to add, remove or list SELinux file contexts.

semanage fcontext commands

OPTION	DESCRIPTION
-a, --add	Add a record of the specified object type
-d, --delete	Delete a record of the specified object type
-l, --list	List records of the specified object type

To ensure that you have the tools to manage SELinux contexts, install the **policycoreutil** package and the **policycoreutil-python** package if needed. These contain the **restorecon** command and **semanage** command, respectively.

To ensure that all files in a directory have the correct file context run the **semanage fcontext -l** followed by the **restorecon** command. In the following example, note the file context of each file before and after the **semanage** and **restorecon** commands run.

```
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
 unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2


[root@host ~]# semanage fcontext -l
...output omitted...
/var/www(/.*)?       all files    system_u:object_r:httpd_sys_content_t:s0
...output omitted...
```

```
[root@host; ~]# restorecon -Rv /var/www/
Relabeled /var/www/html/file1 from unconfined_u:object_r:user_tmp_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
 unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The following example shows how to use **semanage** to add a context for a new directory.

```
[root@host ~]# mkdir /virtual
[root@host ~]# touch /virtual/index.html
[root@host ~]# ls -Zd /virtual/
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual/


[root@host ~]# ls -Z /virtual/
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 index.html
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.*)?'
[root@host ~]# restorecon -RFvv /virtual
[root@host ~]# ls -Zd /virtual/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/
[root@host ~]# ls -Z /virtual/
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 index.html
```

REFERENCES

chcon(1), **restorecon**(8), **semanage**(8), and **semanage-fcontext**(8) man pages

GUIDED EXERCISE

CONTROLLING SELINUX FILE CONTEXTS

In this lab, you will make a persistent change to the SELinux context of a directory and its contents.

OUTCOMES

You should be able to configure the *Apache* HTTP server to publish web content from a non-standard document root.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-filecontexts start`** command. This command runs a start script that determines whether the `servera` machine is reachable on the network. It also installs the `httpd` service and configures the firewall on `servera` to allow HTTP connections.

```
[student@workstation ~]$ lab selinux-filecontexts start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

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

3. Configure Apache to use a document root in a non-standard location.

   3.1. Create the new document root, **`/custom`** using the **`mkdir`** command.

```
[root@servera ~]# mkdir /custom
```

   3.2. Create the **`index.html`** file in the **`/custom`** document root using the **`echo`** command.

```
[root@servera ~]# echo 'This is SERVERA.' > /custom/index.html
```

3.3. Configure Apache to use the new document root location. You need to replace the two occurrences of **/var/www/html** with **/custom** in the Apache configuration file, **/etc/httpd/conf/httpd.conf**.

```
[root@servera ~]# vim /etc/httpd/conf/httpd.conf
[root@servera ~]# grep custom /etc/httpd/conf/httpd.conf
DocumentRoot "/custom"
<Directory "/custom">
```

4. Start and enable the Apache web service and confirm that the service is running.

4.1. Start and enable the Apache web service using the **systemctl** command.

```
[root@servera ~]# systemctl enable --now httpd
```

4.2. Use the **systemctl** command to confirm that the service is running.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
 vendor preset: disabled)
   Active: active (running) since Mon 2019-03-25 19:16:48 CET; 15h ago
     Docs: man:httpd.service(8)
 Main PID: 6565 (httpd)
   Status: "Total requests: 16; Idle/Busy workers 100/0;Requests/sec:
 0.000285; Bytes served/sec:   0 B/sec"
   Tasks: 213 (limit: 11406)
   Memory: 37.3M
   CGroup: /system.slice/httpd.service
           ├─6565 /usr/sbin/httpd -DFOREGROUND
           ├─6566 /usr/sbin/httpd -DFOREGROUND
           ├─6567 /usr/sbin/httpd -DFOREGROUND
           ├─6568 /usr/sbin/httpd -DFOREGROUND
           └─6569 /usr/sbin/httpd -DFOREGROUND

Mar 25 19:16:48 servera.lab.example.com systemd[1]: Starting The
 Apache HTTP Server...
Mar 25 19:16:48 servera.lab.example.com httpd[6565]: Server
 configured, listening on: port 80
```

5. Open a web browser on `workstation` and try to view `http://servera/index.html`. You will get an error message that says you do not have permission to access the file.

6. To permit access to the **index.html** file on servera, SELinux must be configured. Define an SELinux file context rule that sets the context type to **httpd_sys_content_t** for the **/custom** directory and all the files below it.

```
[root@servera ~]# semanage fcontext -a -t httpd_sys_content_t '/
custom(/.*)?'
```

7. Use the **restorecon** command to change the file contexts.

```
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
```

8. Try to view http://servera/index.html again. You should see the message **This is SERVERA.** displayed.

9. Exit from servera.

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

Finish

On workstation, run the **lab selinux-filecontexts finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-filecontexts finish
```

This concludes the guided exercise.

ADJUSTING SELINUX POLICY WITH BOOLEANS

OBJECTIVES

After completing this section, you should be able to:

- Activate and deactivate SELinux policy rules using **setsebool**.
- Manage the persistent value of SELinux booleans using the **semanage boolean -l** command.
- Consult man pages that end with **_selinux** to find useful information about SELinux booleans.

SELINUX BOOLEANS

SELinux booleans are switches that change the behavior of the SELinux policy. SELinux booleans are rules that can be enabled or disabled. They can be used by security administrators to tune the policy to make selective adjustments.

The SELinux man pages, provided with the *selinux-policy-doc* package, describe the purpose of the available booleans. The **man -k '_selinux'** command lists these man pages.

Commands useful for managing SELinux booleans include **getsebool**, which lists booleans and their state, and **setsebool** which modifies booleans. **setsebool -P** modifies the SELinux policy to make the modification persistent. And **semanage boolean -l** reports on whether or not a boolean is persistent, along with a short description of the boolean.

Non-privileged users can run the **getsebool** command, but you must be a superuser to run **semanage boolean -l** and **setsebool -P**.

```
[user@host ~]$ getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
antivirus_can_scan_system --> off
antivirus_use_jit --> off
...output omitted...
[user@host ~]$ getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
```

```
[user@host ~]$ setsebool httpd_enable_homedirs on
Could not change active booleans. Please try as root: Permission denied
[user@host ~]$ sudo setsebool httpd_enable_homedirs on
[user@host ~]$ sudo semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on   ,  off)  Allow httpd to enable homedirs
[user@host ~]$ getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

The **-P** option writes all pending values to the policy, making them persistent across reboots. In the example that follows, note the values in parentheses: both are now set to **on**.

```
[user@host ~]$ setsebool -P httpd_enable_homedirs on
[user@host ~]$ sudo semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on   ,   on)  Allow httpd to enable homedirs
```

To list booleans in which the current state differs from the default state, run **semanage boolean -l -C**.

```
[user@host ~]$ sudo semanage boolean -l -C
SELinux boolean                State  Default Description

cron_can_relabel               (off   ,   on)  Allow cron to can relabel
```

REFERENCES

booleans(8), **getsebool**(8), **setsebool**(8), **semanage**(8), **semanage-boolean**(8) man pages

GUIDED EXERCISE

ADJUSTING SELINUX POLICY WITH BOOLEANS

Apache can publish web content hosted in users' home directories, but SELinux prevents this by default. In this exercise, you will identify and change the SELinux boolean that permits Apache to access user home directories.

OUTCOMES

You should be able to configure *Apache* to publish web content from users' home directories.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-booleans start`** command. This command runs a start script that determines whether the `servera` machine is reachable on the network. It also installs the `httpd` service and configures the firewall on `servera` to allow HTTP connections.

```
[student@workstation ~]$ lab selinux-booleans start
```

1. Use the **ssh** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

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

3. To enable the Apache feature that permits users to publish web content from their home directories, you must edit the **`/etc/httpd/conf.d/userdir.conf`** configuration file. Comment out the line that sets **`UserDir`** to **`disabled`** and uncomment the line that sets **`UserDir`** to **`public_html`**.

```
[root@servera ~]# vim /etc/httpd/conf.d/userdir.conf
#UserDir disabled
UserDir public_html
```

4. Use the **grep** command to confirm the changes.

```
[root@servera ~]# grep '#UserDir' /etc/httpd/conf.d/userdir.conf
#UserDir disabled
[root@servera ~]# grep '^ *UserDir' /etc/httpd/conf.d/userdir.conf
UserDir public_html
```

5. Start and enable the Apache web service to make the changes take effect.

```
[root@servera ~]# systemctl enable --now httpd
```

6. In another terminal window log in as `student`. SSH into `servera`. Create some web content that is published from a user's home directory.

   6.1. In another terminal window log in as `student`. Use the **ssh** command to log in to `servera` as the `student` user.

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

   6.2. Use the **mkdir** command to create a directory called **~/public_html**.

```
[student@servera ~]$ mkdir ~/public_html
```

   6.3. Create the **index.html** file with the following content:

```
[student@servera ~]$ echo 'This is student content on SERVERA.' > \
~/public_html/index.html
```

   6.4. Use the **chmod** command to change the permissions on `student`'s home directory so Apache can access the **public_html** subdirectory.

```
[student@servera ~]$ chmod 711 ~
```

7. Open a web browser on `workstation` and try to view the following URL: `http://servera/~student/index.html`. You get an error message that says you do not have permission to access the file.

8. In the terminal window with **root** access, use the **getsebool** command to see if there are any booleans that restrict access to home directories.

```
[root@servera ~]# getsebool -a | grep home
...output omitted...
httpd_enable_homedirs --> off
...output omitted...
```

9. In the terminal window with **root** access, use the **setsebool** command to enable home directory access persistently.

```
[root@servera ~]# setsebool -P httpd_enable_homedirs on
```

10. Try to view `http://servera/~student/index.html` again. You should see the message: **This is student content on SERVERA.**

11. Exit from `servera`.

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

Finish

On `workstation`, run the **lab selinux-booleans finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-booleans finish
```

This concludes the guided exercise.

INVESTIGATING AND RESOLVING SELINUX ISSUES

OBJECTIVES

After completing this section, you should be able to:

- Use SELinux log analysis tools.
- Display useful information during SELinux troubleshooting using the **sealert** command.

TROUBLESHOOTING SELINUX ISSUES

It is important to understand what actions you must take when SELinux prevents access to files on a server that you know should be accessible. Use the following steps as a guide to troubleshooting these issues:

1. Before thinking of making any adjustments, consider that SELinux may be doing its job correctly by prohibiting the attempted access. If a web server tries to access files in **/home**, this could signal a compromise of the service if web content is not published by users. If access should have been granted, then additional steps need to be taken to solve the problem.
2. The most common SELinux issue is an incorrect file context. This can occur when a file is created in a location with one file context and moved into a place where a different context is expected. In most cases, running **restorecon** will correct the issue. Correcting issues in this way has a very narrow impact on the security of the rest of the system.
3. Another remedy for overly restrictive access could be the adjustment of a Boolean. For example, the **ftpd_anon_write** boolean controls whether anonymous FTP users can upload files. You must turn this boolean on to permit anonymous FTP users to upload files to a server. Adjusting booleans requires more care because they can have a broad impact on system security.
4. It is possible that the SELinux policy has a bug that prevents a legitimate access. Since SELinux has matured, this is a rare occurrence. When it is clear that a policy bug has been identified, contact Red Hat support to report the bug so it can be resolved.

MONITORING SELINUX VIOLATIONS

Install the *setroubleshoot-server* package to send SELinux messages to **/var/log/messages**. **setroubleshoot-server** listens for audit messages in **/var/log/audit/audit.log** and sends a short summary to **/var/log/messages**. This summary includes *unique identifiers (UUID)* for SELinux violations that can be used to gather further information. The **sealert -l *UUID*** command is used to produce a report for a specific incident. Use **sealert -a /var/log/audit/audit.log** to produce reports for all incidents in that file.

Consider the following sample sequence of commands on a standard Apache web server:

```
[root@host ~]# touch /root/file3
[root@host ~]# mv /root/file3 /var/www/html
[root@host ~]# systemctl start httpd
[root@host ~]# curl http://localhost/file3
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /file3
on this server.</p>
</body></html>
```

You expect the web server to deliver the contents of **file3** but instead it returns a **permission denied** error. Inspecting both **/var/log/audit/audit.log** and **/var/log/messages** reveals extra information about this error.

```
[root@host ~]# tail /var/log/audit/audit.log
...output omitted...
type=AVC msg=audit(1392944135.482:429): avc:  denied  { getattr } for
  pid=1609 comm="httpd" path="/var/www/html/file3" dev="vda1" ino=8980981
  scontext=system_u:system_r:httpd_t:s0
  tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file
...output omitted...
[root@host ~]# tail /var/log/messages
...output omitted...
Feb 20 19:55:42 host setroubleshoot: SELinux is preventing /usr/sbin/httpd
  from getattr access on the file . For complete SELinux messages. run
  sealert -l 613ca624-248d-48a2-a7d9-d28f5bbe2763
```

Both log files indicate that an SELinux denial is the culprit. The **sealert** command that is part of the output in **/var/log/messages** provides extra information, including a possible fix.

```
[root@host ~]# sealert -l 613ca624-248d-48a2-a7d9-d28f5bbe2763
SELinux is preventing /usr/sbin/httpd from getattr access on the file .

*****  Plugin catchall (100. confidence) suggests   **************************

If you believe that httpd should be allowed getattr access on the
  file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# grep httpd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -i mypol.pp


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:admin_home_t:s0
Target Objects                 [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          servera
Source RPM Packages           httpd-2.4.6-14.el7.x86_64
```

```
Target RPM Packages
Policy RPM                    selinux-policy-3.12.1-124.el7.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Host Name                     servera
Platform                      Linux servera 3.10.0-84.el7.x86_64 #1
                              SMP Tue Feb 4 16:28:19 EST 2014 x86_64 x86_64
Alert Count                   2
First Seen                    2014-02-20 19:55:35 EST
Last Seen                     2014-02-20 19:55:35 EST
Local ID                      613ca624-248d-48a2-a7d9-d28f5bbe2763

Raw Audit Messages
type=AVC msg=audit(1392944135.482:429): avc:  denied  { getattr } for
  pid=1609 comm="httpd" path="/var/www/html/file3" dev="vda1" ino=8980981
  scontext=system_u:system_r:httpd_t:s0
  tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file

type=SYSCALL msg=audit(1392944135.482:429): arch=x86_64 syscall=lstat
  success=no exit=EACCES a0=7f9fed0edea8 a1=7fff7bffc770 a2=7fff7bffc770
  a3=0 items=0 ppid=1608 pid=1609 auid=4294967295 uid=48 gid=48 euid=48
  suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295
  comm=httpd exe=/usr/sbin/httpd subj=system_u:system_r:httpd_t:s0 key=(null)

Hash: httpd,httpd_t,admin_home_t,file,getattr
```

NOTE

The **Raw Audit Messages** section reveals the target file that is the problem, **/var/www/html/file3**. Also, the target context, **tcontext**, does not look like it belongs with a web server. Use the **restorecon /var/www/html/file3** command to fix the file context. If there are other files that need to be adjusted, **restorecon** can recursively reset the context: **restorecon -R /var/www/**.

The **Raw Audit Messages** section of the **sealert** command contains information from **/var/log/audit.log**. To search the **/var/log/audit.log** file use the **ausearch** command. The **-m** searches on the message type. The **-ts** option searches based on time.

```
[root@host ~]# ausearch -m AVC -ts recent
----
time->Tue Apr  9 13:13:07 2019
type=PROCTITLE msg=audit(1554808387.778:4002):
 proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1554808387.778:4002): arch=c000003e syscall=49
 success=no exit=-13 a0=3 a1=55620b8c9280 a2=10 a3=7ffed967661c items=0
 ppid=1 pid=9340 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
 subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1554808387.778:4002): avc:  denied  { name_bind }
 for  pid=9340 comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0
 tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
```

WEB CONSOLE

If *Web Console* is installed it can also be used for troubleshooting SELinux issues. Log on to Web Console and select ***SELinux*** from the menu on the left. The SELinux Policy window informs you of the current enforce policy. Any issues are detailed in the ***SELinux Access Control Errors*** section.

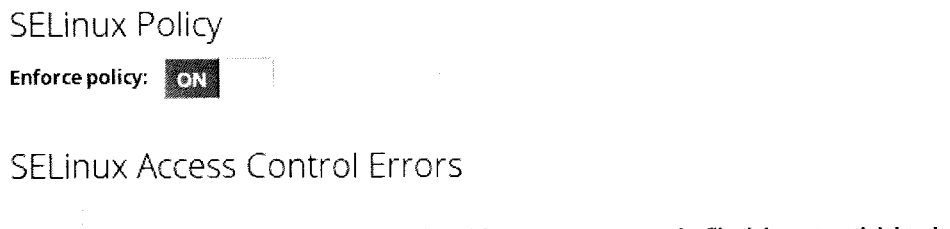


Figure 5.3: SELinux Policy in Web Console

Click on the **>** character to show error details. Click on ***solution details*** to show all details and possible solution.

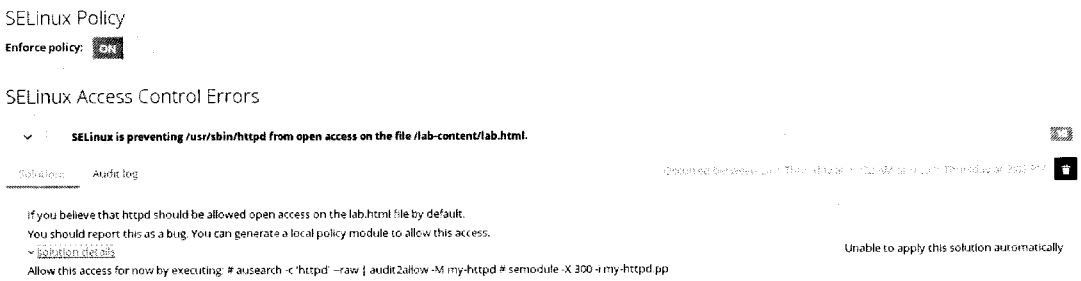


Figure 5.4: SELinux Policy Solution in Web Console

Once the problem has been solved, the ***SELinux Access Control Errors*** section should no longer show the error. If the message **`No SELinux alerts`** appears, then all issues have been fixed.

Figure 5.5: No SELinux Alerts in Web Console

REFERENCES

`sealert`(8) man page

GUIDED EXERCISE

INVESTIGATING AND RESOLVING SELINUX ISSUES

In this lab, you will learn how to troubleshoot SELinux security denials.

OUTCOMES

You will get some experience using SELinux troubleshooting tools.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab selinux-issues start`** command. This command runs a start script that determines whether the `servera` machine is reachable on the network. It installs the `httpd` service, configures the firewall on `servera` to allow HTTP connections, and removes the SELinux context for the **`/custom`** directory.

```
[student@workstation ~]$ lab selinux-issues start
```

1. Use the **`ssh`** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

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

3. Open a web browser on `workstation` and try to view `http://servera/index.html`. You will get an error message that says you do not have permission to access the file.

4. Using the **`less`** command, view the contents of **`/var/log/messages`**. Use the **`/`** key and search for **`sealert`**. Copy the suggested **`sealert`** command so that it can be used in the next step. Use the **`q`** key to quit the **`less`** command.

```
[root@servera ~]# less /var/log/messages
...output omitted...
Mar 28 06:07:03 servera setroubleshoot[15326]: SELinux is preventing /usr/
sbin/httpd from getattr access on the file /custom/index.html. For complete
 SELinux messages run: sealert -l b1c9cc8f-a953-4625-b79b-82c4f4f1fee3
Mar 28 06:07:03 servera platform-python[15326]: SELinux is
 preventing /usr/sbin/httpd from getattr access on the file /custom/
index.html.#012#012*****  Plugin catchall (100. confidence) suggests
 **************************#012#012If you believe that httpd should be
 allowed getattr access on the index.html file by default.#012Then you
 should report this as a bug.#012You can generate a local policy module to
 allow this access.#012Do#012allow this access for now by executing:#012#
 ausearch -c 'httpd' --raw | audit2allow -M my-httpd#012# semodule -X 300 -i
 my-httpd.pp#012
Mar 28 06:07:04 servera setroubleshoot[15326]: failed to retrieve rpm info
 for /custom/index.html
...output omitted...
```

5. Run the suggested **sealert** command. Note the source context, the target objects, the policy, and the enforcing mode.

```
[root@servera ~]# sealert -l b1c9cc8f-a953-4625-b79b-82c4f4f1fee3
SELinux is preventing /usr/sbin/httpd from getattr access on the file /
custom/index.html.

*****  Plugin catchall (100. confidence) suggests
 **************************

If you believe that httpd should be allowed getattr access on the index.html
 file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:default_t:s0
Target Objects                /custom/index.html [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          servera.lab.example.com
Source RPM Packages
Target RPM Packages
Policy RPM                    selinux-policy-3.14.1-59.el8.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
```

```
Host Name                    servera.lab.example.com
Platform                     Linux servera.lab.example.com
 4.18.0-67.el8.x86_64
                             #1 SMP Sat Feb 9 12:44:00 UTC 2019 x86_64
 x86_64
                             Alert Count                  18
                             First Seen                   2019-03-25
 19:25:28 CET
                             Last Seen                    2019-03-28
 11:07:00 CET
                             Local ID                     b1c9cc8f-
a953-4625-b79b-82c4f4f1fee3

                             Raw Audit Messages
                             type=AVC msg=audit(1553767620.970:16958):
 avc:  denied  { getattr } for  pid=15067 comm="httpd" path="/custom/
index.html" dev="vda1" ino=4208311 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0


                             Hash: httpd,httpd_t,default_t,file,getattr
```

6. The **Raw Audit Messages** section of the **sealert** command contains information from the **/var/log/audit/audit.log**. Use the **ausearch** command to search the **/var/log/audit/audit.log** file. The **-m** option searches on the message type. The **-ts** option searches based on time. This entry identifies the relevant process and file causing the alert. The process is the **httpd** Apache web server, the file is **/custom/index.html**, and the context is **system_r:httpd_t**.

```
[root@servera ~]# ausearch -m AVC -ts recent
----
time->Thu Mar 28 13:39:30 2019
type=PROCTITLE msg=audit(1553776770.651:17000):
 proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1553776770.651:17000): arch=c000003e syscall=257
 success=no exit=-13 a0=ffffff9c a1=7f8db803f598 a2=80000 a3=0 items=0
 ppid=15063 pid=15065 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48
 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/
sbin/httpd" subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1553776770.651:17000): avc:  denied
 { open } for  pid=15065 comm="httpd" path="/custom/index.html"
 dev="vda1" ino=4208311 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

7. To resolve the issue use the **semanage** and **restorecon** commands. The context to manage is **httpd_sys_content_t**.

```
[root@servera ~]# semanage fcontext -a -t httpd_sys_content_t '/
custom(/.*)?'
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
 unconfined_u:object_r:httpd_sys_content_t:s0
```

8. Try to view `http://servera/index.html` again. You should see the message **This is SERVERA.** displayed.

9. Exit from `servera`.

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

Finish

On `workstation`, run the **lab selinux-issues finish** script to complete this exercise.

```
[student@workstation ~]$ lab selinux-issues finish
```

This concludes the guided exercise.

▶ LAB

MANAGING SELINUX SECURITY

PERFORMANCE CHECKLIST

In this lab, you will solve an SELinux access denial problem. System administrators are having trouble getting a new web server to deliver content to clients when SELinux is in enforcing mode.

OUTCOMES

You should be able to:

- Identify issues in system log files.
- Adjust the SELinux configuration.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab selinux-review start`** command. This command runs a start script that determines whether the `serverb` machine is reachable on the network. It also installs the httpd *Apache* server, creates a new DocumentRoot for *Apache*, and updates the configuration file.

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

1. Log in to `serverb` as the root user.
2. Launch a web browser on `workstation` and browse to `http://serverb/lab.html`. You will see the error message: **`You do not have permission to access /lab.html on this server.`**
3. Research and identify the SELinux issue that is preventing Apache from serving web content.
4. Display the SELinux context of the new HTTP document root and the original HTTP document root. Resolve the SELinux issue preventing Apache from serving web content.
5. Verify that the SELinux issue has been resolved and Apache is able to serve web content.
6. Exit from `serverb`.

Evaluation

On `workstation`, run the **`lab selinux-review grade`** script to confirm success on this exercise.

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

Finish

On `workstation`, run the **`lab selinux-review finish`** script to complete the lab.

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

This concludes the lab.

SOLUTION

MANAGING SELINUX SECURITY

PERFORMANCE CHECKLIST

In this lab, you will solve an SELinux access denial problem. System administrators are having trouble getting a new web server to deliver content to clients when SELinux is in enforcing mode.

OUTCOMES

You should be able to:

- Identify issues in system log files.
- Adjust the SELinux configuration.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab selinux-review start`** command. This command runs a start script that determines whether the `serverb` machine is reachable on the network. It also installs the httpd *Apache* server, creates a new DocumentRoot for *Apache*, and updates the configuration file.

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

1. Log in to `serverb` as the root user.

   1.1. Use the **ssh** command to log in to `serverb` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

   1.2. Use the **`sudo -i`** command to switch to the `root` user. The password for the `student` user is `student`.

   ```
   [student@serverb ~]$ sudo -i
   [sudo] password for student: student
   [root@serverb ~]#
   ```

2. Launch a web browser on `workstation` and browse to `http://serverb/lab.html`. You will see the error message: **`You do not have permission to access /lab.html on this server.`**

3. Research and identify the SELinux issue that is preventing Apache from serving web content.

3.1. Using the **less** command, view the contents of **/var/log/messages**. Use the **/** key and search for **sealert**. Use the **q** key to quit the **less** command.

```
[root@serverb ~]# less /var/log/messages
Mar 28 10:19:51 serverb setroubleshoot[27387]: SELinux is
 preventing /usr/sbin/httpd from getattr access on the file /lab-
content/lab.html. For complete SELinux messages run: sealert -l
 8824e73d-3ab0-4caf-8258-86e8792fee2d
Mar 28 10:19:51 serverb platform-python[27387]: SELinux is preventing /
usr/sbin/httpd from getattr access on the file /lab-content/
lab.html.#012#012*****  Plugin catchall (100. confidence) suggests
  **************************#012#012If you believe that httpd should
 be allowed getattr access on the lab.html file by default.#012Then
 you should report this as a bug.#012You can generate a local policy
 module to allow this access.#012Do#012allow this access for now
 by executing:#012# ausearch -c 'httpd' --raw | audit2allow -M my-
httpd#012# semodule -X 300 -i my-httpd.pp#012
```

3.2. Run the suggested **sealert** command. Note the source context, the target objects, the policy, and the enforcing mode.

```
[root@serverb ~]# sealert -l 8824e73d-3ab0-4caf-8258-86e8792fee2d
SELinux is preventing /usr/sbin/httpd from getattr access on the file /
lab-content/lab.html.

*****  Plugin catchall (100. confidence) suggests
 **************************

If you believe that httpd should be allowed getattr access on the
 lab.html file by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp


Additional Information:
Source Context                system_u:system_r:httpd_t:s0
Target Context                unconfined_u:object_r:default_t:s0
Target Objects                /lab-content/lab.html [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          serverb.lab.example.com
Source RPM Packages
Target RPM Packages
Policy RPM                    selinux-policy-3.14.1-59.el8.noarch
Selinux Enabled               True
Policy Type                   targeted
Enforcing Mode                Enforcing
Host Name                     serverb.lab.example.com
```

```
Platform                      Linux serverb.lab.example.com
 4.18.0-67.el8.x86_64
                              #1 SMP Sat Feb 9 12:44:00 UTC 2019 x86_64
 x86_64
                              Alert Count                  2
                              First Seen                   2019-03-28
 15:19:46 CET
                              Last Seen                    2019-03-28
 15:19:46 CET
                              Local ID
 8824e73d-3ab0-4caf-8258-86e8792fee2d

                              Raw Audit Messages
                              type=AVC msg=audit(1553782786.213:864):
 avc:  denied  { getattr } for  pid=15606
 comm="httpd" path="/lab-content/lab.html" dev="vda1"
 ino=8763212 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0


Hash: httpd,httpd_t,default_t,file,getattr
```

3.3. The **Raw Audit Messages** section of the **sealert** command contains information from the **/var/log/audit/audit.log**. Use the **ausearch** command to search the **/var/log/audit/audit.log** file. The **-m** option searches on the message type. The **ts** option searches based on time. This entry identifies the relevant process and file causing the alert. The process is the **httpd** Apache web server, the file is **/lab-content/lab.html**, and the context is **system_r:httpd_t**.

```
[root@serverb ~]# ausearch -m AVC -ts recent
time->Thu Mar 28 15:19:46 2019
type=PROCTITLE msg=audit(1553782786.213:864):
 proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1553782786.213:864): arch=c000003e
 syscall=6 success=no exit=-13 a0=7fb900004930 a1=7fb92dfca8e0
 a2=7fb92dfca8e0 a3=1 items=0 ppid=15491 pid=15606 auid=4294967295
 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48
 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
 subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1553782786.213:864): avc:  denied  { getattr }
 for  pid=15606 comm="httpd" path="/lab-content/lab.html"
 dev="vda1" ino=8763212 scontext=system_u:system_r:httpd_t:s0
 tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

4. Display the SELinux context of the new HTTP document root and the original HTTP document root. Resolve the SELinux issue preventing Apache from serving web content.

   4.1. Use the **ls -dZ** to compare the document root of **/lab-content** and **/var/www/html**.

```
[root@serverb ~]# ls -dZ /lab-content /var/www/html
unconfined_u:object_r:default_t:s0 /lab-content/
 system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
```

   4.2. Create a file context rule that sets the default type to **httpd_sys_content_** for **/lab-content** and all the files below it.

```
[root@serverb ~]# semanage fcontext -a -t httpd_sys_content_t '/lab-content(/.*)?'
```

   4.3. Use the **restorecon** command to set the SELinux context for the files in **/lab-content**.

```
[root@serverb ~]# restorecon -R /lab-content/
```

5. Verify that the SELinux issue has been resolved and Apache is able to serve web content.

   Use your web browser to refresh the `http://serverb/lab.html` link. Now you should see some web content.

```
This is the html file for the SELinux final lab on SERVERB.
```

6. Exit from `serverb`.

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

Evaluation

On `workstation`, run the **lab selinux-review grade** script to confirm success on this exercise.

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

Finish

On `workstation`, run the **lab selinux-review finish** script to complete the lab.

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

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The **`getenforce`** and **`setenforce`** commands are used to manage the SELinux mode of a system.
- The **`semanage`** command is used to manage SELinux policy rules. The **`restorecon`** command applies the context defined by the policy.
- Booleans are switches that change the behavior of the SELinux policy. They can be enabled or disabled and are used to tune the policy.
- The **`sealert`** displays useful information to help with SELinux troubleshooting.

CHAPTER 6

TUNING SYSTEM PERFORMANCE

GOAL	Evaluate and control processes, set tuning parameters and adjust process scheduling priorities on a Red Hat Enterprise Linux system.
OBJECTIVES	• Control and terminate processes that are not associated with your shell, and forcibly end user sessions and processes. • Describe what load average is and determine processes responsible for high resource use on a server. • Optimize system performance by selecting a tuning profile managed by the tuned daemon. • Prioritize or de-prioritize specific processes, with the nice and renice commands.
SECTIONS	• Killing Processes (and Guided Exercise) • Monitoring Process Activity (and Guided Exercise) • Adjusting Tuning Profiles (and Guided Exercise) • Influencing Process Scheduling (and Guided Exercise)
LAB	Tuning System Performance

KILLING PROCESSES

OBJECTIVES

After completing this section, you should be able to:

- Use commands to kill and communicate with processes.
- Define the characteristics of a daemon process.
- End user sessions and processes.

PROCESS CONTROL USING SIGNALS

A signal is a software interrupt delivered to a process. Signals report events to an executing program. Events that generate a signal can be an error, external event (an I/O request or an expired timer), or by explicit use of a signal-sending command or keyboard sequence.

The following table lists the fundamental signals used by system administrators for routine process management. Refer to signals by either their short (HUP) or proper (SIGHUP) name.

Fundamental Process Management Signals

SIGNAL NUMBER	SHORT NAME	DEFINITION	PURPOSE
1	HUP	Hangup	Used to report termination of the controlling process of a terminal. Also used to request process reinitialization (configuration reload) without termination.
2	INT	Keyboard interrupt	Causes program termination. Can be blocked or handled. Sent by pressing INTR key sequence (**`Ctrl`**+**`c`**).
3	QUIT	Keyboard quit	Similar to SIGINT; adds a process dump at termination. Sent by pressing QUIT key sequence (**`Ctrl`**+\).
9	KILL	Kill, unblockable	Causes abrupt program termination. Cannot be blocked, ignored, or handled; always fatal.
15 *default*	TERM	Terminate	Causes program termination. Unlike SIGKILL, can be blocked, ignored, or handled. The "polite" way to ask a program to terminate; allows self-cleanup.
18	CONT	Continue	Sent to a process to resume, if stopped. Cannot be blocked. Even if handled, always resumes the process.
19	STOP	Stop, unblockable	Suspends the process. Cannot be blocked or handled.
20	TSTP	Keyboard stop	Unlike SIGSTOP, can be blocked, ignored, or handled. Sent by pressing SUSP key sequence (**`Ctrl`**+**`z`**).

NOTE
Signal numbers vary on different Linux hardware platforms, but signal names and meanings are standardized. For command use, it is advised to use signal names instead of numbers. The numbers discussed in this section are for x86_64 systems.

Each signal has a *default action*, usually one of the following:

- `Term` - Cause a program to terminate (exit) at once.
- `Core` - Cause a program to save a memory image (core dump), then terminate.
- `Stop` - Cause a program to stop executing (suspend) and wait to continue (resume).

Programs can be prepared to react to expected event signals by implementing handler routines to ignore, replace, or extend a signal's default action.

Commands for Sending Signals by Explicit Request

You signal their current foreground process by pressing a keyboard control sequence to suspend (**`Ctrl`**+**`z`**), kill (**`Ctrl`**+**`c`**), or core dump (**`Ctrl`**+**`\`**) the process. However, you will use signal-sending commands to send signals to a background process or to processes in a different session.

Signals can be specified as options either by name (for example, **`-HUP`** or **`-SIGHUP`**) or by number (the related **`-1`**). Users may kill their own processes, but root privilege is required to kill processes owned by others.

The **`kill`** command sends a signal to a process by PID number. Despite its name, the kill command can be used for sending any signal, not just those for terminating programs. You can use the **`kill -l`** command to list the names and numbers of all available signals.

```
[user@host ~]$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
...output omitted...
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1    S+   16:41   0:00 grep --color=auto job
[user@host ~]$ kill 5194
[user@host ~]$ ps aux | grep job
user   5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job2
user   5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job3
user   5783  0.0  0.0 221860   964 pts/1    S+   16:43   0:00 grep --color=auto
 job
[1]   Terminated              control job1
[user@host ~]$ kill -9 5199
```

```
[user@host ~]$ ps aux | grep job
user   5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/
user/bin/control job3
user   5930  0.0  0.0 221860  1048 pts/1    S+   16:44   0:00 grep --color=auto
 job
[2]-  Killed                  control job2
[user@host ~]$ kill -SIGTERM 5205
user   5986  0.0  0.0 221860  1048 pts/1    S+   16:45   0:00 grep --color=auto
 job
[3]+  Terminated              control job3
```

The **killall** command can signal multiple processes, based on their command name.

```
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1    S    16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1    S+   16:41   0:00 grep --color=auto job
[user@host ~]$ killall control
[1]   Terminated              control job1
[2]-  Terminated              control job2
[3]+  Terminated              control job3
```

Use **pkill** to send a signal to one or more processes which match selection criteria. Selection criteria can be a command name, a processes owned by a specific user, or all system-wide processes. The **pkill** command includes advanced selection criteria:

- Command - Processes with a pattern-matched command name.
- UID - Processes owned by a Linux user account, effective or real.
- GID - Processes owned by a Linux group account, effective or real.
- Parent - Child processes of a specific parent process.
- Terminal - Processes running on a specific controlling terminal.

```
[user@host ~]$ ps aux | grep pkill
user   5992  0.0  0.1 222448  3040 pts/1    S    16:59   0:00 /bin/bash /home/
user/bin/control pkill1
user   5996  0.0  0.1 222448  3048 pts/1    S    16:59   0:00 /bin/bash /home/
user/bin/control pkill2
user   6004  0.0  0.1 222448  3048 pts/1    S    16:59   0:00 /bin/bash /home/
user/bin/control pkill3
[user@host ~]$ pkill control
[1]   Terminated              control pkill1
[2]-  Terminated              control pkill2
[user@host ~]$ ps aux | grep pkill
user   6219  0.0  0.0 221860  1052 pts/1    S+   17:00   0:00 grep --color=auto
 pkill
[3]+  Terminated              control pkill3
[user@host ~]$ ps aux | grep test
user   6281  0.0  0.1 222448  3012 pts/1    S    17:04   0:00 /bin/bash /home/
user/bin/control test1
```

```
user   6285  0.0  0.1 222448  3128 pts/1    S    17:04   0:00 /bin/bash /home/
user/bin/control test2
user   6292  0.0  0.1 222448  3064 pts/1    S    17:04   0:00 /bin/bash /home/
user/bin/control test3
user   6318  0.0  0.0 221860  1080 pts/1    S+   17:04   0:00 grep --color=auto
 test
[user@host ~]$ pkill -U user
[user@host ~]$ ps aux | grep test
user   6870  0.0  0.0 221860  1048 pts/0    S+   17:07   0:00 grep --color=auto
 test
```

LOGGING USERS OUT ADMINISTRATIVELY

You may need to log other users off for any of a variety of reasons. To name a few of the many possibilities: the user committed a security violation; the user may have overused resources; the user may have an unresponsive system; or the user has improper access to materials. In these cases, you may need to administratively terminate their session using signals.

To log off a user, first identify the login session to be terminated. Use the **w** command to list user logins and current running processes. Note the **TTY** and **FROM** columns to determine the sessions to close.

All user login sessions are associated with a terminal device (TTY). If the device name is of the form **pts/*N***, it is a *pseudo-terminal* associated with a graphical terminal window or remote login session. If it is of the form **tty*N***, the user is on a system console, alternate console, or other directly connected terminal device.

```
[user@host ~]$ w
 12:43:06 up 27 min,  5 users,  load average: 0.03, 0.17, 0.66
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty2                      12:26   14:58   0.04s  0.04s -bash
bob      tty3                      12:28   14:42   0.02s  0.02s -bash
user     pts/1    desk.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. For each session, CPU resources consumed by current jobs, including background tasks and child processes, are in the **JCPU** column. Current foreground process CPU consumption is in the **PCPU** column.

Processes and sessions can be individually or collectively signaled. To terminate all processes for one user, use the **pkill** command. Because the initial process in a login session (*session leader*) is designed to handle session termination requests and ignore unintended keyboard signals, killing all of a user's processes and login shells requires using the SIGKILL signal.

IMPORTANT

SIGKILL is commonly used too quickly by administrators.

Since the SIGKILL signal cannot be handled or ignored, it is always fatal. However, it forces termination without allowing the killed process to run self-cleanup routines. It is recommended to send SIGTERM first, then try SIGINT, and only if both fail retry with SIGKILL.

First identify the PID numbers to be killed using **pgrep**, which operates much like **pkill**, including using the same options, except that **pgrep** lists processes rather than killing them.

```
[root@host ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@host ~]# pkill -SIGKILL -u bob
[root@host ~]# pgrep -l -u bob
[root@host ~]#
```

When processes requiring attention are in the same login session, it may not be necessary to kill all of a user's processes. Determine the controlling terminal for the session using the **w** command, then kill only processes referencing the same terminal ID. Unless **SIGKILL** is specified, the session leader (here, the ***Bash*** login shell) successfully handles and survives the termination request, but all other session processes are terminated.

```
[root@host ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@host ~]# w -h -u bob
bob      tty3      18:37    5:04   0.03s  0.03s -bash
[root@host ~]# pkill -t tty3
[root@host ~]# pgrep -l -u bob
7391 bash
[root@host ~]# pkill -SIGKILL -t tty3
[root@host ~]# pgrep -l -u bob
```

The same selective process termination can be applied using 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 they have created. This time, the parent ***Bash*** login shell survives because the signal is directed only at its child processes.

```
[root@host ~]# pstree -p bob
bash(8391)─┬─sleep(8425)
           ├─sleep(8426)
           └─sleep(8427)
[root@host ~]# pkill -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
[root@host ~]# pkill -SIGKILL -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
```

REFERENCES

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

- Section 24: Signal Handling

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

- Section 26: Processes

`kill`(1), `killall`(1), `pgrep`(1), `pkill`(1), `pstree`(1), `signal`(7), and `w`(1) man pages

GUIDED EXERCISE

KILLING PROCESSES

In this exercise, you will use signals to manage and stop processes.

OUTCOMES

You should be able to start and stop multiple shell processes.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab processes-kill start`** command. The command runs a start script that determines whether the host, `servera`, is reachable on the network.

```
[student@workstation ~]$ lab processes-kill start
```

1. On `workstation`, 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 `servera` as the `student` user.

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

2. In the *left* window, create a new directory called **`/home/student/bin`**. In the new directory, create a shell script called **`killing`**. Make the script executable.

   2.1. Use the **`mkdir`** command to create a new directory called **`/home/student/bin`**.

```
[student@servera ~]$ mkdir /home/student/bin
```

   2.2. Use the **`vim`** command to create a script called **`killing`** in the **`/home/student/bin`** directory. Press the **`i`** key to enter ***Vim*** interactive mode. Use the **`:wq`** command to save the file.

```
[student@servera ~]$ vim /home/student/bin/killing
#!/bin/bash
while true; do
  echo -n "$@ " >> ~/killing_outfile
  sleep 5
done
```

NOTE
The **killing** script runs until terminated. It appends command line arguments to the **~/killing_outfile** once every 5 seconds.

2.3. Use the **chmod** command to make the **killing** file executable.

```
[student@servera ~]$ chmod +x /home/student/bin/killing
```

3. In the left terminal shell, use the **cd** command to change into the **/home/student/bin/** directory. Start three **killing** processes with the arguments **network**, **interface**, and **connection**, respectively. Start three processes called network, interface, and connection. Use the ampersand (**&**) to start the processes in the background.

```
[student@servera ~]$ cd /home/student/bin
[student@servera bin]$ killing network &
[1] 3460
[student@servera bin]$ killing interface &
[2] 3482
[student@servera bin]$ killing connection &
[3] 3516
```

Your processes will have different PID numbers.

4. In the right terminal shell, use the **tail** command with the **-f** option to confirm that all three processes are appending to the **/home/student/killing_outfile** file.

```
[student@servera ~]$ tail -f ~/killing_outfile
network interface network connection interface network connection interface
 network
...output omitted...
```

5. In the left terminal shell, use the **jobs** command to list jobs.

```
[student@servera bin]$ jobs
[1]   Running                 killing network &
[2]-  Running                 killing interface &
[3]+  Running                 killing connection &
```

6. Use signals to suspend the `network` process. Confirm that the `network` process is **`Stopped`**. In the right terminal shell, confirm that the `network` process is no longer appending output to the **`~/killing_output`**.

6.1. Use the **`kill`** with the **`-SIGSTOP`** option to stop the `network` process. Run the **`jobs`** to confirm it is stopped.

```
[student@servera bin]$ kill -SIGSTOP %1
[1]+  Stopped                 killing network
[student@servera bin]$ jobs
[1]+  Stopped                 killing network
[2]   Running                 killing interface &
[3]-  Running                 killing connection &
```

6.2. In the right terminal shell, look at the output from the **`tail`** command. Confirm that the word **`network`** is no longer being appended to the **`~/killing_outfile`** file.

```
...output omitted...
interface connection interface connection interface connection
 interface
```

7. In the left terminal shell, terminate the `interface` process using signals. Confirm that the `interface` process has disappeared. In the right terminal shell, confirm that `interface` process output is no longer appended to the **`~/killing_outfile`** file.

7.1. Use the **`kill`** command with the **`-SIGSTERM`** option to terminate the `interface` process. Run the **`jobs`** command to confirm that it has been terminated.

```
[student@servera bin]$ kill -SIGTERM %2
[student@servera bin]$ jobs
[1]+  Stopped                 killing network
[2]   Terminated              killing interface
[3]-  Running                 killing connection &
```

7.2. In the right terminal shell, look at the output from the **`tail`** command. Confirm that the word **`interface`** is no longer being appended to the **`~/killing_outfile`** file.

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

8. In the left terminal shell, resume the `network` process using signals. Confirm that the `network` process is **Running**. In the right window, confirm that `network` process output is being appended to the **~/killing_outfile** file.

8.1. Use the **kill** command with the **-SIGCONT** to resume the `network` process. Run the **jobs** command to confirm that the process is **Running**.

```
[student@servera bin]$ kill -SIGCONT %1
[student@servera bin]$ jobs
[1]+  Running                 killing network &
[3]-  Running                 killing connection &
```

8.2. In the right terminal shell, look at the output from the **tail** command. Confirm that the word **network** is being appended to the **~/killing_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. Confirm that no jobs remain and that output has stopped.

9.1. Use the **kill** command with the **-SIGTERM** option to terminate the `network` process. Use the same command to terminate the `connection` process.

```
[student@servera bin]$ kill -SIGTERM %1
[student@servera bin]$ kill -SIGTERM %3
[1]+  Terminated              killing network
[student@servera bin]$ jobs
[3]+  Terminated              killing connection
```

10. In the left terminal shell, list `tail` processes running in all open terminal shells. Terminate running tail commands. Confirm that the process is no longer running.

10.1. Use the **ps** command with the **-ef** option to list all running tail processes. Refine the search using the **grep** command.

```
[student@servera bin]$ ps -ef | grep tail
student   4581 31358  0 10:02 pts/0    00:00:00 tail -f
 killing_outfile
student   4869  2252  0 10:33 pts/1    00:00:00 grep --color=auto tail
```

10.2. Use the **pkill** command with the **-SIGTERM** option to kill the `tail` process. Use the **ps** to confirm it is no longer present.

```
[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, confirm that the **tail** command is no longer running.

```
...output omitted...
network connection network connection network connection Terminated
[student@servera ~]$
```

11. Exit from both terminal windows. Failure to exit from all sessions causes the finish script to fail.

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

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

Finish

On `workstation`, run the **lab processes-kill finish** script to complete this exercise.

```
[student@workstation ~]$ lab processes-kill finish
```

This concludes the guided exercise.

MONITORING PROCESS ACTIVITY

OBJECTIVES

After completing this section, you should be able to describe what load average is and determine processes responsible for high resource use on a server.

DESCRIBING LOAD AVERAGE

Load average is a measurement provided by the Linux kernel that is a simple way to represent the perceived system load over time. It can be used as a rough gauge of how many system resource requests are pending, and to determine whether system load is increasing or decreasing over time.

Every five seconds, the kernel collects the current *load number*, 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.

Understanding the Linux Load Average Calculation

The load average represents the perceived system load over a time period. Linux determines this 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 essentially based on the number of processes that are ready to run (in process state **R**) and are waiting for I/O to complete (in process state **D**).
- Some UNIX systems only consider CPU utilization or run queue length to indicate system load. Linux also includes disk or network utilization because that can have as significant an impact on system performance as CPU load. When experiencing 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 can 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 if the CPU is idle. Either way, system load is impacted and the system appears to run more slowly because processes are waiting to run.

Interpreting Displayed 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. A quick glance indicates whether system load appears to be increasing or decreasing.

If the main contribution to load average is from processes waiting for the CPU, you can calculate the approximate *per CPU* load value to determine whether the system is experiencing significant waiting.

The **lscpu** command can help you determine how many CPUs a system has.

In the following example, the system is a dual-core single socket system with two hyperthreads per core. Roughly speaking, Linux will treat this 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
```

For a moment, imagine that the only contribution to the load number is from processes that need CPU time. Then you can divide the displayed load average values by the number of logical CPUs in the system. A value below 1 indicates satisfactory resource utilization and minimal wait times. A value above 1 indicates resource saturation and some amount of 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 waiting for a CPU adds a count of 1 to the load number. If one process is running on a CPU, the load number is one, the resource (the CPU) is in use, but there are no requests waiting. If that process is running for a full minute, its contribution to the one-minute load average will be 1.

However, processes uninterruptibly sleeping for critical I/O due to a busy disk or network resource are also included in the count and increase the load average. While not an indication of CPU utilization, these processes are added to the queue count because they are waiting for resources and cannot run on a CPU until they get them. This is still system load due to resource limitations that is causing processes not to run.

Until resource saturation, a load average remains below 1, since tasks are seldom found waiting in queue. Load average only increases when resource saturation causes requests to remain queued and are counted by the load calculation routine. When resource utilization approaches 100%, each additional request starts experiencing service wait time.

A number of additional tools report load average, including **w** and **top**.

REAL-TIME PROCESS MONITORING

The **top** program is a dynamic view of the system's processes, displaying a summary header followed by a process or thread list similar to **ps** information. Unlike the static **ps** output, **top** continuously refreshes at a configurable interval, and provides capabilities for column reordering, sorting, and highlighting. User configurations can be saved and made persistent.

Default output columns are recognizable from other resource tools:

- The process ID (**PID**).
- User name (**USER**) is the process owner.
- Virtual memory (**VIRT**) is all memory the process is using, 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 used by the process, including any resident shared objects. (Labeled **RSS** in the **ps** command.)
- Process state (**S**) displays as:
  - **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. May be toggled to include cumulative time of all previous children.
- The process command name (**COMMAND**).

Fundamental Keystrokes in top

KEY	PURPOSE
? *or* **h**	Help for interactive keystrokes.
l, t, m	Toggles for load, threads, and memory header lines.
1	Toggle showing individual CPUs or a summary for all CPUs in header.
s *(1)*	Change the refresh (screen) rate, in decimal seconds (e.g., 0.5, 1, 5).
b	Toggle reverse highlighting for **Running** processes; default is bold only.
Shift+b	Enables use of bold in display, in the header, and for *Running* processes.
Shift+h	Toggle threads; show process summary or individual threads.
u, Shift+u	Filter for any user name (effective, real).
Shift+m	Sorts process listing by memory usage, in descending order.
Shift+p	Sorts process listing by processor utilization, in descending order.

KEY	PURPOSE
k (1)	Kill a process. When prompted, enter **PID**, then **signal**.
r (1)	Renice a process. When prompted, enter **PID**, then **nice_value**.
Shift+w	Write (save) the current display configuration for use at the next **top** restart.
q	Quit.
f	Manage the columns by enabling or disabling fields. Also allows you to set the sort field for **top**.
Note:	(1) Not available if top started in secure mode. See **top**(1).

REFERENCES

ps(1), top(1), uptime(1), and w(1) man pages

GUIDED EXERCISE

MONITORING PROCESS ACTIVITY

In this exercise, you will use the **top** command to dynamically examine running processes and control them.

OUTCOMES

You should be able to manage processes in real time.

BEFORE YOU BEGIN

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

On workstation, run the **lab processes-monitor start** command. The command runs a start script that determines if the host, servera, is reachable on the network.

```
[student@workstation ~]$ lab processes-monitor start
```

1. On workstation open two terminal windows side by side. These terminals are referred to as *left* and *right*. On each terminal, use the **ssh** command to log in to servera as the student user.

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

2. In the *left* terminal shell, create a new directory called **/home/student/bin**. In the new directory create a shell script called **monitor**. Ensure the script is executable.

   2.1. Use the **mkdir** command to create a new directory called **/home/student/bin**.

```
[student@servera ~]$ mkdir /home/student/bin
```

   2.2. Use the **vim** command to create a script called **monitor** in the **/home/student/bin** directory. Press the **i** key to enter ***Vim*** interactive mode. Use the **:wq** command to save the file.

```
[student@servera ~]$ vim /home/student/bin/monitor
#!/bin/bash
while true; do
  var=1
  while [[ var -lt 50000 ]]; do
    var=$(($var+1))
  done
  sleep 1
done
```

NOTE

The **monitor** script runs until terminated. It generates artificial CPU load by performing fifty thousand addition problems. It then sleeps for one second, resets the variable, and repeats.

2.3. Use the **chmod** command to make the **monitor** file executable.

```
[student@servera ~]$ chmod a+x /home/student/bin/monitor
```

3. In the right terminal shell, run the **top** utility. Size the window to be as tall as possible.

```
[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
```

4. In the left terminal shell use the **lscpu** command to determine 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** executable. Use the ampersand (&) to run the process in the background.

```
[student@servera ~]$ monitor &
[1] 6071
```

6. In the right terminal shell, observe the **top** display. Use the single keystrokes **l**, **t**, and **m** to toggle the load, threads, and memory header lines. After observing this behavior, ensure that all headers are displaying.

7. Note the process ID (PID) for **monitor**. View the CPU percentage for the process, which is expected to hover around 15% to 20%.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
071 student   20   0  222448   2964   2716 S  18.7   0.2   0:27.35 monitor
...output omitted...
```

View the load averages. The one minute load average is currently less than a value of 1. The value observed may be affected by resource contention from another virtual machine or 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 **monitor**. Use the ampersand (&) to run the process 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, also expected to hover between 15% and 20%.

```
[student@servera ~]$ top
 PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
6071 student   20   0  222448   2964   2716 S  19.0   0.2   1:36.53 monitor
6498 student   20   0  222448   2996   2748 R  15.7   0.2   0:16.34 monitor
...output omitted...
```

View the one minute load average again, which is still less than 1. It is important to wait for at least one minute to allow 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 **monitor**. Use the ampersand (&) to run the process 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, again expected to hover between 15% and 20%.

```
[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 2 CPUs so only 3 processes are not enough to stress it. Start three more `monitor` processes. View the one minute load average again, which now is expected to be above 1. It is important to wait for at least one minute to allow 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 finished observing the load average values, terminate each of the `monitor` processes from within **top**.

12.1. In the right terminal shell, press **k**. Observe the prompt below the headers and above the columns.

```
...output omitted...
PID to signal/kill [default pid = 11338]
```

12.2. The prompt has chosen 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.

Confirm that the selected process is no longer observed in **top**. If the PID still remains, repeat these terminating steps, substituting 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** instance. Confirm that no monitor processes remain in **top**.

14. In the right terminal shell, press **q** to exit **top**. Exit from servera on both terminal windows.

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

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

Finish

On workstation, run the **lab processes-monitor finish** script to complete this exercise.

```
[student@workstation ~]$ lab processes-monitor finish
```

This concludes the guided exercise.

ADJUSTING TUNING PROFILES

OBJECTIVES

After completing this section, you should be able to optimize system performance by selecting a tuning profile managed by the `tuned` daemon.

TUNING SYSTEMS

System administrators can optimize the performance of a system by adjusting various device settings based on a variety of use case workloads. The `tuned` daemon applies tuning adjustments both statically and dynamically, using tuning profiles that reflect particular workload requirements.

Configuring Static Tuning

The `tuned` daemon applies system settings when the service starts or upon selection of a new tuning profile. Static tuning configures predefined `kernel` parameters in profiles that **`tuned`** applies at runtime. With static tuning, kernel parameters are set for overall performance expectations, and are not adjusted as activity levels change.

Configuring Dynamic Tuning

With dynamic tuning, the `tuned` daemon monitors system activity and adjusts settings depending on runtime behavior changes. Dynamic tuning is continuously adjusting tuning to fit the current workload, starting with the initial settings declared in the chosen tuning profile.

For example, storage devices experience high use during startup and login, but have minimal activity when user workloads consist of using web browsers and email clients. Similarly, CPU and network devices experience activity increases during peak usage throughout a workday. The `tuned` daemon monitors the activity of these components and adjusts parameter settings to maximize performance during high-activity times and reduce settings during low activity. The `tuned` daemon uses performance parameters provided in predefined tuning profiles.

INSTALLING AND ENABLING TUNED

A minimal Red Hat Enterprise Linux 8 installation includes and enables the *tuned* package by default. To install and enable the package manually:

```
[root@host ~]$ yum install tuned
[root@host ~]$ systemctl enable --now tuned
Created symlink /etc/systemd/system/multi-user.target.wants/tuned.service → /usr/
lib/systemd/system/tuned.service.
```

SELECTING A TUNING PROFILE

The ***Tuned*** application provides profiles divided into the following categories:

- Power-saving profiles
- Performance-boosting profiles

The performance-boosting profiles include profiles that focus on the following aspects:

- Low latency for storage and network
- High throughput for storage and network
- Virtual machine performance
- Virtualization host performance

Tuning Profiles Distributed with Red Hat Enterprise Linux 8

TUNED PROFILE	PURPOSE
`balanced`	Ideal for systems that require a compromise between power saving and performance.
`desktop`	Derived from the `balanced` profile. Provides faster response of interactive applications.
`throughput-performance`	Tunes the system for maximum throughput.
`latency-performance`	Ideal for server systems that require low latency at the expense of power consumption.
`network-latency`	Derived from the `latency-performance` profile. It enables additional network tuning parameters to provide low network latency.
`network-throughput`	Derived from the `throughput-performance` profile. Additional network tuning parameters are applied for maximum network throughput.
`powersave`	Tunes the system for maximum power saving.
`oracle`	Optimized for Oracle database loads based on the `throughput-performance` profile.
`virtual-guest`	Tunes the system for maximum performance if it runs on a virtual machine.
`virtual-host`	Tunes the system for maximum performance if it acts as a host for virtual machines.

MANAGING PROFILES FROM THE COMMAND LINE

The **`tuned-adm`** command is used to change settings of the `tuned` daemon. The **`tuned-adm`** command can query current settings, list available profiles, recommend a tuning profile for the system, change profiles directly, or turn off tuning.

A system administrator identifies the currently active tuning profile with **`tuned-adm active`**.

```
[root@host ~]# tuned-adm active
Current active profile: virtual-guest
```

The **`tuned-adm list`** command lists all available tuning profiles, including both built-in profiles and custom tuning profiles created by a system administrator.

```
[root@host ~]# tuned-adm list
Available profiles:
- balanced
- desktop
- latency-performance
- network-latency
- network-throughput
- powersave
- sap
- throughput-performance
- virtual-guest
- virtual-host
Current active profile: virtual-guest
```

Use **`tuned-adm profile`** ***`profilename`*** to switch the active profile to a different one that better matches the system's current tuning requirements.

```
[root@host ~]$ tuned-adm profile throughput-performance
[root@host ~]$ tuned-adm active
Current active profile: throughput-performance
```

The **`tuned-adm`** command can recommend a tuning profile for the system. This mechanism is used to determine the default profile of a system after installation.

```
[root@host ~]$ tuned-adm recommend
virtual-guest
```

NOTE

The **`tuned-adm recommend`** output is based on various system characteristics, including whether the system is a virtual machine and other predefined categories selected during system installation.

To revert the setting changes made by the current profile, either switch to another profile or deactivate the tuned daemon. Turn off `tuned` tuning activity with **`tuned-adm off`**.

```
[root@host ~]$ tuned-adm off
[root@host ~]$ tuned-adm active
No current active profile.
```

MANAGING PROFILES WITH WEB CONSOLE

To manage system performance profiles with Web Console, log in with privileged access. Click the ***Reuse my password for privileged tasks*** option. This permits the user to execute commands, with sudo privileges, that modify system performance profiles.

Figure 6.1: Web Console privileged login

As a privileged user, click the ***Systems*** menu option in the left navigation bar. The current active profile is displayed in the ***Performance Profile*** field. To select a different profile, click the active profile link.

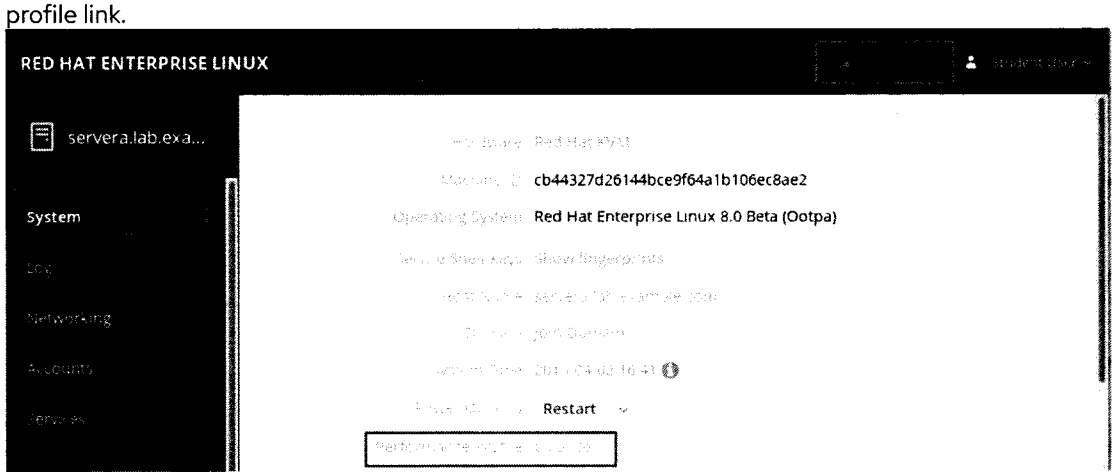


Figure 6.2: Active performance profile

In the ***Change Performance Profile*** user interface, scroll through the profile list to select one that best suits the system purpose.

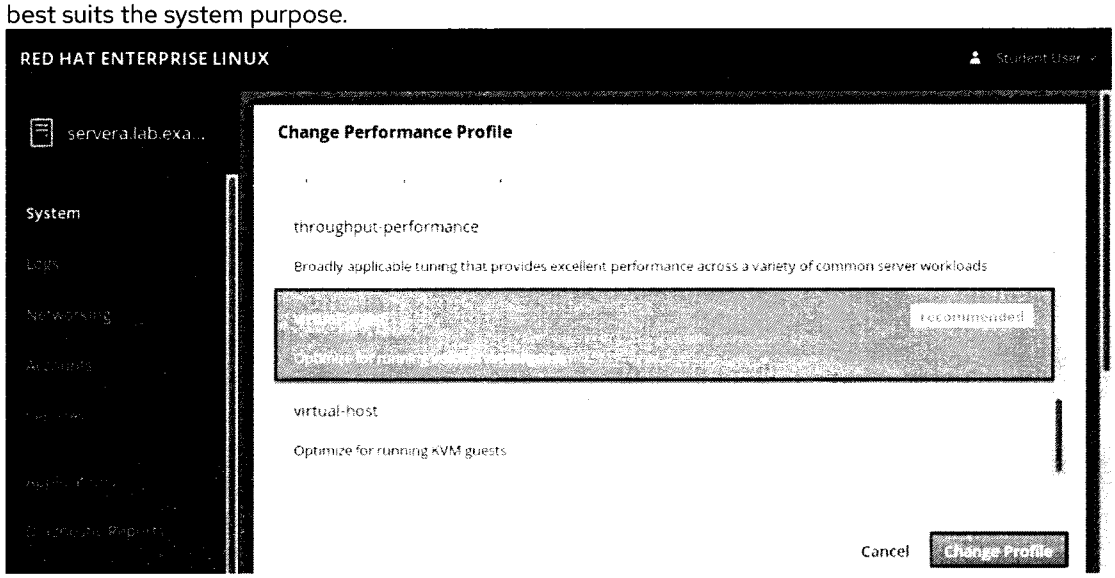


Figure 6.3: Select a preferred performance profile

To verify changes, return to the main ***System*** page and confirm that it displays the active profile in the ***Performance Profile*** field.

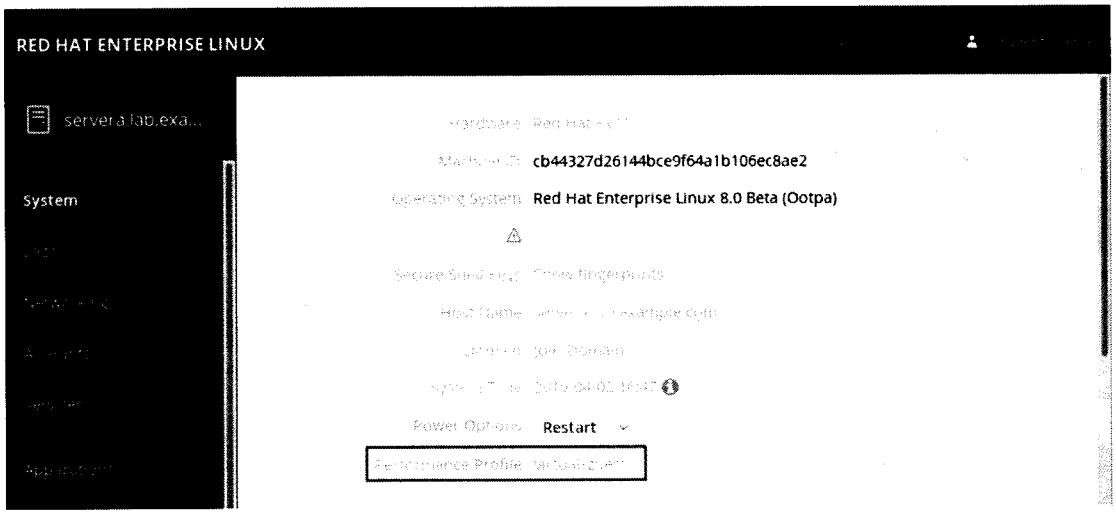


Figure 6.4: Verify active performance profile

REFERENCES

tuned(8), **tuned.conf**(5), **tuned-main.conf**(5) and, **tuned-adm**(1) man pages

GUIDED EXERCISE

ADJUSTING TUNING PROFILES

In this exercise, you will tune a server's performance by activating the `tuned` service and applying a tuning profile.

OUTCOMES

You should be able to configure a system to use a tuning profile.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab tuning-profiles start`** command. The command runs a start script to determine if the `servera` host is reachable on the network.

```
[student@workstation ~]$ lab tuning-profiles start
```

1. From `workstation`, use SSH to log in to `servera` 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 ~]$
```

2. Verify that the *tuned* package is installed, enabled, and started.

   2.1. Use **yum** to confirm that the *tuned* package is installed.

```
[student@servera ~]$ yum list tuned
...output omitted...
Installed Packages
tuned.noarch                    2.10.0-15.el8
 @anaconda
```

   2.2. The **`systemctl is-enabled tuned; systemctl is-active tuned`** command displays its enablement and run state.

```
[student@servera ~]$ systemctl is-enabled tuned; systemctl is-active
 tuned
enabled
active
```

3. List the available tuning profiles and identify the active profile. If `sudo` prompts for a password, enter `student` after the prompt.

```
[student@servera ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- latency-performance         - Optimize for deterministic performance at the
 cost of
                                increased power consumption
- network-latency             - Optimize for deterministic performance at the
 cost of
                                increased power consumption, focused on low
 latency
                                network performance
- network-throughput          - Optimize for streaming network throughput,
 generally
                                only necessary on older CPUs or 40G+ networks
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides excellent
                                performance across a variety of common server
 workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
Current active profile: virtual-guest
```

4. Change the current active tuning profile to `powersave`, then confirm the results. If `sudo` prompts for a password, enter `student` after the prompt.

   4.1. Change the current active tuning profile.

```
[student@servera ~]$ sudo tuned-adm profile powersave
```

   4.2. Confirm that `powersave` is the active tuning profile.

```
[student@servera ~]$ sudo tuned-adm active
Current active profile: powersave
```

5. Exit from `servera`.

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

Finish

On `workstation`, run the **`lab tuning-profiles finish`** script to finish this exercise.

```
[student@workstation ~]$ lab tuning-profiles finish
```

This concludes the guided exercise.

INFLUENCING PROCESS SCHEDULING

OBJECTIVES

After completing this section, you should be able to prioritize or de-prioritize specific processes, with the **`nice`** and **`renice`** commands.

LINUX PROCESS SCHEDULING AND MULTITASKING

Modern computer systems range from low-end systems that have single CPUs that can only execute a single instruction at any instance of time, to high-performing supercomputers with hundreds of CPUs each and dozens or even hundreds of processing cores on each CPU, allowing the execution of huge numbers of instructions in parallel. All these systems still have one thing in common: the need to run more process threads than they have CPUs.

Linux and other operating systems run more processes than there are processing units using a technique called *time-slicing* or *multitasking*. The operating system *process scheduler* rapidly switches between processes on a single core, giving the impression that there are multiple processes running at the same time.

RELATIVE PRIORITIES

Different processes have different levels of importance. The process scheduler can be configured to use different scheduling policies for different processes. The scheduling policy used for most processes running on a regular system is called `SCHED_OTHER` (also called `SCHED_NORMAL`), but other policies exist for various workload needs.

Since not all processes are equally important, processes running with the `SCHED_NORMAL` policy can be given a relative priority. This priority is called the *nice value* of a process, which are organized as **`40`** different levels of niceness for any process.

The nice level values range from -20 (highest priority) to 19 (lowest priority). By default, processes inherit their nice level from their parent, which is usually 0. Higher nice levels indicate less priority (the process easily gives up its CPU usage), while lower nice levels indicate a higher priority (the process is less inclined to give up the CPU). If there is no contention for resources, for example, when there are fewer active processes than available CPU cores, even processes with a high nice level will still use all available CPU resources they can. However, when there are more processes requesting CPU time than available cores, the processes with a higher nice level will receive less CPU time than those with a lower nice level.

SETTING NICE LEVELS AND PERMISSIONS

Since setting a low nice level on a CPU-hungry process might negatively impact the performance of other processes running on the same system, only the `root` user may *reduce* a process nice level.

Unprivileged users are only permitted to *increase* nice levels on their own processes. They cannot lower the nice levels on their processes, nor can they modify the nice level of other users' processes.

REPORTING NICE LEVELS

Several tools display the nice levels of running processes. Process management tools, such as ***top***, display the nice level by default. Other tools, such as the **ps** command, display nice levels when using the proper options.

Displaying Nice Levels with Top

Use the **top** command to interactively view and manage processes. The default configuration displays two columns of interest about nice levels and priorities. The **NI** column displays the process nice value and the **PR** column displays its scheduled priority. In the **top** interface, the nice level maps to an internal system priority queue as displayed in the following graphic. For example, a nice level of -20 maps to 0 in the **PR** column. A nice level of 19 maps to a priority of 39 in the **PR** column.

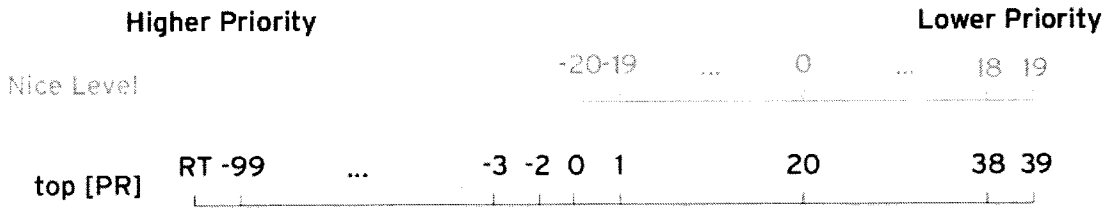


Figure 6.5: Nice levels as reported by top

Displaying Nice Levels from the Command Line

The **ps** command displays process nice levels, but only by including the correct formatting options.

The following **ps** command lists all processes with their PID, process name, nice level, and scheduling class, sorted in descending order by nice level. Processes that display **TS** in the **CLS** scheduling class column, run under the SCHED_NORMAL scheduling policy. Processes with a dash (**-**) as their nice level, run under other scheduling policies and are interpreted as a higher priority by the scheduler. Details of the additional scheduling policies are beyond the scope of this course.

```
[user@host ~]$ ps axo pid,comm,nice,cls --sort=-nice
  PID COMMAND          NI CLS
   30 khugepaged       19  TS
   29 ksmd              5  TS
    1 systemd           0  TS
    2 kthreadd          0  TS
    9 ksoftirqd/0       0  TS
   10 rcu_sched         0  TS
   11 migration/0       -  FF
   12 watchdog/0        -  FF
...output omitted...
```

STARTING PROCESSES WITH DIFFERENT NICE LEVELS

During process creation, a process inherit its parent's nice level. When a process is started from the command line, it will inherit its nice level from the shell process where it was started. Typically, this results in new processes running with a nice level of 0.

The following example starts a process from the shell, and displays the process's nice value. Note the use of the PID option in the **ps** to specify the output requested.

```
[user@host ~]$ sha1sum /dev/zero &
[1] 3480
[user@host ~]$ ps -o pid,comm,nice 3480
  PID COMMAND          NI
 3480 sha1sum           0
```

The **nice** command can be used by all users to start commands with a default or higher nice level. Without options, the **nice** command starts a process with the default nice value of 10.

The following example starts the **sha1sum** command as a background job with the default nice level and displays the process's nice level:

```
[user@host ~]$ nice sha1sum /dev/zero &
[1] 3517
[user@host ~]$ ps -o pid,comm,nice 3517
  PID COMMAND          NI
 3517 sha1sum          10
```

Use the **-n** option to apply a user-defined nice level to the starting process. The default is to add 10 to the process' current nice level. The following example starts a command as a background job with a user-defined nice value and displays the process's nice level:

```
[user@host ~]$ nice -n 15 sha1sum &
[1] 3521
[user@host ~]$ ps -o pid,comm,nice 3521
  PID COMMAND          NI
 3521 sha1sum          15
```

IMPORTANT

Unprivileged users may only increase the nice level from its current value, to a maximum of 19. Once increased, unprivileged users cannot reduce the value to return to the previous nice level. The `root` use may reduce the nice level from any current level, to a minimum of -20.

CHANGING THE NICE LEVEL OF AN EXISTING PROCESS

The nice level of an existing process can be changed using the **renice** command. This example uses the PID identifier from the previous example to change from the current nice level of 15 to the desired nice level of 19.

```
[user@host ~]$ renice -n 19 3521
3521 (process ID) old priority 15, new priority 19
```

The **top** command can also be used to change the nice level on a process. From within the **top** interactive interface, press the **r** option to access the **renice** command, followed by the PID to be changed and the new nice level.

REFERENCES

nice(1), **renice**(1), **top**(1), and **sched_setscheduler**(2) man pages.

GUIDED EXERCISE

INFLUENCING PROCESS SCHEDULING

In this exercise, you will adjust the scheduling priority of processes with the **nice** and **renice** commands and observe the effects this has on process execution.

OUTCOMES

You should be able to adjust scheduling priorities for processes.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **lab tuning-procscheduling start** command. The command runs a start script to determine if the `servera` host is reachable on the network.

```
[student@workstation ~]$ lab tuning-procscheduling start
```

1. From `workstation` use SSH to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

2. Determine the number of CPU cores on `servera` and then start two instances of the **sha1sum /dev/zero &** command for each core.

   2.1. Use **grep** to parse the number of existing virtual processors (CPU cores) from the **/proc/cpuinfo** file.

```
[student@servera ~]$ grep -c '^processor' /proc/cpuinfo
2
```

   2.2. Use a looping command to start multiple instances of the **sha1sum /dev/zero &** command. Start two per virtual processor found in the previous step. In this example, that would be four instances. The PID values in your output will vary from the example.

```
[student@servera ~]$ for i in $(seq 1 4); do sha1sum /dev/zero & done
[1] 2643
[2] 2644
[3] 2645
[4] 2646
```

3. Verify that the background jobs are running for each of the `sha1sum` processes.

```
[student@servera ~]$ jobs
[1]   Running                 sha1sum /dev/zero &
[2]   Running                 sha1sum /dev/zero &
[3]-  Running                 sha1sum /dev/zero &
[4]+  Running                 sha1sum /dev/zero &
```

4. Use the **ps** and **pgrep** commands to display the percentage of CPU usage for each `sha1sum` process.

```
[student@servera ~]$ ps u $(pgrep sha1sum)
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
student   2643 49.8  0.0 228360  1744 pts/0    R    11:15   6:09 sha1sum /
dev/zero
student   2644 49.8  0.0 228360  1780 pts/0    R    11:15   6:09 sha1sum /
dev/zero
student   2645 49.8  0.0 228360  1748 pts/0    R    11:15   6:09 sha1sum /
dev/zero
student   2646 49.8  0.0 228360  1780 pts/0    R    11:15   6:09 sha1sum /
dev/zero
```

5. Terminate all `sha1sum` processes, then verify that there are no running jobs.

   5.1. Use the **pkill** command to terminate all running processes with the name pattern `sha1sum`.

```
[student@servera ~]$ pkill sha1sum
[2]   Terminated              sha1sum /dev/zero
[4]+  Terminated              sha1sum /dev/zero
[1]-  Terminated              sha1sum /dev/zero
[3]+  Terminated              sha1sum /dev/zero
```

   5.2. Verify that there are no running jobs.

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

6. Start multiple instances of **sha1sum /dev/zero &**, then start one additional instance of **sha1sum /dev/zero &** with a nice level of 10. Start at least as many instances as the

system has virtual processors. In this example, 3 regular instances are started, plus another with the higher nice level.

6.1. Use looping to start three instances of **sha1sum /dev/zero &**.

```
[student@servera ~]$ for i in $(seq 1 3); do sha1sum /dev/zero & done
[1] 1947
[2] 1948
[3] 1949
```

6.2. Use the **nice** command to start the fourth instance with a 10 nice level.

```
[student@servera ~]$ nice -n 10 sha1sum /dev/zero &
[4] 1953
```

7. Use the **ps** and **pgrep** commands to display the PID, percentage of CPU usage, nice value, and executable name for each process. The instance with the nice value of 10 should display a lower percentage of CPU usage than the other instances.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
  PID %CPU  NI COMMAND
 1947 66.0   0 sha1sum
 1948 65.7   0 sha1sum
 1949 66.1   0 sha1sum
 1953  6.7  10 sha1sum
```

8. Use the **sudo renice** command to lower the nice level of a process from the previous step. Note the PID value from the process instance with the nice level of 10. Use that process PID to lower its nice level to 5.

```
[student@servera ~]$ sudo renice -n 5 1953
[sudo] password for student: student
1953 (process ID) old priority 10, new priority 5
```

9. Repeat the **ps** and **pgrep** commands to redisplay the CPU percent and nice level.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
  PID %CPU  NI COMMAND
 1947 63.8   0 sha1sum
 1948 62.8   0 sha1sum
 1949 65.3   0 sha1sum
 1953  9.1   5 sha1sum
```

10. Exit from servera.

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

Finish

On `workstation`, run the **`lab tuning-procscheduling finish`** script to finish this exercise.

```
[student@workstation ~]$ lab tuning-procscheduling finish
```

This concludes the guided exercise.

LAB

TUNING SYSTEM PERFORMANCE

PERFORMANCE CHECKLIST

In this lab, you will apply a specific tuning profile and adjust the scheduling priority of an existing process with high CPU usage.

OUTCOMES

You should be able to:

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab tuning-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

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

1. Change the current tuning profile for `serverb` to `balanced`, a general non-specialized tuned profile.
2. Two processes on `serverb` are consuming a high percentage of CPU usage. Adjust each process's `nice` level to 10 to allow more CPU time for other processes.

Evaluation

On `workstation`, run the **`lab tuning-review grade`** command to confirm success of this lab exercise.

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

Finish

On `workstation`, run the **`lab tuning-review finish`** script to complete this exercise.

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

This concludes the lab.

SOLUTION

TUNING SYSTEM PERFORMANCE

PERFORMANCE CHECKLIST

In this lab, you will apply a specific tuning profile and adjust the scheduling priority of an existing process with high CPU usage.

OUTCOMES

You should be able to:

- Activate a specific tuning profile for a computer system.
- Adjust the CPU scheduling priority of a process.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab tuning-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

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

1. Change the current tuning profile for `serverb` to `balanced`, a general non-specialized tuned profile.

   1.1. From `workstation`, open an SSH session to `serverb` as `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

   1.2. Use **yum** to confirm that the *tuned* package is installed.

   ```
   [student@serverb ~]$ yum list tuned
   ...output omitted...
   Installed Packages
   tuned.noarch                    2.10.0-15.el8
    @anaconda
   ```

   1.3. Use the **`systemctl is-enabled tuned`** command to display the `tuned` service enablement state.

```
[student@serverb ~]$ systemctl is-enabled tuned
enabled
```

1.4. List all available tuning profiles and their descriptions. Note that the current active profile is `virtual-guest`.

```
[student@serverb ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- balanced              - General non-specialized tuned profile
- desktop               - Optimize for the desktop use-case
- latency-performance   - Optimize for deterministic performance at
 the cost of
                          increased power consumption
- network-latency       - Optimize for deterministic performance at
 the cost of
                          increased power consumption, focused on low
 latency
                          network performance
- network-throughput    - Optimize for streaming network throughput,
 generally
                          only necessary on older CPUs or 40G+
 networks
- powersave             - Optimize for low power consumption
- throughput-performance - Broadly applicable tuning that provides
 excellent
                          performance across a variety of common
 server workloads
- virtual-guest         - Optimize for running inside a virtual guest
- virtual-host          - Optimize for running KVM guests
Current active profile: virtual-guest
```

1.5. Change the current active tuning profile to the `balanced` profile.

```
[student@serverb ~]$ sudo tuned-adm profile balanced
```

1.6. List summary information of the current active tuned profile.

Use the **`tuned-adm profile_info`** command to confirm that the active profile is the `balanced` profile.

```
[student@serverb ~]$ sudo tuned-adm profile_info
Profile name:
balanced

Profile summary:
General non-specialized tuned profile
...output omitted...
```

2. Two processes on `serverb` are consuming a high percentage of CPU usage. Adjust each process's `nice` level to 10 to allow more CPU time for other processes.

   2.1. Determine the top two CPU consumers on `serverb`. The top CPU consumers are listed last in the command output. CPU percentage values will vary.

```
[student@serverb ~]$ ps aux --sort=pcpu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
 COMMAND
...output omitted...
root      2983  100  0.0 228360  1744 ?        R<   21:08
 0:23 md5sum /dev/zero
root      2967  101  0.0 228360  1732 ?        RN   21:08
 0:23 sha1sum /dev/zero
[student@serverb ~]$
```

   2.2. Identify the current `nice` level for each of the top two CPU consumers.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum;pgrep
 md5sum)
  PID %CPU  NI COMMAND
 2967 99.6   2 sha1sum
 2983 99.7  -2 md5sum
```

   2.3. Use the **`sudo renice -n 10 2967 2983`** command to adjust the `nice` level for each process to **`10`**. Use PID values identified in the previous command output.

```
[student@serverb ~]$ sudo renice -n 10 2967 2983
[sudo] password for student: student
2967 (process ID) old priority 2, new priority 10
2983 (process ID) old priority -2, new priority 10
```

   2.4. Verify that the current `nice` level for each process is 10.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum;pgrep
 md5sum)
  PID %CPU  NI COMMAND
 2967 99.6  10 sha1sum
 2983 99.7  10 md5sum
```

   2.5. Exit from `serverb`.

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

Evaluation

On `workstation`, run the **`lab tuning-review grade`** command to confirm success of this lab exercise.

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

Finish

On `workstation`, run the **`lab tuning-review finish`** script to complete this exercise.

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

This concludes the lab.

SUMMARY

In this chapter, you learned:

- A signal is a software interrupt that reports events to an executing program. The **`kill`**, **`pkill`**, and **`killall`** commands use signals to control processes.
- The `tuned` service will automatically modify device settings to meet specific system needs based on a pre-defined selected tuning profile.
- To revert all changes made to the system settings by a selected profile, either switch to another profile or deactivate the `tuned` service.
- A relative priority is assigned to a process to determine its CPU access. This priority is called the nice value of a process.
- The **`nice`** command assigns a priority to a process when it starts. The **`renice`** command modifies the priority of a running process.

CHAPTER 7

INSTALLING AND UPDATING SOFTWARE PACKAGES

GOAL

Download, install, update, and manage software packages from Red Hat and Yum package repositories.

OBJECTIVES

- Register a system to your Red Hat account and assign it entitlements for software updates and support services using Red Hat Subscription Management.
- Find, install, and update software packages using the **yum** command.
- Enable and disable use of Red Hat or third-party Yum repositories by a server.
- Explain how modules allow installation of specific versions of software, list, enable, and switch module streams, and install and update packages from a module.

SECTIONS

- Registering Systems for Red Hat Support (and Quiz)
- Installing and Updating Software Packages with Yum (and Guided Exercise)
- Enabling Yum Software Repositories (and Guided Exercise)
- Managing Package Module Streams (and Guided Exercise)

LAB

Installing and Updating Software Packages

REGISTERING SYSTEMS FOR RED HAT SUPPORT

OBJECTIVES

After completing this section, you should be able to register a system to your Red Hat account and assign it entitlements for software updates and support services using Red Hat Subscription Management.

RED HAT SUBSCRIPTION MANAGEMENT

Red Hat Subscription Management provides tools that can be used to entitle machines to product subscriptions, allowing administrators to get updates to software packages and track information about support contracts and subscriptions used by the systems. Standard tools such as ***PackageKit*** and **`yum`** can obtain software packages and updates through a content distribution network provided by Red Hat.

There are four basic tasks performed with Red Hat Subscription Management tools:

- **Register** a system to associate that system to a Red Hat account. This allows Subscription Manager to uniquely inventory the system. When no longer in use, a system may be unregistered.
- **Subscribe** a system to entitle it to updates for selected Red Hat products. Subscriptions have specific levels of support, expiration dates, and default repositories. The tools can be used to either auto-attach or select a specific entitlement. As needs change, subscriptions may be removed.
- **Enable repositories** to provide software packages. Multiple repositories are enabled by default with each subscription, but other repositories such as updates or source code can be enabled or disabled as needed.
- **Review and track** entitlements that are available or consumed. Subscription information can be viewed locally on a specific system or, for an account, in either the Red Hat Customer Portal Subscriptions page or the *Subscription Asset Manager (SAM)*.

Registering a System

There are a number of different ways to register a system with Red Hat Customer Portal. There is a graphical interface that you can access with a GNOME application or through the Web Console service, and there is a command-line tool.

To register a system with the GNOME application, launch ***Red Hat Subscription Manager*** by selecting ***Activities***. Type *subscription* in the ***Type to search...*** field and click on ***Red Hat Subscription Manager***. Enter the appropriate password when prompted to authenticate. This displays the following ***Subscriptions*** window:

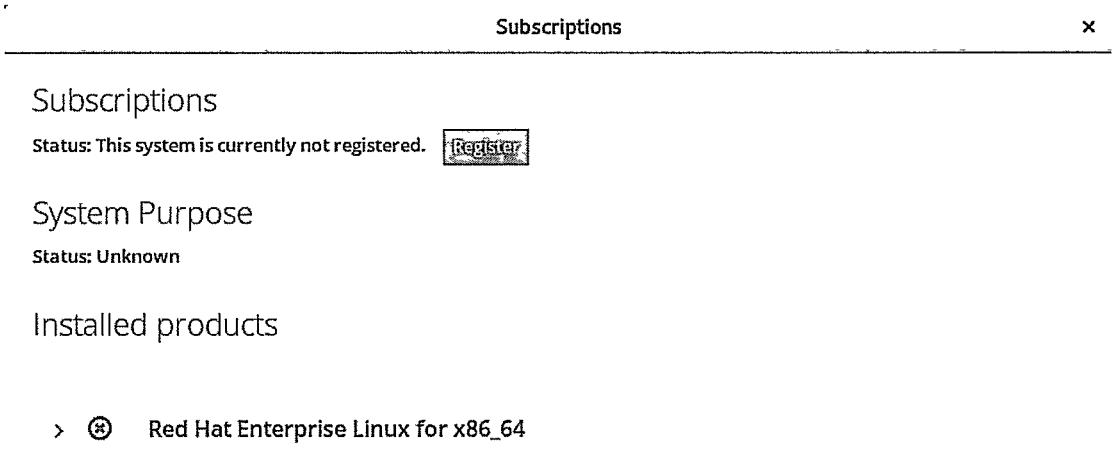


Figure 7.1: The main window of Red Hat Subscription Manager

To register the system, click the ***Register*** button in the ***Subscriptions*** window. This displays the following dialog:

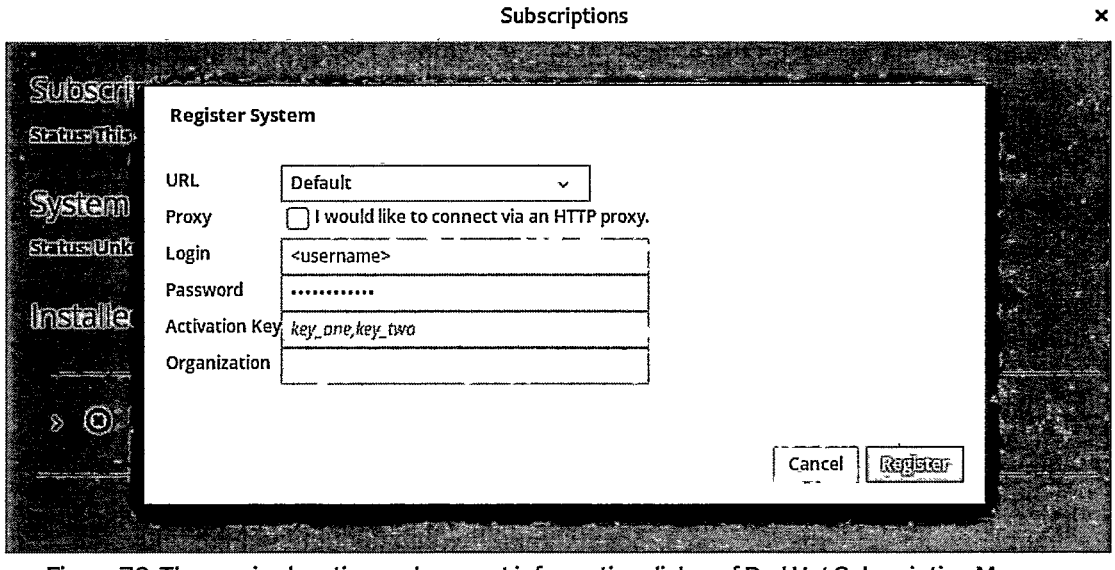


Figure 7.2: The service location and account information dialog of Red Hat Subscription Manager

This dialog box registers a system with a subscription server. By default, it registers the server to Red Hat Customer Portal. Provide the ***Login*** and ***Password*** for the Red Hat Customer Portal account to which the system should be registered, and click the ***Register*** button.

When registered, the system automatically has a subscription attached if one is available.

After the system is registered and a subscription has been assigned, close the ***Subscriptions*** window. The system is now properly subscribed and ready to receive updates or install new software from Red Hat.

REGISTRATION FROM THE COMMAND LINE

Use the **`subscription-manager`**(8) to register a system without using a graphical environment. The **`subscription-manager`** command can automatically attach a system to the best-matched compatible subscriptions for the system.

- Register a system to a Red Hat account:

```
[user@host ~]$ subscription-manager register --username=yourusername \
--password=yourpassword
```

- View available subscriptions:

```
[user@host ~]$ subscription-manager list --available | less
```

- Auto-attach a subscription:

```
[user@host ~]$ subscription-manager attach --auto
```

- Alternatively, attach a subscription from a specific pool from the list of available subscriptions:

```
[user@host ~]$ subscription-manager attach --pool=poolID
```

- View consumed subscriptions:

```
[user@host ~]$ subscription-manager list --consumed
```

- Unregister a system:

```
[user@host ~]$ subscription-manager unregister
```

NOTE

`subscription-manager` can also be used in conjunction with *activation keys*, allowing registration and assignment of predefined subscriptions, without using a username or password. This method of registration can be very useful for automated installations and deployments. Activation keys are often issued by an on-premise subscription management service, such as Subscription Asset Manager or Red Hat Satellite, and are not discussed in detail in this course.

ENTITLEMENT CERTIFICATES

An entitlement is a subscription that has been attached to a system. Digital certificates are used to store current information about entitlements on the local system. Once registered, entitlement certificates are stored in **`/etc/pki`** and its subdirectories.

- **`/etc/pki/product`** contains certificates indicating which Red Hat products are installed on the system.
- **`/etc/pki/consumer`** contains certificates identifying the Red Hat account to which the system is registered.
- **`/etc/pki/entitlement`** contains certificates indicating which subscriptions are attached to the system.

The certificates can be inspected with the **`rct`** utility directly, but the **`subscription-manager`** tools provide easier ways to examine the subscriptions attached to the system.

REFERENCES

`subscription-manager`(8) and `rct`(8) man pages

Get started with Red Hat Subscription Management
https://access.redhat.com/site/articles/433903

QUIZ

REGISTERING SYSTEMS FOR RED HAT SUPPORT

Choose the correct answer to the following questions:

1. **Which command is used to register a system without using a graphical environment?**
   a. `rct`
   b. `subscription-manager`
   c. `rpm`
   d. `yum`

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 task(s) can be performed with Red Hat Subscription Management tools?**
   a. Register a system.
   b. Subscribe a system.
   c. Enable repositories.
   d. Review and track entitlements.
   e. All of the above.

SOLUTION

REGISTERING SYSTEMS FOR RED HAT SUPPORT

Choose the correct answer to the following questions:

1. **Which command is used to register a system without using a graphical environment?**
   a. `rct`
   b. `subscription-manager`
   c. `rpm`
   d. `yum`

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 task(s) can be performed with Red Hat Subscription Management tools?**
   a. Register a system.
   b. Subscribe a system.
   c. Enable repositories.
   d. Review and track entitlements.
   e. All of the above.

INSTALLING AND UPDATING SOFTWARE PACKAGES WITH YUM

OBJECTIVES

After completing this section, you should be able to find, install, and update software packages, using the **yum** command.

MANAGING SOFTWARE PACKAGES WITH YUM

The low-level **rpm** command can be used to install packages, but it is not designed to work with package repositories or resolve dependencies from multiple sources automatically.

Yum is designed to be a better system for managing RPM-based software installation and updates. The **yum** command allows you to install, update, remove, and get information about software packages and their dependencies. You can get a history of transactions performed and work with multiple Red Hat and third-party software repositories.

Finding Software with Yum

- **yum help** displays usage information.
- **yum list** displays installed and available packages.

```
[user@host ~]$ yum list 'http*'
Available Packages
http-parser.i686              2.8.0-2.el8                        rhel8-appstream
http-parser.x86_64            2.8.0-2.el8                        rhel8-appstream
httpcomponents-client.noarch  4.5.5-4.module+el8+2452+b359bfcd   rhel8-appstream
httpcomponents-core.noarch    4.4.10-3.module+el8+2452+b359bfcd  rhel8-appstream
httpd.x86_64                  2.4.37-7.module+el8+2443+605475b7  rhel8-appstream
httpd-devel.x86_64            2.4.37-7.module+el8+2443+605475b7  rhel8-appstream
httpd-filesystem.noarch       2.4.37-7.module+el8+2443+605475b7  rhel8-appstream
httpd-manual.noarch           2.4.37-7.module+el8+2443+605475b7  rhel8-appstream
httpd-tools.x86_64            2.4.37-7.module+el8+2443+605475b7  rhel8-appstream
```

- **yum search** ***KEYWORD*** lists packages by keywords found in the name and summary fields only.

  To search for packages that have "web server" in their name, summary, and description fields, use **search all**:

```
[user@host ~]$ yum search all 'web server'
================= Summary & Description Matched: web server ====================
pcp-pmda-weblog.x86_64 : Performance Co-Pilot (PCP) metrics from web server logs
nginx.x86_64 : A high performance web server and reverse proxy server
======================== 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
libcurl.x86_64 : A library for getting files from web servers
====================== Description Matched: web server =========================
```

```
httpd.x86_64 : Apache HTTP Server
git-instaweb.x86_64 : Repository browser in gitweb
...output omitted...
```

- **yum info *PACKAGENAME*** returns detailed information about a package, including the disk space needed for installation.

  To get information on the Apache HTTP Server:

```
[user@host ~]$ yum info httpd
Available Packages
Name         : httpd
Version      : 2.4.37
Release      : 7.module+el8+2443+605475b7
Arch         : x86_64
Size         : 1.4 M
Source       : httpd-2.4.37-7.module+el8+2443+605475b7.src.rpm
Repo         : rhel8-appstream
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.
```

- **yum provides *PATHNAME*** displays packages that match the path name specified (which often include wildcard characters).

  To find packages that provide the **/var/www/html** directory, use:

```
[user@host ~]$ yum provides /var/www/html
httpd-filesystem-2.4.37-7.module+el8+2443+605475b7.noarch : The basic directory
 layout for the Apache HTTP server
Repo        : rhel8-appstream
Matched from:
Filename    : /var/www/html
```

Installing and removing software with yum

- **yum install *PACKAGENAME*** obtains and installs a software package, including any dependencies.

```
[user@host ~]$ yum install httpd
Dependencies resolved.
================================================================================
 Package                  Arch       Version              Repository        Size
================================================================================
Installing:
 httpd                    x86_64     2.4.37-7.module...   rhel8-appstream   1.4 M
Installing dependencies:
 apr                      x86_64     1.6.3-8.el8          rhel8-appstream   125 k
 apr-util                 x86_64     1.6.1-6.el8          rhel8-appstream   105 k
...output omitted...
Transaction Summary
```

```
================================================================================
Install  9 Packages

Total download size: 2.0 M
Installed size: 5.4 M
Is this ok [y/N]: y
Downloading Packages:
(1/9): apr-util-bdb-1.6.1-6.el8.x86_64.rpm          464 kB/s |  25 kB     00:00
(2/9): apr-1.6.3-8.el8.x86_64.rpm                   1.9 MB/s | 125 kB     00:00
(3/9): apr-util-1.6.1-6.el8.x86_64.rpm              1.3 MB/s | 105 kB     00:00
...output omitted...
Total                                               8.6 MB/s | 2.0 MB     00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
  Installing       : apr-1.6.3-8.el8.x86_64                                 1/9
  Running scriptlet: apr-1.6.3-8.el8.x86_64                                 1/9
  Installing       : apr-util-bdb-1.6.1-6.el8.x86_64                        2/9
...output omitted...
Installed:
  httpd-2.4.37-7.module+el8+2443+605475b7.x86_64 apr-util-bdb-1.6.1-6.el8.x86_64
  apr-util-openssl-1.6.1-6.el8.x86_64            apr-1.6.3-8.el8.x86_64
...output omitted...
Complete!
```

- **`yum update`** ***PACKAGENAME*** obtains and installs a newer version of the specified package, including any dependencies. Generally the process tries to preserve configuration files in place, but in some cases, they may be renamed if the packager thinks the old one will not work after the update. With no PACKAGENAME specified, it installs all relevant updates.

```
[user@host ~]$ sudo yum update
```

Since a new kernel can only be tested by booting to that kernel, the package is specifically designed so that multiple versions may be installed at once. If the new kernel fails to boot, the old kernel is still available. Using **`yum update kernel`** will actually *install* the new kernel. The configuration files hold a list of packages to *always install* even if the administrator requests an update.

NOTE

Use **yum list kernel** to list all installed and available kernels. To view the currently running kernel, use the **uname** command. The **-r** option only shows the kernel version and release, and the **-a** option shows the kernel release and additional information.

```
[user@host ~]$ yum list kernel
Installed Packages
kernel.x86_64          4.18.0-60.el8          @anaconda
kernel.x86_64          4.18.0-67.el8          @rhel-8-for-x86_64-baseos-
htb-rpms
[user@host ~]$ uname -r
4.18.0-60.el8.x86_64
[user@host ~]$ uname -a
Linux host.lab.example.com 4.18.0-60.el8.x86_64 #1 SMP Fri Jan 11
 19:08:11 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

- **yum remove *PACKAGENAME*** removes an installed software package, including any supported packages.

```
[user@host ~]$ sudo yum remove httpd
```

WARNING

The **yum remove** command removes the packages listed *and any package that requires the packages being removed* (and packages which require those packages, and so on). This can lead to unexpected removal of packages, so carefully review the list of packages to be removed.

Installing and removing groups of software with yum

- **yum** also has the concept of *groups*, which are collections of related software installed together for a particular purpose. In Red Hat Enterprise Linux 8, there are two kinds of groups. Regular groups are collections of packages. *Environment groups* are collections of regular groups. The packages or groups provided by a group may be **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).

  Like **yum list**, the **yum group list** command shows the names of installed and available groups.

```
[user@host ~]$ yum group list
Available Environment Groups:
   Server with GUI
   Minimal Install
   Server
...output omitted...
Available Groups:
   Container Management
```

```
   .NET Core Development
   RPM Development Tools
...output omitted...
```

Some groups are normally installed through environment groups and are hidden by default. List these hidden groups with the **`yum group list hidden`** command.

- **`yum group info`** displays information about a group. It includes a list of mandatory, default, and optional package names.

```
[user@host ~]$ yum group info "RPM Development Tools"
Group: RPM Development Tools
 Description: These tools include core development tools such rpmbuild.
  Mandatory Packages:
    redhat-rpm-config
    rpm-build
  Default Packages:
    rpmdevtools
  Optional Packages:
    rpmlint
```

- **`yum group install`** installs a group that installs its mandatory and default packages and the packages they depend on.

```
[user@host ~]$ sudo yum group install "RPM Development Tools"
...output omitted...
Installing Groups:
 RPM Development Tools

 Transaction Summary
 ================================================================================
 Install  64 Packages

 Total download size: 21 M
 Installed size: 62 M
 Is this ok [y/N]: y
```

IMPORTANT

The behavior of Yum groups changed starting in Red Hat Enterprise Linux 7. In RHEL 7 and later, groups are treated as *objects*, and are tracked by the system. If an installed group is updated, and new mandatory or default packages have been added to the group by the Yum repository, those new packages are installed upon update.

RHEL 6 and earlier consider a group to be installed if all its mandatory packages have been 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. The command **`yum group mark install GROUPNAME`** can be used to mark a group as installed, and any missing packages and their dependencies are installed upon the next update.

Finally, RHEL 6 and earlier 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 command **`yum group list`** did not.

Viewing transaction history

- All install and remove transactions are logged in **`/var/log/dnf.rpm.log`**.

```
[user@host ~]$ tail -5 /var/log/dnf.rpm.log
2019-02-26T18:27:00Z SUBDEBUG Installed: rpm-build-4.14.2-9.el8.x86_64
2019-02-26T18:27:01Z SUBDEBUG Installed: rpm-build-4.14.2-9.el8.x86_64
2019-02-26T18:27:01Z SUBDEBUG Installed: rpmdevtools-8.10-7.el8.noarch
2019-02-26T18:27:01Z SUBDEBUG Installed: rpmdevtools-8.10-7.el8.noarch
2019-02-26T18:38:40Z INFO --- logging initialized ---
```

- **`yum history`** displays a summary of install and remove transactions.

```
[user@host ~]$ sudo yum history
ID     | Command line             | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
     7 | group install RPM Develo | 2019-02-26 13:26 | Install        |   65
     6 | update kernel            | 2019-02-26 11:41 | Install        |    4
     5 | install httpd            | 2019-02-25 14:31 | Install        |    9
     4 | -y install @base firewal | 2019-02-04 11:27 | Install        |  127 EE
     3 | -C -y remove firewalld - | 2019-01-16 13:12 | Removed        |   11 EE
     2 | -C -y remove linux-firmw | 2019-01-16 13:12 | Removed        |    1
     1 |                          | 2019-01-16 13:05 | Install        |  447 EE
```

- The **`history undo`** option reverses a transaction.

```
[user@host ~]$ sudo yum history undo 5
Undoing transaction 7, from Tue 26 Feb 2019 10:40:32 AM EST
    Install apr-1.6.3-8.el8.x86_64                              @rhel8-appstream
    Install apr-util-1.6.1-6.el8.x86_64                         @rhel8-appstream
    Install apr-util-bdb-1.6.1-6.el8.x86_64                     @rhel8-appstream
    Install apr-util-openssl-1.6.1-6.el8.x86_64                 @rhel8-appstream
    Install httpd-2.4.37-7.module+el8+2443+605475b7.x86_64      @rhel8-appstream
...output omitted...
```

SUMMARY OF YUM COMMANDS

Packages can be located, installed, updated, and removed by name or by package groups.

TASK:	COMMAND:
List installed and available packages by name	`yum list [NAME-PATTERN]`
List installed and available groups	`yum group list`
Search for a package by keyword	`yum search KEYWORD`
Show details of a package	`yum info PACKAGENAME`
Install a package	`yum install PACKAGENAME`
Install a package group	`yum group install GROUPNAME`
Update all packages	`yum update`
Remove a package	`yum remove PACKAGENAME`
Display transaction history	`yum history`

REFERENCES

`yum`(1) and `yum.conf`(5) man pages

For more information, refer to the *Installing software with yum* chapter in the *Red Hat Enterprise Linux 8.0 Configuring basic system settings* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/installing-software-with-yum_configuring-basic-system-settings

GUIDED EXERCISE

INSTALLING AND UPDATING SOFTWARE PACKAGES WITH YUM

In this exercise, you will install and remove packages and package groups.

OUTCOMES

You should be able to install and remove packages with dependencies.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation` run the **`lab software-yum start`** command. The command runs a start script that determines if the host, `servera`, is reachable on the network.

```
[student@workstation ~]$ lab software-yum start
```

1. Use the **ssh** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to `servera`.

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

2. Use the **su -** to switch to the `root` user at the shell prompt.

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

3. Search for a specific package.

   3.1. Attempt to run the command **`guile`**. You should find that it is not installed.

```
[root@servera ~]# guile
-bash: guile: command not found
```

   3.2. Use the **`yum search`** command to search for packages that have **`guile`** as part of their name or summary.

```
[root@servera ~]# yum search guile
========================= Name Exactly Matched: guile
 =========================
guile.i686 : A GNU implementation of Scheme for application
 extensibility
guile.x86_64 : A GNU implementation of Scheme for application
 extensibility
```

3.3. Use the **yum info** command to find out more information about the **guile** package.

```
[root@servera ~]# yum info guile
Available Packages
Name         : guile
Epoch        : 5
Version      : 2.0.14
Release      : 7.el8
...output omitted...
```

4. Use the **yum install** command to install the **guile** package.

```
[root@servera ~]# yum install guile
...output omitted...
Dependencies resolved.
================================================================================
 Package       Arch   Version          Repository
 Size
================================================================================
Installing:
 guile         x86_64 5:2.0.14-7.el8   rhel-8.0-for-x86_64-appstream-rpms
 3.5 M
Installing dependencies:
 gc            x86_64 7.6.4-3.el8      rhel-8.0-for-x86_64-appstream-rpms
 109 k
 libatomic_ops x86_64 7.6.2-3.el8      rhel-8.0-for-x86_64-appstream-rpms
 38 k
 libtool-ltdl  x86_64 2.4.6-25.el8     rhel-8.0-for-x86_64-baseos-rpms
 58 k

Transaction Summary
================================================================================
Install  4 Packages

Total download size: 3.7 M
Installed size: 12 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

5. Remove packages.

5.1. Use the **yum remove** command to remove the **guile** package, but respond with **no** when prompted. How many packages would be removed?

```
[root@servera ~]# yum remove guile
...output omitted...
Dependencies resolved.
================================================================================
 Package        Arch   Version          Repository
     Size
================================================================================
Removing:
 guile          x86_64 5:2.0.14-7.el8  @rhel-8.0-for-x86_64-appstream-
rpms  12 M
Removing unused dependencies:
 gc             x86_64 7.6.4-3.el8     @rhel-8.0-for-x86_64-appstream-
rpms 221 k
 libatomic_ops x86_64 7.6.2-3.el8     @rhel-8.0-for-x86_64-appstream-
rpms  75 k
 libtool-ltdl  x86_64 2.4.6-25.el8    @rhel-8.0-for-x86_64-baseos-rpms
     69 k

Transaction Summary
================================================================================
Remove 4 Packages

Freed space: 12 M
Is this ok [y/N]: n
Operation aborted.
```

5.2. Use the **yum remove** command to remove the **gc** package, but respond with **no** when prompted. How many packages would be removed?

```
[root@servera ~]# yum remove gc
...output omitted...
Dependencies resolved.
================================================================================
 Package        Arch   Version          Repository
     Size
================================================================================
Removing:
 gc             x86_64 7.6.4-3.el8     @rhel-8.0-for-x86_64-appstream-
rpms 221 k
Removing dependent packages:
 guile          x86_64 5:2.0.14-7.el8  @rhel-8.0-for-x86_64-appstream-
rpms  12 M
Removing unused dependencies:
 libatomic_ops x86_64 7.6.2-3.el8     @rhel-8.0-for-x86_64-appstream-
rpms  75 k
 libtool-ltdl  x86_64 2.4.6-25.el8    @rhel-8.0-for-x86_64-baseos-rpms
     69 k

Transaction Summary
```

```
================================================================================
Remove  4 Packages

Freed space: 12 M
Is this ok [y/N]: n
Operation aborted.
```

6. Gather information about the "RPM Development Tools" component group and install it on `servera`.

6.1. Use the **`yum group list`** command to list all available component groups.

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

6.2. Use the **`yum group info`** command to find out more information about the RPM `Development Tools` component group, including a list of included packages.

```
[root@servera ~]# yum group info "RPM Development Tools"
Group: RPM Development Tools
 Description: These tools include core development tools such
 rpmbuild.
 Mandatory Packages:
   redhat-rpm-config
   rpm-build
 Default Packages:
   rpmdevtools
 Optional Packages:
   rpmlint
```

6.3. Use the **`yum group install`** command to install the RPM `Development Tools` component group.

```
[root@servera ~]# yum group install "RPM Development Tools"
Dependencies resolved.
================================================================================
 Package          Arch   Version       Repository
     Size
================================================================================
Installing group/module packages:
 redhat-rpm-config
                  noarch 115-1.el8     rhel-8.0-for-x86_64-appstream-
rpms  82 k
 rpm-build        x86_64 4.14.2-9.el8  rhel-8.0-for-x86_64-appstream-
rpms 166 k
Installing dependencies:
 dwz              x86_64 0.12-9.el8    rhel-8.0-for-x86_64-appstream-
rpm  109 k
 efi-srpm-macros  noarch 3-2.el8       rhel-8.0-for-x86_64-appstream-
rpm   22 k
...output omitted...

Transaction Summary
```

```
================================================================================
Install  60 Packages

Total download size: 17 M
Installed size: 50 M
Is this ok [y/N]: y
...output omitted...
  Installing       : perl-Exporter-5.72-396.el8.noarch
      1/60
  Installing       : perl-libs-4:5.26.3-416.el8.x86_64
      2/60
  Installing       : perl-Carp-1.42-396.el8.noarch
      3/60
...output omitted...
  Verifying        : dwz-0.12-9.el8.x86_64
      1/60
  Verifying        : efi-srpm-macros-3-2.el8.noarch
      2/60
  Verifying        : gdb-headless-8.2-5.el8.x86_64
      3/60
...output omitted...
Installed:
  redhat-rpm-config-115-1.el8.noarch
  rpm-build-4.14.2-9.el8.x86_64
  rpmdevtools-8.10-7.el8.noarch
...output omitted...

Complete!
```

7. Explore the history and undo options of **yum**.

7.1. Use the **yum history** command to display recent **yum** history.

```
[root@servera ~]# yum history
ID     | Command line             | Date and time    | Action(s)
 | Altered
-------------------------------------------------------------------------------
     6 | group install RPM Develo | 2019-02-26 17:11 | Install
 |   61
     5 | install guile            | 2019-05-26 17:05 | Install
 |    4
```

```
     4 | -y install @base firewal | 2019-02-04 11:27 | Install
|  127 EE
     3 | -C -y remove firewalld - | 2019-01-16 13:12 | Removed
|   11 EE
     2 | -C -y remove linux-firmw | 2019-01-16 13:12 | Removed
|    1
     1 |                          | 2019-01-16 13:05 | Install
|  447 EE
```

7.2. Use the **`yum history info`** command to confirm that the last transaction is the group installation.

```
[root@servera ~]# yum history info 6
Transaction ID : 6
Begin time     : Tue 26 Feb 2019 05:11:25 PM EST
Begin rpmdb    : 563:bf48c46156982a78e290795400482694072f5ebb
End time       : Tue 26 Feb 2019 05:11:33 PM EST (8 seconds)
End rpmdb      : 623:bf25b424ccf451dd0a6e674fb48e497e66636203
User           : Student User <student>
Return-Code    : Success
Releasever     : 8
Command Line   : group install RPM Development Tools
Packages Altered:
    Install dwz-0.12-9.el8.x86_64          @rhel-8.0-for-x86_64-
appstream-rpms
    Install efi-srpm-macros-3-2.el8.noarch @rhel-8.0-for-x86_64-
appstream-rpms
```

7.3. Use the **`yum history undo`** command to remove the set of packages that were installed when the **`guile`** package was installed.

```
[root@servera ~]# yum history undo 5
```

8. Log out of the `servera` system.

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

Finish

On `workstation`, run the **`lab software-yum finish`** script to finish this exercise.

```
[student@workstation ~]$ lab software-yum finish
```

This concludes the guided exercise.

ENABLING YUM SOFTWARE REPOSITORIES

OBJECTIVES

After completing this section, you should be able to enable and disable use of Red Hat or third-party Yum repositories by a server.

ENABLING RED HAT SOFTWARE REPOSITORIES

Registering a system to the subscription management service automatically configures access to software repositories based on the attached subscriptions. To view all available repositories:

```
[user@host ~]$ yum repolist all
...output omitted...
rhel-8-for-x86_64-appstream-debug-rpms   ... AppStream (Debug RPMs)  disabled
rhel-8-for-x86_64-appstream-rpms         ... AppStream (RPMs)        enabled:
 5,045
rhel-8-for-x86_64-appstream-source-rpms  ... AppStream (Source RPMs) disabled
rhel-8-for-x86_64-baseos-debug-rpms      ... BaseOS (Debug RPMs)     enabled:
 2,270
rhel-8-for-x86_64-baseos-rpms            ... BaseOS (RPMs)           enabled:
 1,963
...output omitted...
```

The **yum-config-manager** command can be used to enable or disable repositories. To enable a repository, the command sets the `enabled` parameter to **1**. For example, the following command enables the **rhel-8-server-debug-rpms** repository:

```
[user@host ~]$ yum-config-manager --enable rhel-8-server-debug-rpms
Updating Subscription Management repositories.
============ repo: rhel-8-for-x86_64-baseos-debug-rpms ============
[rhel-8-for-x86_64-baseos-debug-rpms]
bandwidth = 0
baseurl = [https://cdn.redhat.com/content/dist/rhel8/8/x86_64/baseos/debug]
cachedir = /var/cache/dnf
cost = 1000
deltarpm = 1
deltarpm_percentage = 75
enabled = 1
```

Non-Red Hat sources provide software through third-party repositories, which can be accessed by the **yum** command from a website, FTP server, or the local file system. For example, Adobe provides some of its software for Linux through a Yum repository. In a Red Hat classroom, the `content.example.com` classroom server hosts Yum repositories.

To enable support for a new third-party repository, create a file in the **/etc/yum.repos.d/** directory. Repository configuration files must end with a **.repo** extension. The repository definition contains the URL of the repository, a name, whether to use GPG to check the package signatures, and if so, the URL pointing to the trusted GPG key.

Creating Yum Repositories

Create Yum repositories with the **yum-config-manager** command.
The following command creates a file named **/etc/yum.repos.d/dl.fedoraproject.org_pub_epel_8_x86_64_.repo** with the output shown.

> **WARNING**
> EPEL 8 had not yet been released by the community project when this section was written. An EPEL release can lag the Red Hat Enterprise Linux release since it is not provided by Red Hat but by a volunteer community. The below URL might not be valid, we have used the below example to illustrate how to enable third-party package repository.

```
[user@host ~]$ yum-config-manager --add-repo="http://dl.fedoraproject.org/pub/
epel/8/x86_64/"
Loaded plugins: langpacks
adding repo from: http://dl.fedoraproject.org/pub/epel/8/x86_64/

[dl.fedoraproject.org_pub_epel_8_x86_64_]
name=added from: http://dl.fedoraproject.org/pub/epel/8/x86_64/
baseurl=http://dl.fedoraproject.org/pub/epel/8/x86_64/
enabled=1
```

Modify this file to provide customized values and location of a GPG key. Keys are stored in various locations on the remote repository site, such as, `http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-8`. Administrators should download the key to a local file rather than allowing **yum** to retrieve the key from an external source. For example:

```
[EPEL]
name=EPEL 8
baseurl=http://dl.fedoraproject.org/pub/epel/8/x86_64/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8
```

RPM Configuration Packages for Local Repositories

Some repositories provide a configuration file and GPG public key as part of an RPM package that can be downloaded and installed using the **yum localinstall** command. For example, the volunteer project called Extra Packages for Enterprise Linux (EPEL) provides software not supported by Red Hat but compatible with Red Hat Enterprise Linux.

> **WARNING**
> EPEL 8 had not yet been released by the community project when this section was written. We have written this section to reflect the expected configuration process for EPEL 8, based on the EPEL 7 process, as an example of how to enable a third-party package repository that provides its repository configuration to clients by using an RPM package.

The following command installs the Red Hat Enterprise Linux 8 EPEL repository package:

```
[user@host ~]$ rpm --import http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-
EPEL-8
[user@host ~]$ yum install http://dl.fedoraproject.org/pub/epel/8/x86_64/e/epel-
release-8-2.noarch.rpm
```

Configuration 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 8 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/8/$basearch
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-8&arch=$basearch
failovermethod=priority
enabled=1
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8

[epel-debuginfo]
name=Extra Packages for Enterprise Linux 8 - $basearch - Debug
#baseurl=http://download.fedoraproject.org/pub/epel/8/$basearch/debug
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-8&arch=
$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8
gpgcheck=1

[epel-source]
name=Extra Packages for Enterprise Linux 8 - $basearch - Source
#baseurl=http://download.fedoraproject.org/pub/epel/8/SRPMS
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-source-8&arch=
$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-8
gpgcheck=1
```

To define a repository, but not search it by default, insert the `enabled=0` parameter. Repositories can be enabled and disabled persistently with **`yum-config-manager`** command or temporarily with **`yum`** command options, **`--enablerepo=`*`PATTERN`*** and **`--disablerepo=`*`PATTERN`***.

WARNING

Install the RPM GPG key before installing signed packages. This verifies that the packages belong to a key which has been imported. Otherwise, the **`yum`** command fails due to a missing key. The **`--nogpgcheck`** option can be used to ignore missing GPG keys, but this could cause forged or insecure packages to be installed on the system, potentially compromising its security.

REFERENCES

yum(1), `yum.conf`(5), and `yum-config-manager`(1) man pages

For more information, refer to the *Installing software with yum* chapter in the *Red Hat Enterprise Linux 8.0 Configuring basic system settings* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/installing-software-with-yum_configuring-basic-system-settings

GUIDED EXERCISE

ENABLING YUM SOFTWARE REPOSITORIES

In this exercise, you will configure your server to get packages from a remote Yum repository, then update or install a package from that repository.

OUTCOMES

You should be able to configure a system to obtain software updates from a classroom server and update the system to use latest packages.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation` run the **`lab software-repo start`** command. The command runs a start script that determines whether the host `servera` is reachable on the network. The script also ensures that the *yum* package is installed.

```
[student@workstation ~]$ lab software-repo start
```

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

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

2. Use the **su -** command to switch to `root` at the shell prompt.

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

3. Configure software repositories on `servera` to obtain custom packages and updates from the following URL:

   - Custom packages provided at `http://content.example.com/rhel8.0/x86_64/rhcsa-practice/rht`
   - Updates of the custom packages provided at `http://content.example.com/rhel8.0/x86_64/rhcsa-practice/errata`

   3.1. Use **`yum-config-manager`** to add the custom packages repository.

```
[root@servera ~]# yum-config-manager \
--add-repo "http://content.example.com/rhel8.0/x86_64/rhcsa-practice/
rht"
Adding repo from: http://content.example.com/rhel8.0/x86_64/rhcsa-
practice/rht
```

3.2. Examine the software repository file created by the previous command in the **/etc/yum.repos.d** directory. Use the **vim** command to edit the file and add the *gpgcheck=0* parameter to disable the GPG key check for the repository.

```
[root@servera ~]# vim \
/etc/yum.repos.d/content.example.com_rhel8.0_x86_64_rhcsa-
practice_rht.repo
[content.example.com_rhel8.0_x86_64_rhcsa-practice_rht]
name=created by dnf config-manager from http://content.example.com/
rhel8.0/x86_64/rhcsa-practice/rht
baseurl=http://content.example.com/rhel8.0/x86_64/rhcsa-practice/rht
enabled=1
gpgcheck=0
```

3.3. 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/rhel8.0/x86_64/rhcsa-practice/
errata
enabled=1
gpgcheck=0
```

3.4. Use the **yum repolist all** command to list all repositories on the system:

```
[root@servera ~]# yum repolist all
repo id                                                  repo name
 status
content.example.com_rhel8.0_x86_64_rhcsa-practice_rht  created by ....
 enabled: 2
rht-updates                                              rht updates
 enabled: 2
...output omitted...
```

4. Disable the `rht-updates` software repository and install the *rht-system* package.

4.1. Use **`yum-config-manager --disable`** to disable the `rht-updates` repository.

```
[root@servera ~]# yum-config-manager --disable rht-updates
```

4.2. List, and then install, the **`rht-system`** package:

```
[root@servera ~]# yum list rht-system
Available Packages
rht-system.noarch  1.0.0-1 content.example.com_rhel8.0_x86_64_rhcsa-
practice_rht
[root@servera ~]# yum 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!
```

4.3. Verify that the *rht-system* package is installed, and note the version number of the package.

```
[root@servera ~]# yum list rht-system
Installed Packages
rht-system.noarch  1.0.0-1 @content.example.com_rhel8.0_x86_64_rhcsa-
practice_rht
```

5. Enable the `rht-updates` software repository and update all relevant software packages.

5.1. Use **`yum-config-manager --enable`** to enable the `rht-updates` repository.

```
[root@servera ~]# yum-config-manager --enable rht-updates
```

5.2. Use the **`yum update`** command to update all software packages on `servera`.

```
[root@servera ~]# yum update
Dependencies resolved.
================================================================================
 Package            Arch           Version                Repository
     Size
================================================================================
Upgrading:
```

```
 rht-system          x86_64          1.0.0-2.el7            rht-updates
     3.9 k
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

5.3. Verify that the *rht-system* package is upgraded, and note the version number of the package.

```
[root@servera ~]# yum list rht-system
Installed Packages
rht-system.noarch      1.0.0-2.el7          @rht-updates
```

6. Exit from `servera`.

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

Finish

On `workstation`, run the **`lab software-repo finish`** script to finish this exercise. This script removes all the software repositories and packages installed on `servera` during the exercise.

```
[student@workstation ~]$ lab software-repo finish
```

This concludes the guided exercise.

MANAGING PACKAGE MODULE STREAMS

OBJECTIVES

After completing this section, you should be able to:

- Explain how modules allow installation of specific versions of software.
- How to list, enable, and switch module streams.
- Install, and update packages from a module.

INTRODUCTION TO APPLICATION STREAM

Red Hat Enterprise Linux 8.0 introduces the concept of Application Streams. Multiple versions of user space components shipped with the distribution are now delivered at the same time. They may be updated more frequently than the core operating system packages. This provides you with greater flexibility to customize Red Hat Enterprise Linux without impacting the underlying stability of the platform or specific deployments.

Traditionally, managing alternate versions of an application's software package and its related packages meant maintaining different repositories for each different version. For developers who wanted the latest version of an application and administrators who wanted the most stable version of the application, this created a situation that was tedious to manage. This process is simplified in Red Hat Enterprise Linux 8 using a new technology called *Modularity*. Modularity allows a single repository to host multiple versions of an application's package and its dependencies.

Red Hat Enterprise Linux 8 content is distributed through two main software repositories: *BaseOS* and *Application Stream (AppStream)*.

BaseOS

The BaseOS repository provides the core operating system content for Red Hat Enterprise Linux as RPM packages. BaseOS components have a life cycle identical to that of content in previous Red Hat Enterprise Linux releases.

Application Stream

The Application Stream repository provides content with varying life cycles as both modules and traditional packages. Application Stream contains necessary parts of the system, as well as a wide range of applications previously available as a part of Red Hat Software Collections and other products and programs.

> **IMPORTANT**
> Both BaseOS and AppStream are a necessary part of a Red Hat Enterprise Linux 8 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.

MODULES

A module is a set of RPM packages that are a consistent set that belong together. Typically, this is organized around a specific version of a software application or programming language. A typical module can contain packages with an application, packages with the application's specific dependency libraries, packages with documentation for the application, and packages with helper utilities.

Module Streams

Each module can have 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, only one of its streams can be enabled and provide its packages.

Module Profiles

Each module can have one or more profiles. A profile is a list of certain packages to be installed together for a particular use-case such as for a server, client, development, minimal install, or other.

Installing a particular module profile simply 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, the module will install its *default profile*.

MANAGING MODULES USING YUM

Yum version 4, new in Red Hat Enterprise Linux 8, adds support for the new modular features of Application Stream.

For handling the modular content, the **`yum module`** command has been added. Otherwise, **`yum`** works with modules much like does with regular packages.

Listing Modules

To display a list of available modules, use **`yum module list`**:

```
[user@host ~]$ yum module list
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name                    Stream        Profiles    Summary
389-ds                  1.4           default     389 Directory Server (base)
ant                     1.10 [d]      common [d] Java build tool
container-tools         1.0 [d]       common [d] Common tools and dependencies for
 container runtimes
...output omitted...
   Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

NOTE

Use the *Hint* at the end of the output to help determine which streams and profiles are enabled, disabled, installed, as well as which ones are the defaults.

To list the module streams for a specific module and retrieve their status:

```
[user@host ~]$ yum module list perl
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name  Stream       Profiles             Summary
perl  5.24         common [d], minimal  Practical Extraction and Report Language
perl  5.26 [d]     common [d], minimal  Practical Extraction and Report Language
```

To display details of a module:

```
[user@host ~]$ yum module info perl
Name             : perl
Stream           : 5.24
Version          : 820190207164249
Context          : ee766497
Profiles         : common [d], minimal
Default profiles : common
Repo             : rhel-8-for-x86_64-appstream-rpms
Summary          : Practical Extraction and Report Language
...output omitted...
Artifacts        : perl-4:5.24.4-403.module+el8+2770+c759b41a.x86_64
                 : perl-Algorithm-Diff-0:1.1903-9.module+el8+2464+d274aed1.noarch
                 : perl-Archive-Tar-0:2.30-1.module+el8+2464+d274aed1.noarch
```

NOTE

Without specifying a module stream, **`yum module info`** shows list of packages installed by default profile of a module using the default stream. Use the *module-name:stream* format to view a specific module stream. Add the **`--profile`** option to display information about packages installed by each of the module's profiles. For example:

```
[user@host ~]$ yum module info --profile perl:5.24
```

Enabling Module Streams and Installing Modules

Module streams must be enabled in order to install their module. To simplify this process, when a module is installed it enables its module stream if necessary. Module streams can be enabled manually using **`yum module enable`** and providing the name of the module stream.

IMPORTANT

Only one module stream may be enabled for a given module. Enabling an additional module stream will disable the original module stream.

Install a module using the default stream and profiles:

```
[user@host ~]$ sudo yum module install perl
Dependencies resolved.
================================================================================
 Package         Arch   Version      Repository                            Size
================================================================================
Installing group/module packages:
 perl            x86_64 4:5.26.3-416.el8
                                     rhel-8-for-x86_64-appstream-htb-rpms  72 k
Installing dependencies:
...output omitted...
Running transaction
  Preparing        :                                                        1/1
  Installing       : perl-Exporter-5.72-396.el8.noarch                    1/155
  Installing       : perl-Carp-1.42-396.el8.noarch                        2/155
...output omitted...
Installed:
  perl-4:5.26.3-416.el8.x86_64
  perl-Encode-Locale-1.05-9.el8.noarch
...output omitted...
Complete!
```

NOTE

The same results could have been accomplished by running **`yum install @perl`**. The @ notation informs **yum** that the argument is a module name instead of a package name.

To verify the status of the module stream and the installed profile:

```
[user@host ~]$ yum module list perl
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name  Stream       Profiles             Summary
perl  5.24         common, minimal      Practical Extraction and Report Language
perl  5.26 [d][e]  common [i], minimal  Practical Extraction and Report Language

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

Removing Modules and Disabling Module Streams

Removing a module removes all of the packages installed by profiles of the currently enabled module stream, and any further packages and modules that depend on these. Packages installed from this module stream not listed in any of its profiles remain installed on the system and can be removed manually.

WARNING

Removing modules and switching module streams can be a bit tricky. Switching the stream enabled for a module is equivalent to resetting the current stream and enabling the new stream. It does not automatically change any installed packages. You have to do that manually.

Directly installing a module stream that is different than the one that is currently installed is not recommended, because upgrade scripts might run during the installation that would break things with the original module stream. That could lead to data loss or other configuration issues.

Proceed with caution.

To remove an installed module:

```
[user@host ~]$ sudo yum module remove perl
Dependencies resolved.
================================================================================
 Package                        ArchVersion            Repository
           Size
================================================================================
Removing:
 perl                           x86_644:5.26.3-416.el8   @rhel-8.0-for-x86_64-
appstream-rpms 0
Removing unused dependencies:
...output omitted...
Running transaction
Preparing        :                                                         1/1
Erasing          : perl-4:5.26.3-416.el8.x86_64                          1/155
Erasing          : perl-CPAN-2.18-397.el8.noarch                         2/155
...output omitted...
Removed:
perl-4:5.26.3-416.el8.x86_64
dwz-0.12-9.el8.x86_64
...output omitted...
Complete!
```

After the module is removed, the module stream is still enabled. To verify the module stream is still enabled:

```
[user@host ~]$ yum module list perl
Red Hat Enterprise Linux 8 for x86_64 - AppStream (RPMs)
Name  Stream       Profiles             Summary
perl  5.24         common [d], minimal  Practical Extraction and Report Language
perl  5.26 [d][e]   common [d], minimal  Practical Extraction and Report Language

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

To disable the module stream:

```
[user@host ~]$ sudo yum module disable perl
  ...output omitted...
Dependencies resolved.
================================================================================
 Package           Arch             Version              Repository         Size
================================================================================
Disabling module streams:
 perl                               5.26
Is this ok [y/N]: y
Complete!
```

Switching Module Streams

Switching module streams generally requires upgrading or downgrading the content to a different version.

To ensure a clean switch, you should remove the modules provided by the module stream first. That will remove all the packages installed by the profiles of the module, and any modules and packages that those packages have dependencies on.

To list the packages installed from the module, in the example below the *postgresql:9.6* module is installed:

```
[user@host ~]$ sudo yum module info postgresql | grep module+el8 | \
sed 's/.*: //g;s/\n/ /g' | xargs yum list installed
Installed Packages
postgresql.x86_64          9.6.10-1.module+el8+2470+d1bafa0e   @rhel-8.0-for-
x86_64-appstream-rpms
postgresql-server.x86_64   9.6.10-1.module+el8+2470+d1bafa0e   @rhel-8.0-for-
x86_64-appstream-rpms
```

Remove the packages listed from the previous command. Mark the module profiles to be uninstalled.

```
[user@host ~]$ sudo yum module remove postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Removed:
  postgresql-server-9.6.10-1.module+el8+2470+d1bafa0e.x86_64
 libpq-10.5-1.el8.x86_64  postgresql-9.6.10-1.module+el8+2470+d1bafa0e.x86_64
Complete
```

After removing the module profiles reset the module stream. Use the **`yum module reset`** command to reset the module stream.

```
[user@host ~]$ sudo yum module reset postgresql
================================================================
 Package       Arch             Version     Repository      Size
================================================================
Resetting module streams:
postgresql                     9.6
```

```
Transaction Summary
==================================================================

Is this ok [y/N]: y
Complete!
```

To enable a different module stream and install the module:

```
[user@host ~]$ sudo yum module install postgresql:10
```

The new module stream will be enabled and the current stream disabled. It may be necessary to update or downgrade packages from the previous module stream that are not listed in the new profile. Use the **`yum distro-sync`** to perform this task if required. There may also be packages that remain installed from the previous module stream. Remove those using **`yum remove`**.

REFERENCES

For more information, refer to the *Using AppStream* chapter in the *Red Hat Enterprise Linux 8.0 Installing, managing, and removing user space components* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/installing_managing_and_removing_user_space_components/

Modularity

https://docs.fedoraproject.org/en-US/modularity/

GUIDED EXERCISE

MANAGING PACKAGE MODULE STREAMS

In this exercise, you will list the available modules, enable a specific module stream, and install packages from that stream.

OUTCOMES

You should be able to:

- List installed modules and examine information of a module.
- Enable and install a module from a stream.
- Switch to a specific module stream.
- Remove and disable a module.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation` run the **`lab software-module start`** command. The command runs a start script that determines whether the host, `servera`, is reachable on the network. The script also ensures that the required software repositories are available and installs the *postgresql:9.6* module.

```
[student@workstation ~]$ lab software-module start
```

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

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

2. Switch to `root` at the shell prompt.

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

3. List available modules, streams, and installed modules. Examine the information for the *python36* module.

   3.1. Use the **`yum module list`** command to list available modules and streams.

```
[root@servera ~]# yum module list
Red Hat Enterprise Linux 8.0 AppStream (dvd)
Name        Stream       Profiles           Summary
...
python27    2.7 [d]      common [d]         Python programming ...,
 version 2.7
python36    3.6 [d]      common [d], build  Python programming ...,
 version 3.6
...
Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

3.2. Use the **`yum module list --installed`** command to list installed modules and streams.

```
[root@servera ~]# yum module list --installed
Red Hat Enterprise Linux 8.0 AppStream (dvd)
Name        Stream    Profiles                Summary
...
postgresql  9.6 [e]   client, server [d] [i]   PostgreSQL server and
 client ...
...
Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

3.3. Use the **`yum module info`** command to examine details of the *python36* module.

```
[root@servera ~]# yum module info python36
Name             : python36
Stream           : 3.6 [d]
Version          : 820190123171828
Context          : 17efdbc7
Profiles         : common [d], build
Default profiles : common
Repo             : rhel-8.0-for-x86_64-appstream-rpms
Summary          : Python programming language, version 3.6
...output omitted...
Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled, [a]ctive]
```

4. Install the *python36* module from the `3.6` stream and the `common` profile. Verify the current status of the module.

4.1. Use the **`yum module install`** command to install the *python36* module. Use the **`name:stream/profile`** syntax to install the *python36* module from the `3.6` stream and the `common` profile.

> **NOTE**
> You can omit **`/profile`** to use the default profile and **`:stream`** to use the default stream.

```
[root@servera ~]# yum module install python36:3.6/common
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

4.2. Examine the current status of the *python36* module.

```
[root@servera ~]# yum module list python36
Red Hat Enterprise Linux 8.0 AppStream (dvd)
Name      Stream      Profiles                Summary
python36  3.6 [d][e]  common [d] [i], build   Python programming ...,
 version 3.6

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

5. Switch the *postgresql* module of the `server` profile to use the `10` stream.

5.1. Use the **`yum module list`** command to list the *postgresql* module and the stream. Notice that the *postgresql:9.6* module stream is currently installed.

```
[root@servera ~]# yum module list postgresql
Red Hat Enterprise Linux 8.0 AppStream (dvd)
Name          Stream    Profiles                Summary
postgresql    10 [d]    client, server [d]      PostgreSQL server and
 client ...
postgresql    9.6 [e]   client, server [d] [i]  PostgreSQL server and
 client ...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

5.2. Remove and disable the *postgresql* module stream along with all the packages installed by the profile.

```
[root@servera ~]# yum module remove postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Removed:
  postgresql-server-9.6.10-1.module+el8+2470+d1bafa0e.x86_64
   libpq-10.5-1.el8.x86_64  postgresql-9.6.10-1.module
+el8+2470+d1bafa0e.x86_64
Complete
```

5.3. Reset the *postgresql* module and its streams.

```
[root@servera ~]# yum module reset postgresql
================================================================
 Package       Arch            Version     Repository     Size
================================================================
Resetting module streams:
postgresql                    9.6

Transaction Summary
```

```
================================================================

Is this ok [y/N]: y
Complete!
```

5.4. Use the **`yum module install`** command to switch to the *postgresql:10* module stream.

```
[root@servera ~]# yum module install postgresql:10
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

5.5. Verify that the *postgresql* module is switched to the `10` stream.

```
[root@servera ~]# yum module list postgresql
Red Hat Enterprise Linux 8.0 AppStream (dvd)
Name          Stream     Profiles                Summary
postgresql    10 [d] [e] client, server [d] [i]  PostgreSQL server
 and client ...
postgresql    9.6        client, server [d]      PostgreSQL server
 and client ...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

6. Remove and disable the *postgresql* module stream along with all the packages installed by the profile.

   6.1. Use the **`yum remove module`** command to remove the *postgresql* module. The command also removes all packages installed from this module.

```
[root@servera ~]# yum module remove postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

   6.2. Disable the *postgresql* module stream.

```
[root@servera ~]# yum module disable postgresql
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

   6.3. Verify that the *postgresql* module stream is removed and disabled.

```
[root@servera ~]# yum module list postgresql
Red Hat Enterprise Linux 8.0 AppStream (dvd)
Name          Stream     Profiles                Summary
postgresql    10 [d][x]  client, server [d]      PostgreSQL server
 and client ...
postgresql    9.6 [x]    client, server [d]      PostgreSQL server
 and client ...

Hint: [d]efault, [e]nabled, [x]disabled, [i]nstalled
```

7. Exit from `servera`.

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

Finish

On `workstation`, run the **`lab software-module finish`** script to finish this exercise. This script removes all the modules installed on `servera` during the exercise.

```
[student@workstation ~]$ lab software-module finish
```

This concludes the guided exercise.

▶ LAB

INSTALLING AND UPDATING SOFTWARE PACKAGES

PERFORMANCE CHECKLIST

In this lab, you will manage software repositories and module streams, and install and upgrade packages from those repositories and streams.

OUTCOMES

You should be able to:

- Manage software repositories and module streams.
- Install and upgrade packages from repositories and streams.
- Install an RPM package.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab software-review start`** command. This script ensures that `serverb` is available. It also downloads any packages required for the lab exercise.

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

1. On `serverb` configure a software repository to obtain updates. Name the repository as `errata` and configure the repository in the **`/etc/yum.repos.d/errata.repo`** file. It should access `http://content.example.com/rhel8.0/x86_64/rhcsa-practice/errata`. Do not check GPG signatures.
2. On `serverb`, install new package *xsane-gimp* and the *Apache HTTP Server* module from the `2.4` stream and the `common` profile.
3. For security reasons, `serverb` should not be able to send anything to print. Achieve this by removing the *cups* package. Exit from the `root` account.
4. The start script downloads the *rhcsa-script-1.0.0-1.noarch.rpm* package in the **`/home/student`** directory on `serverb`.

   Confirm that the package *rhcsa-script-1.0.0-1.noarch.rpm* is available on `serverb`. Install the package. You will need to gain superuser privileges to install the package. Verify that the package is installed. Exit from `serverb`.

Evaluation

On `workstation`, run the **`lab software-review grade`** script to confirm success on this lab.

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

Finish

On `workstation`, run the **`lab software-review finish`** script to complete this exercise. This script removes the repository and packages created during this exercise.

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

This concludes the lab.

SOLUTION

INSTALLING AND UPDATING SOFTWARE PACKAGES

PERFORMANCE CHECKLIST

In this lab, you will manage software repositories and module streams, and install and upgrade packages from those repositories and streams.

OUTCOMES

You should be able to:

- Manage software repositories and module streams.
- Install and upgrade packages from repositories and streams.
- Install an RPM package.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab software-review start`** command. This script ensures that `serverb` is available. It also downloads any packages required for the lab exercise.

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

1. On `serverb` configure a software repository to obtain updates. Name the repository as `errata` and configure the repository in the **`/etc/yum.repos.d/errata.repo`** file. It should access `http://content.example.com/rhel8.0/x86_64/rhcsa-practice/errata`. Do not check GPG signatures.

   1.1. From `workstation` use the **ssh** command to log in to `serverb` as the `student` user.

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

   1.2. Use the **su -** command to switch to the `root` user. The password is `redhat`.

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

   1.3. Create the file **`/etc/yum.repos.d/errata.repo`** with the following content:

```
[errata]
name=Red Hat Updates
baseurl=http://content.example.com/rhel8.0/x86_64/rhcsa-practice/errata
enabled=1
gpgcheck=0
```

2. On `serverb`, install new package *xsane-gimp* and the *Apache HTTP Server* module from the `2.4` stream and the `common` profile.

   2.1. Use the **`yum list`** command to list the available packages for *xsane-gimp*.

```
[root@serverb ~]# yum list xsane-gimp
Last metadata expiration check: 0:24:30 ago on Thu 07 Mar 2019 03:50:55
 PM CET.
Available Packages
xsane-gimp.x86_64     0.999-30.el8    rhel-8.0-for-x86_64-appstream-
rpms
```

   2.2. Install the latest version of the *xsane-gimp* package using the **`yum install`** command.

```
[root@serverb ~]# yum install xsane-gimp
...output omitted...
Install  57 Packages

Total download size: 51 M
Installed size: 205 M
Is this ok [y/N]: y
...output omitted...
Complete!
[root@serverb ~]#
```

   2.3. List available modules and streams. Look for the *httpd* module. Use the **`yum install`** command to install the *httpd* module with the `2.4` stream and the `common` profile.

```
[student@serverb ~]$ yum module list
Name        Stream      Profiles                      Summary
...output omitted...
httpd       2.4 [d]     common [d], devel, minimal    Apache HTTP Server
...output omitted...
[root@serverb ~]# yum module install httpd:2.4/common
Install  10 Packages

Total download size: 2.1 M
Installed size: 5.7 M
Is this ok [y/N]: y
...output omitted...
Complete!
[root@serverb ~]#
```

3. For security reasons, `serverb` should not be able to send anything to print. Achieve this by removing the *cups* package. Exit from the `root` account.

   3.1. Use the **`yum list`** command to list the installed *cups* package.

```
[root@serverb ~]# yum list cups
Installed Packages
cups.x86_64          1:2.2.6-25.el8          @rhel-8.0-for-x86_64-
appstream-rpms
[root@serverb ~]#
```

   3.2. Use the **`yum remove`** command to remove the *cups* package.

```
[root@serverb ~]# yum remove cups.x86_64
...output omitted...
Remove  9 Packages

Freed space: 11 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

   3.3. Exit from the `root` account.

```
[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 `serverb`.

   Confirm that the package *rhcsa-script-1.0.0-1.noarch.rpm* is available on `serverb`. Install the package. You will need to gain superuser privileges to install the package. Verify that the package is installed. Exit from `serverb`.

   4.1. Use the **`rpm`** command to confirm that the *rhcsa-script-1.0.0-1.noarch.rpm* package is available on `serverb` by viewing the package information.

```
[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        : 1056
License     : GPL
Signature   : (none)
Source RPM  : rhcsa-script-1.0.0-1.src.rpm
Build Date  : Wed 06 Mar 2019 11:29:46 AM CET
Build Host  : foundation0.ilt.example.com
Relocations : (not relocatable)
Packager    : Snehangshu Karmakar
URL         : http://example.com
```

```
Summary     : RHCSA Practice Script
Description :
A RHCSA practice script.
The package changes the motd.
```

4.2. Use the **sudo yum localinstall** command to install the *rhcsa-script-1.0.0-1.noarch.rpm* package. The password is student.

```
[student@serverb ~]$ sudo yum localinstall rhcsa-
script-1.0.0-1.noarch.rpm
[sudo] password for student: student
Last metadata expiration check: 1:31:22 ago on Thu 07 Mar 2019 03:50:55
 PM CET.
Dependencies resolved.
=================================================================
 Package             Arch    Version    Repository         Size
=================================================================
 Installing:
  rhcsa-script       noarch  1.0.0-1    @commandline       7.6 k

  Transaction Summary
  =================================================================
  Install  1 Package

  Total size: 7.6 k
  Installed size: 1.0 k
  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. Use the **rpm** command to verify that the package is installed.

```
[student@serverb ~]$ rpm -q rhcsa-script
rhcsa-script-1.0.0-1.noarch
[student@serverb ~]$
```

4.4. Exit from serverb

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

Evaluation

On workstation, run the **lab software-review grade** script to confirm success on this lab.

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

Finish

On workstation, run the **lab software-review finish** script to complete this exercise. This script removes the repository and packages created during this exercise.

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

SUMMARY

In this chapter, you learned:

- Red Hat Subscription Management provides tools to entitle machines to product subscriptions, get updates to software packages, and track information about support contracts and subscriptions used by the systems.
- Software is provided as RPM packages, which make it easy to install, upgrade, and uninstall software from the system.
- The **rpm** command can be used to query a local database to provide information about the contents of installed packages and install downloaded package files.
- **yum** is a powerful command-line tool that can be used to install, update, remove, and query software packages.
- Red Hat Enterprise Linux 8 uses Application Streams to provide a single repository to host multiple versions of an application's packages and its dependencies.

CHAPTER 8

MANAGING BASIC STORAGE

GOAL

Create and manage storage devices, partitions, file systems, and swap spaces from the command line.

OBJECTIVES

- Explain what a block device is, interpret the file names of storage devices, and identify the storage device used by the file system for a particular directory or file.
- Access file systems by attaching them to a directory in the file system hierarchy.
- Create storage partitions, format them with the file systems, and mount them for use.
- Create and manage swap spaces to supplement physical memory.

SECTIONS

- Identifying File Systems and Devices (and Quiz)
- Mounting and Unmounting File Systems (and Guided Exercise)
- Adding Partitions, File Systems, and Persistent Mounts (and Guided Exercise)
- Managing Swap Space (and Guided Exercise)

LAB

Managing Basic Storage

MOUNTING AND UNMOUNTING FILE SYSTEMS

OBJECTIVES

After completing this section, you should be able to access the contents of file systems by adding and removing file systems from the file system hierarchy.

MOUNTING FILE SYSTEMS MANUALLY

A file system residing on a removable storage device needs to be mounted in order to access it. The **mount** command allows the `root` user to manually mount a file system. The first argument of the **mount** command specifies the file system to mount. The second argument specifies the directory to use as the mount point in the file-system hierarchy.

There are two common ways to specify the file system on a disk partition to the **mount** command:

- With the name of the device file in **/dev** containing the file system.
- With the *UUID* written to the file system, a universally-unique identifier.

Mounting a device is relatively simple. You need to identify the device you want to mount, make sure the mount point exists, and mount the device on the mount point.

Identifying the Block Device

A hot-pluggable storage device, whether a hard disk drive (HDD) or solid-state device (SSD) in a server caddy, or a USB storage device, might be plugged into a different port each time they are attached to a system.

Use the **lsblk** command to list the details of a specified block device or all the available devices.

```
[root@host ~]# lsblk
NAME                    MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda                     253:0    0   12G  0 disk
├─vda1                  253:1    0    1G  0 part /boot
├─vda2                  253:2    0    1G  0 part [SWAP]
└─vda3                  253:3    0   11G  0 part /
vdb                     253:16   0   64G  0 disk
└─vdb1                  253:17   0   64G  0 part
```

If you know that you just added a 64 GB storage device with one partition, then you can guess from the preceding output that **/dev/vdb1** is the partition that you want to mount.

Mounting by Block Device Name

The following example mounts the file system in the **/dev/vdb1** partition on the directory **/mnt/data**.

```
[root@host ~]# mount /dev/vdb1 /mnt/data
```

To mount a file system, the destination directory must already exist. The **/mnt** directory exists by default and is intended for use as a temporary mount point.

You can use **/mnt** directory, or better yet create a subdirectory of **/mnt** to use as a temporary mount point, unless you have a good reason to mount it in a specific location in the file-system hierarchy.

> **IMPORTANT**
> If the directory acting as mount point is not empty, any files copied to that directory before the file system was mounted are not accessible until the file system is unmounted again.

This approach works fine in the short run. However, the order in which the operating system detects disks can change if devices are added to or removed from the system. This will change the device name associated with that storage device. A better approach would be to mount by some characteristic built into the file system.

Mounting by File-system UUID

One stable identifier that is associated with a file system is its UUID, a very long hexadecimal number that acts as a universally-unique identifier. This UUID is part of the file system and remains the same as long as the file system is not recreated.

The **lsblk -fp** command lists the full path of the device, along with the UUIDs and mount points, as well as the type of file system in the partition. If the file system is not mounted, the mount point will be blank.

```
[root@host ~]# lsblk -fp
NAME        FSTYPE LABEL UUID                                   MOUNTPOINT
/dev/vda
├─/dev/vda1 xfs          23ea8803-a396-494a-8e95-1538a53b821c /boot
├─/dev/vda2 swap         cdf61ded-534c-4bd6-b458-cab18b1a72ea [SWAP]
└─/dev/vda3 xfs          44330f15-2f9d-4745-ae2e-20844f22762d /
/dev/vdb
└─/dev/vdb1 xfs          46f543fd-78c9-4526-a857-244811be2d88
```

Mount the file system by the UUID of the file system.

```
[root@host ~]# mount UUID="46f543fd-78c9-4526-a857-244811be2d88" /mnt/data
```

AUTOMATIC MOUNTING OF REMOVABLE STORAGE DEVICES

If you are logged in and using the graphical desktop environment, it will automatically mount any removable storage media when it is inserted.

The removable storage device is mounted at **/run/media/*USERNAME*/*LABEL*** where *USERNAME* is the name of the user logged into the graphical environment and *LABEL* is an identifier, often the name given to the file system when it was created if one is available.

Before removing the device, you should unmount it manually.

UNMOUNTING FILE SYSTEMS

The shutdown and reboot procedures unmount all file systems automatically. As part of this process, any file system data cached in memory is flushed to the storage device thus ensuring that the file system suffers no data corruption.

WARNING

File system data is often cached in memory. Therefore, in order to avoid corrupting data on the disk, it is essential that you unmount removable drives before unplugging them. The unmount procedure synchronizes data before releasing the drive, ensuring data integrity.

To unmount a file system, the **umount** command expects the mount point as an argument.

```
[root@host ~]# umount /mnt/data
```

Unmounting is not possible if the mounted file system is in use. For the **umount** command to succeed, all processes needs to stop accessing data under the mount point.

In the example below, the **umount** fails because the file system is in use (the shell is using **/mnt/data** as its current working directory), generating 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 process accessing them in the provided directory. It is useful to identify which processes currently prevent the file system from successful 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
```

Once the processes are identified, an action can be taken, such as waiting for the process to complete or sending a SIGTERM or SIGKILL signal to the process. In this case, it is sufficient to change the current working directory to a directory outside the mount point.

```
[root@host data]# cd
[root@host ~]# umount /mnt/data
```

NOTE

A common reason for file systems to fail to unmount is that a Bash shell is using the mount point or a subdirectory as a current working directory. Use the **cd** command to change out of the file system to resolve this problem.

REFERENCES

`lsblk`(8), `mount`(8), `umount`(8), and `lsof`(8) man pages

GUIDED EXERCISE

MOUNTING AND UNMOUNTING FILE SYSTEMS

In this exercise, you will practice mounting and unmounting file systems.

OUTCOMES

You should be able to identify and mount a new file system at a specified mount point, then unmount it.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab fs-mount start`** command. The command runs a start script that determines if the host, `servera`, is reachable on the network. The script also creates a partition on the second disk attached to `servera`.

```
[student@workstation ~]$ lab fs-mount start
```

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

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

2. A new partition with a file system has been added to the second disk (**/dev/vdb**) on `servera`. Mount the newly available partition by UUID at the newly created mount point **/mnt/newspace**.

   2.1. Use the **su -** command to switch to `root`, as the `root` user can only manually mount a device.

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

   2.2. Create the **/mnt/newspace** directory.

```
[root@servera ~]# mkdir /mnt/newspace
```

   2.3. Use the **lsblk** command with the **-fp** option to discover the UUID of the device, **/dev/vdb1**.

```
[root@servera ~]# lsblk -fp /dev/vdb
NAME        FSTYPE LABEL UUID
 MOUNTPOINT
/dev/vdb
└─/dev/vdb1 xfs          a04c511a-b805-4ec2-981f-42d190fc9a65
```

2.4. Mount the file system by using UUID on the **/mnt/newspace** directory. Replace the UUID with that of the **/dev/vdb1** disk from the previous command output.

```
[root@servera ~]# mount UUID="a04c511a-b805-4ec2-981f-42d190fc9a65" /
mnt/newspace
```

2.5. Verify that the **/dev/vdb1** device is mounted on the **/mnt/newspace** directory.

```
[root@servera ~]# lsblk -fp /dev/vdb
NAME        FSTYPE LABEL UUID
 MOUNTPOINT
/dev/vdb
└─/dev/vdb1 xfs          a04c511a-b805-4ec2-981f-42d190fc9a65 /mnt/
newspace
```

3. Change to the **/mnt/newspace** directory and create a new directory, **/mnt/newspace/newdir**, with an empty file, **/mnt/newspace/newdir/newfile**.

3.1. Change to the **/mnt/newspace** directory.

```
[root@servera ~]# cd /mnt/newspace
```

3.2. Create a new directory, **/mnt/newspace/newdir**.

```
[root@servera newspace]# mkdir newdir
```

3.3. Create a new empty file, **/mnt/newspace/newdir/newfile**.

```
[root@servera newspace]# touch newdir/newfile
```

4. Unmount the file system mounted on the **/mnt/newspace** directory.

   4.1. Use the **umount** command to unmount **/mnt/newspace** while the current directory on the shell is still **/mnt/newspace**. The **umount** command fails to unmount the device.

```
[root@servera newspace]# umount /mnt/newspace
umount: /mnt/newspace: target is busy.
```

   4.2. Change the current directory on the shell to **/root**.

```
[root@servera newspace]# cd
[root@servera ~]#
```

   4.3. Now, successfully unmount **/mnt/newspace**.

```
[root@servera ~]# umount /mnt/newspace
```

5. Exit from servera.

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

Finish

On workstation, run the **lab fs-mount finish** script to complete this exercise.

```
[student@workstation ~]$ lab fs-mount finish
```

This concludes the guided exercise.

ADDING PARTITIONS, FILE SYSTEMS, AND PERSISTENT MOUNTS

OBJECTIVES

After completing this section, you should be able to create storage partitions, format them with file systems, and mount them for use.

PARTITIONING A DISK

Disk partitioning allows system administrators to divide a hard drive into multiple logical storage units, referred to as partitions. By separating a disk into partitions, system administrators can use different partitions to perform different functions.

For example, disk partitioning is necessary or beneficial in these situations:

- Limit available space to applications or users.
- Separate operating system and program files from user files.
- Create a separate area for memory swapping.
- Limit disk space use to improve the performance of diagnostic tools and backup imaging.

MBR Partitioning Scheme

Since 1982, the *Master Boot Record (MBR)* partitioning scheme has dictated how disks are partitioned on systems running BIOS firmware. This scheme supports a maximum of four primary partitions. On Linux systems, with the use of extended and logical partitions, administrators can create a maximum of 15 partitions. Because partition size data is stored as 32-bit values, disks partitioned with the MBR scheme have a maximum disk and partition size of 2 TiB.

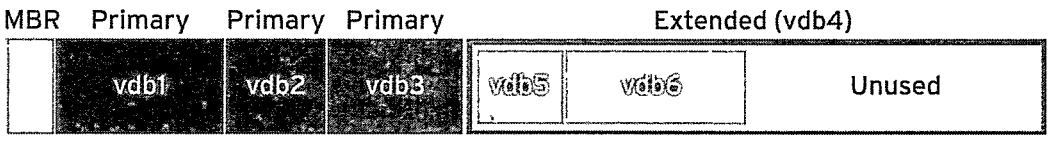


Figure 8.1: MBR Partitioning of the /dev/vdb storage device

Because physical disks are getting larger, and SAN-based volumes even larger than that, the 2 TiB disk and partition size limit of the MBR partitioning scheme is no longer a theoretical limit, but rather a real-world problem that system administrators encounter more and more frequently in production environments. As a result, the legacy MBR scheme is in the process of being superseded by the new *GUID Partition Table (GPT)* for disk partitioning.

GPT Partitioning Scheme

For systems running *Unified Extensible Firmware Interface (UEFI)* firmware, GPT is the standard for laying out partition tables on physical hard disks. GPT is part of the UEFI standard and addresses many of the limitations that the old MBR-based scheme imposes.

A GPT provides a maximum of 128 partitions. Unlike an MBR, which uses 32 bits for storing logical block addresses and size information, a GPT allocates 64 bits for logical block addresses. This

allows a GPT to accommodate partitions and disks of up to eight zebibytes (ZiB) or eight billion tebibytes.

In addition to addressing the limitations of the MBR partitioning scheme, a GPT also offers some additional features and benefits. A GPT uses a *globally unique identifier (GUID)* to identify each disk and partition. In contrast to an MBR, which has a single point of failure, a GPT offers redundancy of its partition table information. The primary GPT resides at the head of the disk, while a backup copy, the secondary GPT, is housed at the end of the disk. A GPT uses a checksum to detect errors and corruptions in the GPT header and partition table.

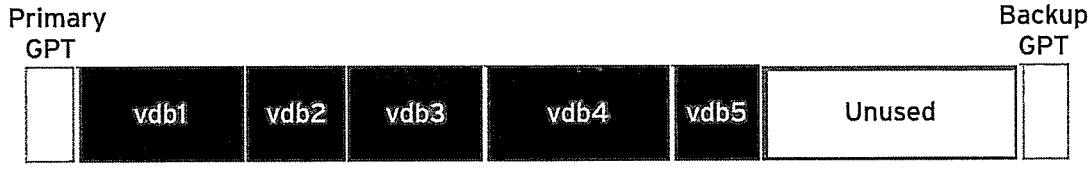


Figure 8.2: GPT Partitioning of the /dev/vdb storage device

MANAGING PARTITIONS WITH PARTED

Partition editors are programs which allow administrators to make changes to a disk's partitions, such as creating partitions, deleting partitions, and changing partition types. To perform these operations, administrators can use the Parted partition editor for both the MBR and the GPT partitioning scheme.

The **parted** command takes the device name of the whole disk as the first argument and one or more subcommands. The following example uses the **print** subcommand to display the partition table on the **/dev/vda** disk.

```
[root@host ~]# parted /dev/vda print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  10.7GB  10.7GB  primary  xfs          boot
 2      10.7GB  53.7GB  42.9GB  primary  xfs
```

If you do not provide a subcommand, **parted** opens an interactive session for issuing commands.

```
[root@host ~]# parted /dev/vda
GNU Parted 3.2
Using /dev/vda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  10.7GB  10.7GB  primary  xfs          boot
```

```
 2      10.7GB  53.7GB  42.9GB  primary  xfs

(parted) quit
```

By default, **parted** displays all the sizes in powers of 10 (KB, MB, GB). You can change that default with the **unit** subcommand which accepts the following parameters:

- **s** for sector
- **B** for byte
- **MiB**, **GiB**, or **TiB** (powers of 2)
- **MB**, **GB**, or **TB** (powers of 10)

```
[root@host ~]# parted /dev/vda unit s print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 104857600s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start      End         Size       Type     File system  Flags
 1      2048s      20971486s   20969439s  primary  xfs          boot
 2      20971520s  104857535s  83886016s  primary  xfs
```

As shown in the example above, you can also specify multiple subcommands (here, **unit** and **print**) on the same line.

Writing the Partition Table on a New Disk

To partition a new drive, you first have to write a disk label to it. The disk label indicates which partitioning scheme to use.

> **NOTE**
> Keep in mind that **parted** makes the changes immediately. A mistake with **parted** could definitely lead to data loss.

As the `root` user, use the following command to write an MBR disk label to a disk.

```
[root@host ~]# parted /dev/vdb mklabel msdos
```

To write a GPT disk label, use the following command.

```
[root@host ~]# parted /dev/vdb mklabel gpt
```

> **WARNING**
> The **mklabel** subcommand wipes the existing partition table. Only use **mklabel** when the intent is to reuse the disk without regard to the existing data. If a new label changes the partition boundaries, all data in existing file systems will become inaccessible.

Creating MBR Partitions

Creating an MBR disk partition involves several steps:

1. Specify the disk device to create the partition on.

   As the `root` user, execute the **`parted`** command and specify the disk device name as an argument. This starts the **`parted`** command in interactive mode and displays a command prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

2. Use the **`mkpart`** subcommand to create a new primary or extended partition.

```
(parted) mkpart
Partition type?  primary/extended? primary
```

> **NOTE**
> For situations where you need more than four partitions on an MBR-partitioned disk, create three primary partitions and one extended partition. This extended partition serves as a container within which you can create multiple logical partitions.

3. Indicate the file-system type that you want to create on the partition, such as **`xfs`** or **`ext4`**. This does not create the file system on the partition; it is only an indication of the partition type.

```
File system type?  [ext2]? xfs
```

   To get the list of the supported file-system types, use the following command:

```
[root@host ~]# parted /dev/vdb help mkpart
  mkpart PART-TYPE [FS-TYPE] START END     make a partition

    PART-TYPE is one of: primary, logical, extended
    FS-TYPE is one of: btrfs, nilfs2, ext4, ext3, ext2, fat32, fat16, hfsx,
    hfs+, hfs, jfs, swsusp, linux-swap(v1), linux-swap(v0), ntfs, reiserfs,
    hp-ufs, sun-ufs, xfs, apfs2, apfs1, asfs, amufs5, amufs4, amufs3,
    amufs2, amufs1, amufs0, amufs, affs7, affs6, affs5, affs4, affs3, affs2,
    affs1, affs0, linux-swap, linux-swap(new), linux-swap(old)
    START and END are disk locations, such as 4GB or 10%.  Negative values
    count from the end of the disk.  For example, -1s specifies exactly the
    last sector.

    'mkpart' makes a partition without creating a new file system on the
    partition.  FS-TYPE may be specified to set an appropriate partition
    ID.
```

4. Specify the sector on the disk that the new partition starts on.

```
Start? 2048s
```

Notice the **s** suffix to provide the value in sectors. You can also use the **MiB**, **GiB**, **TiB**, **MB**, **GB**, or **TB** suffixes. If you do not provide a suffix, **MB** is the default. **parted** may round the value you provide to satisfy disk constraints.

When **parted** starts, it retrieves the disk topology from the device. For example, the disk physical block size is typically a parameter that **parted** collects. With that information, **parted** ensures that the start position you provide correctly aligns the partition with the disk structure. Correct partition alignment is important for optimal performance. If the start position results in a misaligned partition, **parted** displays a warning. With most disks, a start sector that is a multiple of 2048 is a safe assumption.

5. Specify the disk sector where the new partition should end.

```
End? 1000MB
```

With **parted**, you cannot directly provide the size of your partition, but you can quickly compute it with the following formula:

```
Size = End - Start
```

As soon as you provide the end position, **parted** updates the partition table on the disk with the new partition details.

6. Exit **parted**.

```
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

7. Run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the associated device file under the **/dev** directory. It only returns when it is done.

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

As an alternative to the interactive mode, you can also create the partition as follows:

```
[root@host ~]# parted /dev/vdb mkpart primary xfs 2048s 1000MB
```

Creating GPT Partitions

The GTP scheme also uses the **parted** command to create new partitions:

1. Specify the disk device to create the partition on.

As the `root` user, execute the **parted** command with the disk device as the only argument to start **parted** in interactive mode with a command prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

2. Use the **mkpart** subcommand to start creating the new partition.

   With the GPT scheme, each partition is given a name.

```
(parted) mkpart
Partition name?  []? usersdata
```

3. Indicate the file system type that you want to create on the partition, such as **xfs** or **ext4**. This does not create the file system on the partition; it is only an indication of the partition type.

```
File system type?  [ext2]? xfs
```

4. Specify the sector on the disk that the new partition starts on.

```
Start? 2048s
```

5. Specify the disk sector where the new partition should end.

```
End? 1000MB
```

   As soon as you provide the end position, **parted** updates the partition table on the disk with the new partition details.

6. Exit **parted**.

```
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

7. Run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the associated device file under the **/dev** directory. It only returns when it is done.

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

As an alternative to the interactive mode, you can also create the partition as follows:

```
[root@host ~]# parted /dev/vdb mkpart usersdata xfs 2048s 1000MB
```

Deleting Partitions

The following steps apply for both the MBR and GPT partitioning schemes.

1. Specify the disk that contains the partition to be removed.

   As the `root` user, execute the **`parted`** command with the disk device as the only argument to start **`parted`** in interactive mode with a command prompt.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

2. Identify the partition number of the partition to delete.

```
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size   File system  Name       Flags
 1      1049kB  1000MB  999MB  xfs          usersdata
```

3. Delete the partition.

```
(parted) rm 1
```

   The **`rm`** subcommand immediately deletes the partition from the partition table on the disk.

4. Exit **`parted`**.

```
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

CREATING FILE SYSTEMS

After the creation of a block device, the next step is to add a file system to it. Red Hat Enterprise Linux supports many different file system types, but two common ones are XFS and ext4. Anaconda, the installer for Red Hat Enterprise Linux, uses XFS by default.

As `root`, use the **`mkfs.xfs`** command to apply an XFS file system to a block device. For ext4, use **`mkfs.ext4`**.

```
[root@host ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=60992 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
```

```
         =                       reflink=1
data     =                       bsize=4096   blocks=243968, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1566, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

MOUNTING FILE SYSTEMS

After you have added the file system, the last step is to mount the file system to a directory in the directory structure. When you mount a file system onto the directory hierarchy, user-space utilities can access or write files on the device.

Manually Mounting File Systems

Administrators use the **mount** command to manually attach the device onto a directory location, or mount point. The **mount** command expects the device, the mount point, and optionally file system options as arguments. The file-system options customize the behavior of the file system.

```
[root@host ~]# mount /dev/vdb1 /mnt
```

You also use the **mount** command to view currently mounted file systems, the mount points, and the options.

```
[root@host ~]# mount | grep vdb1
/dev/vdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

Persistently Mounting File Systems

Manually mounting a file system is a good way to verify that a formatted device is accessible and working as expected. However, when the server reboots, the system does not automatically mount the file system onto the directory tree again; the data is intact on the file system, but users cannot access it.

To make sure that the system automatically mounts the file system at system boot, add an entry to the **/etc/fstab** file. This configuration file lists the file systems to mount at system boot.

/etc/fstab is a white-space-delimited file with six fields per line.

```
[root@host ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Wed Feb 13 16:39:59 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
```

```
#
UUID=a8063676-44dd-409a-b584-68be2c9f5570   /        xfs   defaults   0 0
UUID=7a20315d-ed8b-4e75-a5b6-24ff9e1f9838   /dbdata  xfs   defaults   0 0
```

When you add or remove an entry in the **/etc/fstab** file, run the **systemctl daemon-reload** command, or reboot the server, for `systemd` to register the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

The *first field* specifies the device. This example uses the UUID to specify the device. File systems create and store the UUID in their super block at creation time. Alternatively, you could use the device file, such as **/dev/vdb1**.

NOTE

Using the UUID is preferable because block device identifiers can change in certain scenarios, such as a cloud provider changing the underlying storage layer of a virtual machine, or the disks being detected in a different order with each system boot. The block device file name may change, but the UUID remains constant in the file system's super block.

Use the **lsblk --fs** command to scan the block devices connected to a machine and retrieve the file system UUIDs.

```
[root@host ~]# lsblk --fs
NAME   FSTYPE LABEL UUID                                   MOUNTPOINT
sr0
vda
└vda1 xfs          a8063676-44dd-409a-b584-68be2c9f5570 /
vdb
└vdb1 xfs          7a20315d-ed8b-4e75-a5b6-24ff9e1f9838 /dbdata
```

The *second field* is the directory mount point, from which the block device will be accessible in the directory structure. The mount point must exist; if not, create it with the **mkdir** command.

The *third field* contains the file-system type, such as **xfs** or **ext4**.

The *fourth field* is the comma-separated list of options to apply to the device. **defaults** is a set of commonly used options. The **mount**(8) man page documents the other available options.

The *fifth field* is used by the **dump** command to back up the device. Other backup applications do not usually use this field.

The *last field*, the **fsck** order field, determines if the **fsck** command should be run at system boot to verify that the file systems are clean. The value in this field indicates the order in which **fsck** should run. For XFS file systems, set this field to **0** because XFS does not use **fsck** to check its file-system status. For ext4 file systems, set it to **1** for the root file system and **2** for the other ext4 file systems. This way, **fsck** processes the root file system first and then checks file systems on separate disks concurrently, and file systems on the same disk in sequence.

NOTE

Having an incorrect entry in **/etc/fstab** may render the machine non-bootable. Administrators should verify that the entry is valid by unmounting the new file system and using **mount /mountpoint**, which reads **/etc/fstab**, to remount the file system. If the **mount** command returns an error, correct it before rebooting the machine.

As an alternative, you can use the **findmnt --verify** command to control the **/etc/fstab** file.

REFERENCES

info parted (*GNU Parted User Manual*)

parted(8), **mkfs**(8), **mount**(8), **lsblk**(8), and **fstab**(5) man pages

For more information, refer to the *Configuring and managing file systems* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_file_systems/

GUIDED EXERCISE

ADDING PARTITIONS, FILE SYSTEMS, AND PERSISTENT MOUNTS

In this exercise, you will create a partition on a new storage device, format it with an XFS file system, configure it to be mounted at boot, and mount it for use.

OUTCOMES

You should be able to use **parted**, **mkfs.xfs**, and other commands to create a partition on a new disk, format it, and persistently mount it.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **lab storage-partitions start** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also prepares the second disk on `servera` for the exercise.

```
[student@workstation ~]$ lab storage-partitions start
```

1. Use the **ssh** command to log in to `servera` 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 ~]$
```

2. Use the **sudo -i** command to switch to the `root` user. If prompted, use `student` as the password.

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

3. Use **parted** to create a new disk label of type **msdos** on the **/dev/vdb** disk to prepare that new disk for the MBR partitioning scheme.

```
[root@servera ~]# parted /dev/vdb mklabel msdos
Information: You may need to update /etc/fstab.
```

4. Add a new primary partition that is 1 GB in size. For proper alignment, start the partition at the sector 2048. Set the partition file system type to XFS.

4.1. Use **parted** interactive mode to help you create the partition.

```
[root@servera ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mkpart
Partition type?  primary/extended? primary
File system type?  [ext2]? xfs
Start? 2048s
End? 1001MB
(parted) quit
Information: You may need to update /etc/fstab.
```

Because the partition starts at the sector 2048, the previous command sets the end position to 1001MB to get a partition size of 1000MB (1 GB).

As an alternative, you can perform the same operation with the following noninteractive command: **parted /dev/vdb mkpart primary xfs 2048s 1001MB**

4.2. Verify your work by listing the partitions on **/dev/vdb**.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  1001MB  1000MB  primary
```

4.3. Run the **udevadm settle** command. This command waits for the system to register the new partition and returns when it is done.

```
[root@servera ~]# udevadm settle
[root@servera ~]#
```

5. Format the new partition with the XFS file system.

```
[root@servera ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=61056 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=244224, imaxpct=25
         =                       sunit=0      swidth=0 blks
```

```
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1566, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

6. Configure the new file system to mount at **/archive** persistently.

6.1. Use **mkdir** to create the **/archive** directory mount point.

```
[root@servera ~]# mkdir /archive
[root@servera ~]#
```

6.2. Use the **lsblk** command with the **--fs** option to discover the UUID of the **/dev/vdb1** device.

```
[root@servera ~]# lsblk --fs /dev/vdb
NAME    FSTYPE LABEL UUID                                   MOUNTPOINT
vdb
└─vdb1 xfs          e3db1abe-6d96-4faa-a213-b96a6f85dcc1
```

The UUID in the previous output is probably different on your system.

6.3. Add an entry to **/etc/fstab**. In the following command, replace the UUID with the one you discovered from the previous step.

```
[root@servera ~]# vim /etc/fstab
...output omitted...
UUID=e3db1abe-6d96-4faa-a213-b96a6f85dcc1   /archive   xfs   defaults
  0   0
```

6.4. Update systemd for the system to register the new **/etc/fstab** configuration.

```
[root@servera ~]# systemctl daemon-reload
[root@servera ~]#
```

6.5. Execute the **mount /archive** command to mount the new file system using the new entry added to **/etc/fstab**.

```
[root@servera ~]# mount /archive
[root@servera ~]#
```

6.6. Verify that the new file system is mounted at **/archive**.

```
[root@servera ~]# mount | grep /archive
/dev/vdb1 on /archive type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

7. Reboot servera. After the server has rebooted, log in and verify that **/dev/vdb1** is mounted at **/archive**. When done, log off from servera.

7.1. Reboot servera.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

7.2. Wait a few minutes for servera to reboot and log in as the student user.

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

7.3. Verify that **/dev/vdb1** is mounted at **/archive**.

```
[student@servera ~]$ mount | grep /archive
/dev/vdb1 on /archive type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

7.4. Log off from servera.

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

Finish

On workstation, run the **lab storage-partitions finish** script to complete this exercise.

```
[student@workstation ~]$ lab storage-partitions finish
```

This concludes the guided exercise.

MANAGING SWAP SPACE

OBJECTIVES

After completing this section, you should be able to create and manage swap spaces to supplement physical memory.

INTRODUCING SWAP SPACE CONCEPTS

A *swap space* is an area of a disk under the control of the Linux kernel memory management subsystem. The kernel uses swap space to supplement the system RAM by holding inactive pages of memory. The combined system RAM plus swap space is called *virtual memory*.

When the memory usage on a system exceeds a defined limit, the kernel searches through RAM looking for idle memory pages assigned to processes. The kernel writes the idle pages to the swap area and reassigns the RAM pages to other processes. If a program requires access to a page on disk, the kernel locates another idle page of memory, writes it to disk, then recalls the needed page from the swap area.

Because swap areas reside on disk, swap is slow when compared with RAM. While it is used to augment system RAM, you should not consider swap space as a sustainable solution for insufficient RAM for your workload.

Sizing the Swap Space

Administrators should size the swap space based on the memory workload on the system. Application vendors sometimes provide recommendations on that subject. The following table provides some guidance based on the total amount of physical memory.

RAM and Swap Space Recommendations

RAM	SWAP SPACE	SWAP SPACE IF ALLOWING FOR HIBERNATION
2 GiB or less	Twice the RAM	Three times the RAM
Between 2 GiB and 8 GiB	Same as RAM	Twice the RAM
Between 8 GiB and 64 GiB	At least 4 GiB	1.5 times the RAM
More than 64 GiB	At least 4 GiB	Hibernation is not recommended

The laptop and desktop hibernation function uses the swap space to save the RAM contents before powering off the system. When you turn the system back on, the kernel restores the RAM contents from the swap space and does not need a complete boot. For those systems, the swap space needs to be greater than the amount of RAM.

The Knowledgebase article in the Reference section at the end of this section gives more guidance on sizing the swap space.

CREATING A SWAP SPACE

To create a swap space, you need to perform the following:

- Create a partition with a file system type of **linux-swap**.
- Place a swap signature on the device.

Creating a Swap Partition

Use **parted** to create a partition of the desired size and set its file system type to **linux-swap**. In the past, tools looked at the partition file system type to determine if the device should be activated; however, that is no longer the case. Even though utilities no longer use the partition file system type, setting that type allows administrators to quickly determine the partition's purpose.

The following example creates a 256 MB partition.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB               data

(parted) mkpart
Partition name?  []? swap1
File system type?  [ext2]? linux-swap
Start? 1001MB
End? 1257MB
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system     Name   Flags
 1      1049kB  1001MB  1000MB                  data
 2      1001MB  1257MB  256MB   linux-swap(v1)  swap1

(parted) quit
Information: You may need to update /etc/fstab.
```

After creating the partition, run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the associated device file in **/dev**. It only returns when it is done.

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

Formatting the Device

The **mkswap** command applies a swap signature to the device. Unlike other formatting utilities, **mkswap** writes a single block of data at the beginning of the device, leaving the rest of the device unformatted so the kernel can use it for storing memory pages.

```
[root@host ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 244 MiB (255848448 bytes)
no label, UUID=39e2667a-9458-42fe-9665-c5c854605881
```

ACTIVATING A SWAP SPACE

You can use the **swapon** command to activate a formatted swap space.

Use **swapon** with the device as a parameter, or use **swapon -a** to activate all the swap spaces listed in the **/etc/fstab** file. Use the **swapon --show** and **free** commands to inspect the available swap spaces.

```
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1873036      134688     1536436       16748      201912     1576044
Swap:             0           0           0
[root@host ~]# swapon /dev/vdb2
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1873036      135044     1536040       16748      201952     1575680
Swap:        249852           0      249852
```

You can deactivate a swap space using the **swapoff** command. If the swap space has pages written to it, **swapoff** tries to move those pages to other active swap spaces or back into memory. If it cannot write data to other places, the **swapoff** command fails with an error, and the swap space stays active.

Activating Swap Space Persistently

To activate a swap space at every boot, place an entry in the **/etc/fstab** file. The example below shows a typical line in **/etc/fstab** based on the swap space created above.

```
UUID=39e2667a-9458-42fe-9665-c5c854605881   swap   swap   defaults   0 0
```

The example uses the UUID as the *first field*. When you format the device, the **mkswap** command displays that UUID. If you lost the output of **mkswap**, use the **lsblk --fs** command. As an alternative, you can also use the device name in the first field.

The *second field* is typically reserved for the mount point. However, for swap devices, which are not accessible through the directory structure, this field takes the placeholder value **swap**.

The *third field* is the file system type. The file system type for swap space is **swap**.

The *fourth field* is for options. The example uses the **defaults** option. The **defaults** option includes the mount option **auto**, which means activate the swap space automatically at system boot.

The final two fields are the **dump** flag and **fsck** order. Swap spaces require neither backing up nor file-system checking and so these fields should be set to zero.

When you add or remove an entry in the **/etc/fstab** file, run the **systemctl daemon-reload** command, or reboot the server, for systemd to register the new configuration.

```
[root@host ~]# systemctl daemon-reload
```

Setting the Swap Space Priority

By default, the system uses swap spaces in series, meaning that the kernel uses the first activated swap space until it is full, then it starts using the second swap space. However, you can define a priority for each swap space to force that order.

To set the priority, use the **pri** option in **/etc/fstab**. The kernel uses the swap space with the highest priority first. The default priority is -2.

The following example shows three swap spaces defined in **/etc/fstab**. The kernel uses the last entry first, with **pri=10**. When that space is full, it uses the second entry, with **pri=4**. Finally, it uses the first entry, which has a default priority of -2.

```
UUID=af30cbb0-3866-466a-825a-58889a49ef33   swap   swap   defaults  0 0
UUID=39e2667a-9458-42fe-9665-c5c854605881   swap   swap   pri=4     0 0
UUID=fbd7fa60-b781-44a8-961b-37ac3ef572bf   swap   swap   pri=10    0 0
```

Use **swapon --show** to display the swap space priorities.

When swap spaces have the same priority, the kernel writes to them in a round-robin fashion.

REFERENCES

mkswap(8), **swapon**(8), **swapoff**(8), **mount**(8), and **parted**(8) man pages

Knowledgebase: What is the recommended swap size for Red Hat platforms?
https://access.redhat.com/solutions/15244

GUIDED EXERCISE

MANAGING SWAP SPACE

In this exercise, you will create and format a partition for use as swap space, format it as swap, and activate it persistently.

OUTCOMES

You should be able to create a partition and a swap space on a disk using the GPT partitioning scheme.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab storage-swap start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also prepares the second disk on `servera` for the exercise.

```
[student@workstation ~]$ lab storage-swap start
```

1. Use the **ssh** command to log in to `servera` 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 ~]$
```

2. Use the **`sudo -i`** command to switch to the `root` user. If prompted, use `student` as the password.

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

3. Use the **`parted`** command to inspect the **`/dev/vdb`** disk.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB               data
```

Notice that the disk already has a partition table and uses the GPT partitioning scheme. Also, a 1 GB partition already exists.

4. Add a new partition that is 500 MB in size for use as swap space. Set the partition type to **linux-swap**.

   4.1. Use **parted** to create the partition. Because the disk uses the GPT partitioning scheme, you need to give a name to the partition. Call it **myswap**.

```
[root@servera ~]# parted /dev/vdb mkpart myswap linux-swap 1001MB
 1501MB
Information: You may need to update /etc/fstab.
```

   Notice in the previous command that the start position, 1001 MB, is the end of the existing first partition. This way **parted** makes sure that the new partition immediately follows the previous one, without any gap.

   Because the partition starts at the 1001 MB position, the command sets the end position to 1501 MB to get a partition size of 500 MB.

   4.2. Verify your work by listing the partitions on **/dev/vdb**.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name   Flags
 1      1049kB  1001MB  1000MB               data
```

   The size of the new partition is not exactly 500 MB. This is because **parted** has to align the partition with the disk layout.

   4.3. Run the **udevadm settle** command. This command waits for the system to register the new partition and returns when it is done.

```
[root@servera ~]# udevadm settle
[root@servera ~]#
```

5. Initialize the newly created partition as swap space.

```
[root@servera ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 476 MiB (499118080 bytes)
no label, UUID=cb7f71ca-ee82-430e-ad4b-7dda12632328
```

6. Enable the newly created swap space.

6.1. Use the **swapon --show** command to show that creating and initializing swap space does not yet enable it for use.

```
[root@servera ~]# swapon --show
[root@servera ~]#
```

6.2. Enable the newly created swap space.

```
[root@servera ~]# swapon /dev/vdb2
[root@servera ~]#
```

6.3. Verify that the newly created swap space is now available.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

6.4. Disable the swap space.

```
[root@servera ~]# swapoff /dev/vdb2
[root@servera ~]#
```

6.5. Confirm that the swap space is disabled.

```
[root@servera ~]# swapon --show
[root@servera ~]#
```

7. Configure the new swap space to be enabled at system boot.

7.1. Use the **lsblk** command with the **--fs** option to discover the UUID of the **/dev/vdb2** device.

```
[root@servera ~]# lsblk --fs /dev/vdb2
NAME FSTYPE LABEL UUID                                   MOUNTPOINT
vdb2 swap         cb7f71ca-ee82-430e-ad4b-7dda12632328
```

The UUID in the previous output is probably different on your system.

7.2. Add an entry to **/etc/fstab**. In the following command, replace the UUID with the one you discovered from the previous step.

```
[root@servera ~]# vim /etc/fstab
...output omitted...
UUID=cb7f71ca-ee82-430e-ad4b-7dda12632328   swap   swap   defaults   0
   0
```

7.3. Update **systemd** for the system to register the new **/etc/fstab** configuration.

```
[root@servera ~]# systemctl daemon-reload
[root@servera ~]#
```

7.4. Enable the swap space using the entry just added to **/etc/fstab**.

```
[root@servera ~]# swapon -a
[root@servera ~]#
```

7.5. Verify that the new swap space is enabled.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

8. Reboot servera. After the server has rebooted, log in and verify that the swap space is enabled. When done, log off from servera.

8.1. Reboot servera.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

8.2. Wait a few minutes for servera to reboot and log in as the student user.

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

8.3. Verify that the swap space is enabled.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

8.4. Log off from servera.

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

Finish

On workstation, run the **lab storage-swap finish** script to complete this exercise.

```
[student@workstation ~]$ lab storage-swap finish
```

This concludes the guided exercise.

▷ LAB

MANAGING BASIC STORAGE

PERFORMANCE CHECKLIST

In this lab, you will create several partitions on a new disk, formatting some with file systems and mounting them, and activating others as swap spaces.

OUTCOMES

You should be able to:

- Display and create partitions using the **parted** command.
- Create new file systems on partitions and persistently mount them.
- Create swap spaces and activate them at boot.

BEFORE YOU BEGIN

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

On workstation, run the **lab storage-review start** command. This command runs a start script that determines if the serverb machine is reachable on the network. It also prepares the second disk on serverb for the exercise.

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

1. New disks are available on serverb. On the first new disk, create a 2 GB GPT partition named **backup**. Because it may be difficult to set the exact size, a size between 1.8 GB and 2.2 GB is acceptable. Set the correct file-system type on that partition to host an XFS file system.

   The password for the student user account on serverb is student. This user has full root access through **sudo**.
2. Format the 2 GB partition with an XFS file system and persistently mount it at **/backup**.
3. On the same new disk, create two 512 MB GPT partitions named **swap1** and **swap2**. A size between 460 MB and 564 MB is acceptable. Set the correct file-system type on those partitions to host swap spaces.
4. Initialize the two 512 MiB partitions as swap spaces and configure them to activate at boot. Set the swap space on the **swap2** partition to be preferred over the other.
5. To verify your work, reboot serverb. Confirm that the system automatically mounts the first partition at **/backup**. Also, confirm that the system activates the two swap spaces.

   When done, log off from serverb.

Evaluation

On workstation, run the **lab storage-review grade** script to confirm success on this exercise.

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

Finish

On `workstation`, run the **`lab storage-review finish`** script to complete the lab.

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

This concludes the lab.

▷ SOLUTION

MANAGING BASIC STORAGE

PERFORMANCE CHECKLIST

In this lab, you will create several partitions on a new disk, formatting some with file systems and mounting them, and activating others as swap spaces.

OUTCOMES

You should be able to:

- Display and create partitions using the **parted** command.
- Create new file systems on partitions and persistently mount them.
- Create swap spaces and activate them at boot.

BEFORE YOU BEGIN

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

On workstation, run the **lab storage-review start** command. This command runs a start script that determines if the serverb machine is reachable on the network. It also prepares the second disk on serverb for the exercise.

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

1. New disks are available on serverb. On the first new disk, create a 2 GB GPT partition named **backup**. Because it may be difficult to set the exact size, a size between 1.8 GB and 2.2 GB is acceptable. Set the correct file-system type on that partition to host an XFS file system.

   The password for the student user account on serverb is student. This user has full root access through **sudo**.

   1.1. Use the **ssh** command to log in to serverb as the student user. The systems are configured to use SSH keys for authentication, therefore a password is not required.

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

   1.2. Because creating partitions and file systems requires root access, use the **sudo -i** command to switch to the root user. If prompted, use student as the password.

   ```
   [student@serverb ~]$ sudo -i
   [sudo] password for student: student
   [root@serverb ~]#
   ```

1.3. Use the **lsblk** command to identify the new disks. Those disks should not have any partitions yet.

```
[root@serverb ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0     11:0    1 1024M  0 rom
vda    252:0    0   10G  0 disk
└vda1 252:1    0   10G  0 part /
vdb    252:16   0    5G  0 disk
vdc    252:32   0    5G  0 disk
vdd    252:48   0    5G  0 disk
```

Notice that the first new disk, **vdb**, does not have any partitions.

1.4. Confirm that the disk has no label.

```
[root@serverb ~]# parted /dev/vdb print
Error: /dev/vdb: unrecognised disk label
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

1.5. Use **parted** and the **mklabel** subcommand to define the GPT partitioning scheme.

```
[root@serverb ~]# parted /dev/vdb mklabel gpt
Information: You may need to update /etc/fstab.
```

1.6. Create the 2 GB partition. Name it **backup** and set its type to **xfs**. Start the partition at sector 2048.

```
[root@serverb ~]# parted /dev/vdb mkpart backup xfs 2048s 2GB
Information: You may need to update /etc/fstab.
```

1.7. Confirm the correct creation of the new partition.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name    Flags
 1      1049kB  2000MB  1999MB               backup
```

1.8. Run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the **/dev/vdb1** device file. It only returns when it is done.

```
[root@serverb ~]# udevadm settle
[root@serverb ~]#
```

2. Format the 2 GB partition with an XFS file system and persistently mount it at **/backup**.

   2.1. Use the **mkfs.xfs** command to format the **/dev/vbd1** partition.

```
[root@serverb ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=121984
 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1,
 rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=487936, imaxpct=25
         =                       sunit=0      swidth=0 blks
```

```
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-
count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

2.2. Create the **/backup** mount point.

```
[root@serverb ~]# mkdir /backup
[root@serverb ~]#
```

2.3. Before adding the new file system to **/etc/fstab**, retrieve its UUID.

```
[root@serverb ~]# lsblk --fs /dev/vdb1
NAME FSTYPE LABEL UUID                                   MOUNTPOINT
vdb1 xfs          a3665c6b-4bfb-49b6-a528-74e268b058dd
```

The UUID on your system is probably different.

2.4. Edit **/etc/fstab** and define the new file system.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=a3665c6b-4bfb-49b6-a528-74e268b058dd   /backup   xfs   defaults
 0 0
```

2.5. Force systemd to reread the **/etc/fstab** file.

```
[root@serverb ~]# systemctl daemon-reload
[root@serverb ~]#
```

2.6. Manually mount **/backup** to verify your work. Confirm that the mount is successful.

```
[root@serverb ~]# mount /backup
[root@serverb ~]# mount | grep /backup
/dev/vdb1 on /backup type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

3. On the same new disk, create two 512 MB GPT partitions named **swap1** and **swap2**. A size between 460 MB and 564 MB is acceptable. Set the correct file-system type on those partitions to host swap spaces.

3.1. Retrieve the end position of the first partition by displaying the current partition table on **/dev/vdb**. In the next step, you use that value as the start of the **swap1** partition.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
```

```
Disk Flags:

Number  Start   End     Size    File system  Name    Flags
 1      1049kB  2000MB  1999MB  xfs          backup
```

3.2. Create the first 512 MB partition named **swap1**. Set its type to **linux-swap**. Use the end position of the first partition as the starting point. The end position is 2000 MB + 512 MB = 2512 MB

```
[root@serverb ~]# parted /dev/vdb mkpart swap1 linux-swap 2000MB 2512M
Information: You may need to update /etc/fstab.
```

3.3. Create the second 512 MB partition named **swap2**. Set its type to **linux-swap**. Use the end position of the previous partition as the starting point: **2512M**. The end position is 2512 MB + 512 MB = 3024 MB

```
[root@serverb ~]# parted /dev/vdb mkpart swap2 linux-swap 2512M 3024M
Information: You may need to update /etc/fstab.
```

3.4. Display the partition table to verify your work.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name    Flags
 1      1049kB  2000MB  1999MB  xfs          backup
 2      2000MB  2512MB  513MB                swap1   swap
```

3.5. Run the **udevadm settle** command. This command waits for the system to register the new partitions and to create the device files.

```
[root@serverb ~]# udevadm settle
[root@serverb ~]#
```

4. Initialize the two 512 MiB partitions as swap spaces and configure them to activate at boot. Set the swap space on the **swap2** partition to be preferred over the other.

4.1. Use the **mkswap** command to initialize the swap partitions.

```
[root@serverb ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 489 MiB (512749568 bytes)
no label, UUID=87976166-4697-47b7-86d1-73a02f0fc803
[root@serverb ~]# mkswap /dev/vdb3
Setting up swapspace version 1, size = 488 MiB (511700992 bytes)
no label, UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942
```

Take note of the UUIDs of the two swap spaces. You use that information in the next step. If you cannot see the **mkswap** output anymore, use the **lsblk --fs** command to retrieve the UUIDs.

4.2. Edit **/etc/fstab** and define the new swap spaces. To set the swap space on the **swap2** partition to be preferred over **swap1**, give it a higher priority with the **pri** option.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=a3665c6b-4bfb-49b6-a528-74e268b058dd   /backup xfs   defaults  0 0
UUID=87976166-4697-47b7-86d1-73a02f0fc803   swap    swap  pri=10    0 0
UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942   swap    swap  pri=20    0 0
```

4.3. Force systemd to reread the **/etc/fstab** file.

```
[root@serverb ~]# systemctl daemon-reload
[root@serverb ~]#
```

4.4. Use the **swapon -a** command to activate the new swap spaces. Use the **swapon --show** command to confirm the correct activation of the swap spaces.

```
[root@serverb ~]# swapon -a
[root@serverb ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 489M   0B   10
/dev/vdb3 partition 488M   0B   20
```

5. To verify your work, reboot `serverb`. Confirm that the system automatically mounts the first partition at **/backup**. Also, confirm that the system activates the two swap spaces.

   When done, log off from `serverb`.

   5.1. Reboot `serverb`.

```
[root@serverb ~]# systemctl reboot
[root@serverb ~]#
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

   5.2. Wait a few minutes for `serverb` to reboot and then log in as the `student` user.

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

   5.3. Verify that the system automatically mounts **/dev/vdb1** at **/backup**.

```
[student@serverb ~]$ mount | grep /backup
/dev/vdb1 on /backup type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

   5.4. Use the **swapon --show** command to confirm that the system activates both swap spaces.

```
[student@serverb ~]$ swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 489M   0B   10
/dev/vdb3 partition 488M   0B   20
```

   5.5. Log off from `serverb`.

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

Evaluation

On `workstation`, run the **lab storage-review grade** script to confirm success on this exercise.

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

Finish

On `workstation`, run the **lab storage-review finish** script to complete the lab.

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

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The **mount** command allows the root user to manually mount a file system.
- **parted** can be used to add, modify, and remove partitions on disks with the MBR or the GPT partitioning scheme.
- XFS file systems are created on disk partitions using **mkfs.xfs**.
- To make file system mounts persistent, they must be added to **/etc/fstab**.
- Swap spaces are initialized using the **mkswap** command.

CHAPTER 9

CONTROLLING SERVICES AND THE BOOT PROCESS

GOAL

Control and monitor network services, system daemons and the boot process using `systemd`.

OBJECTIVES

- List system daemons and network services started by the `systemd` service and socket units.
- Control system daemons and network services, using **`systemctl`**.
- Describe the Red Hat Enterprise Linux boot process, set the default target used when booting, and boot a system to a non-default target.
- Log into a system and change the root password when the current root password has been lost.
- Manually repair file system configuration or corruption issues that stop the boot process.

SECTIONS

- Identifying Automatically Started System Processes (and Guided Exercise)
- Controlling System Services (and Guided Exercise)
- Selecting the Boot Target (and Guided Exercise)
- Resetting the Root Password (and Guided Exercise)
- Repairing File System Issues at Boot (and Guided Exercise)

LAB

Controlling Services and the Boot Process

IDENTIFYING AUTOMATICALLY STARTED SYSTEM PROCESSES

OBJECTIVES

After completing this section, you should be able to list system daemons and network services started by `systemd` service and socket units.

INTRODUCTION TO `systemd`

The *`systemd`* daemon manages startup for Linux, including service startup and service management in general. It 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, performing various tasks. Generally, daemons start automatically at boot time and continue to run until shutdown or until they are manually stopped. It is a convention for names of many daemon programs to end in the letter **d**.

A *service* in the `systemd` sense often refers to one or more daemons, but starting or stopping a service may instead make a one-time change to the state of the system, which does not involve leaving a daemon process running afterward (called `oneshot`).

In Red Hat Enterprise Linux, the first process that starts (PID 1) is `systemd`. A few of the features provided by `systemd` include:

- 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 will not attempt to start up until the network is available.
- A method of tracking related processes together by using Linux control groups.

DESCRIBING SERVICE UNITS

`systemd` uses *units* to manage different types of objects. Some common unit types are listed below:

- Service units have a **`.service`** extension and represent system services. This type of unit is used to start frequently accessed daemons, such as a web server.
- Socket units have a **`.socket`** extension and represent inter-process communication (IPC) sockets that `systemd` should monitor. If a client connects to the socket, `systemd` will start a daemon and pass the connection to it. Socket units are used to delay the start of a service at boot time and to start less frequently used services on demand.
- Path units have a **`.path`** extension and are used to delay the activation of a service until a specific file system change occurs. This is commonly used for services which use spool directories such as a printing system.

The **`systemctl`** command is used to manage units. For example, display available unit types with the **`systemctl -t help`** command.

IMPORTANT
When using **systemctl**, you can abbreviate unit names, process tree entries, and unit descriptions.

LISTING SERVICE UNITS

You use the **systemctl** command to explore the current state of the system. For example, the following command lists all currently loaded service units, paginating the output using **less**.

```
[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...
```

The above output limits the type of unit listed to service units with the **--type=service** option. The output has the following columns:

Columns in the systemctl list-units Command Output

UNIT
The service unit name.

LOAD
Whether systemd properly parsed the unit's configuration and loaded the unit into memory.

ACTIVE
The high-level activation state of the unit. This information indicates whether the unit has started successfully or not.

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 **--all** option lists all service units regardless of the activation states. Use the **--state=** option to filter by the values in the **LOAD**, **ACTIVE**, or **SUB** fields.

```
[root@host ~]# systemctl list-units --type=service --all
UNIT                          LOAD      ACTIVE   SUB     DESCRIPTION
  atd.service                 loaded    active   running Job spooling tools
  auditd.service              loaded    active   running Security Auditing ...
  auth-rpcgss-module.service  loaded    inactive dead    Kernel Module ...
  chronyd.service             loaded    active   running NTP client/server
  cpupower.service            loaded    inactive dead    Configure CPU power ...
```

```
  crond.service                loaded    active   running Command Scheduler
  dbus.service                 loaded    active   running D-Bus System Message Bus
● display-manager.service      not-found inactive dead    display-manager.service
```

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.deviceloaded active plugged   Virtio network...
...
-.mount                               loaded active mounted   Root Mount
boot.mount                            loaded active mounted   /boot
...
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
```

The **systemctl list-units** command displays units that the systemd service attempts to parse and load into memory; it does not display installed, but not enabled, services. To see the state of all unit files installed, use the **systemctl list-unit-files** command. For example:

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                                STATE
arp-ethers.service                       disabled
atd.service                              enabled
auditd.service                           enabled
auth-rpcgss-module.service               static
autovt@.service                          enabled
blk-availability.service                 disabled
...output omitted...
```

In the output of the **systemctl list-units-files** command, valid entries for the **STATE** field are **enabled**, **disabled**, **static**, and **masked**.

VIEWING SERVICE STATES

View the status of a specific unit with **systemctl status** ***name.type***. If the unit type is not provided, **systemctl** will show the status of a service unit, if one exists.

```
[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 Thu 2019-02-14 12:07:45 IST; 7h ago
 Main PID: 1073 (sshd)
   CGroup: /system.slice/sshd.service
```

```
             └─1073 /usr/sbin/sshd -D ...

Feb 14 11:51:39 host.example.com systemd[1]: Started OpenSSH server daemon.
Feb 14 11:51:39 host.example.com sshd[1073]: Could not load host key: /et...y
Feb 14 11:51:39 host.example.com sshd[1073]: Server listening on 0.0.0.0 ....
Feb 14 11:51:39 host.example.com sshd[1073]: Server listening on :: port 22.
Feb 14 11:53:21 host.example.com sshd[1270]: error: Could not load host k...y
Feb 14 11:53:22 host.example.com sshd[1270]: Accepted password for root f...2
```

This command displays the current status of the service. The meaning of the fields are:

Service Unit Information

FIELD	DESCRIPTION
Loaded	Whether the service unit is loaded into memory.
Active	Whether the service unit is running and if so, how long it has been running.
Main PID	The main process ID of the service, including the command name.
Status	Additional information about the service.

Several keywords indicating the state of the service can be found in the status output:

Service States in the Output of `systemctl`

KEYWORD	DESCRIPTION
loaded	Unit configuration file has been processed.
active (running)	Running with one or more continuing processes.
active (exited)	Successfully completed a one-time configuration.
active (waiting)	Running but waiting for an event.
inactive	Not running.
enabled	Is started at boot time.
disabled	Is not set to be started at boot time.
static	Cannot be enabled, but may be started by an enabled unit automatically.

NOTE

The **`systemctl status`** ***`NAME`*** command replaces the **`service`** ***`NAME`*** **`status`** command used in Red Hat Enterprise Linux 6 and earlier.

VERIFYING THE STATUS OF A SERVICE

The **systemctl** command provides methods for verifying the specific states of a service. For example, use the following command to verify that the a service unit is currently active (running):

```
[root@host ~]# systemctl is-active sshd.service
active
```

The command returns state of the service unit, which is usually **active** or **inactive**.

Run the following command 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, which is usually **enabled** or **disabled**.

To verify whether the unit failed during startup, run the following command:

```
[root@host ~]# systemctl is-failed sshd.service
active
```

The command either returns **active** if it is properly running or **failed** if an error has occurred during startup. In case the unit was stopped, it returns **unknown** or **inactive**.

To list all the failed units, run the **systemctl --failed --type=service** command.

REFERENCES

systemd(1), systemd.unit(5), systemd.service(5), systemd.socket(5), and systemctl(1) man pages

For more information, refer to the *Managing services with systemd* chapter in the *Red Hat Enterprise Linux 8.0 Configuring basic system settings* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#managing-services-with-systemd_configuring-basic-system-settings

GUIDED EXERCISE

IDENTIFYING AUTOMATICALLY STARTED SYSTEM PROCESSES

In this exercise, you will list installed service units and identify which services are currently enabled and active on a server.

OUTCOMES

You should be able to list installed service units and identify active and enabled services on the system.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab services-identify start`** command. The command runs a start script that determines if the host, `servera`, is reachable on the network.

```
[student@workstation ~]$ lab services-identify start
```

1. Use the **ssh** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, therefore a password is not required to log in to `servera`.

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

2. List all service units installed on `servera`.

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

Press **q** to exit the command.

3. List all socket units, active and inactive, on `servera`.

```
[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...
systemd-udevd-control.socket    loaded active   running   udev Control
 Socket
systemd-udevd-kernel.socket     loaded active   running   udev Kernel Socket

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.

12 loaded units listed.
To show all installed unit files use 'systemctl list-unit-files'.
```

4. Explore the status of the chronyd service. This service is used for network time 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 Wed 2019-02-06 12:46:57 IST; 4h 7min
 ago
     Docs: man:chronyd(8)
           man:chrony.conf(5)
  Process: 684 ExecStartPost=/usr/libexec/chrony-helper update-daemon
 (code=exited, status=0/SUCCESS)
  Process: 673 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited,
 status=0/SUCCESS)
  Main PID: 680 (chronyd)
    Tasks: 1 (limit: 11406)
   Memory: 1.5M
   CGroup: /system.slice/chronyd.service
           └─680 /usr/sbin/chronyd

... jegui.ilt.example.com systemd[1]: Starting NTP client/server...
...output omitted...
```

```
... jegui.ilt.example.com systemd[1]: Started NTP client/server.
... servera.lab.example.com chronyd[680]: Source 172.25.254.254
 offline
... servera.lab.example.com chronyd[680]: Source 172.25.254.254 online
... servera.lab.example.com chronyd[680]: Selected source
 172.25.254.254
```

Press **q** to exit the command.

4.2. Confirm that the listed daemon is running. In the preceding command, the output of the process ID associated with the chronyd service is 680. The process ID might differ on your system.

```
[student@servera ~]$ ps -p 680
  PID TTY          TIME CMD
  680 ?        00:00:00 chronyd
```

5. Explore the status of the sshd service. This service is used 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 if the sshd service is active without displaying all of the status information.

```
[student@servera ~]$ systemctl is-active sshd
active
```

5.3. Display the status of the sshd service.

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled;
 vendor preset: enabled)
   Active: active (running) since Wed 2019-02-06 12:46:58 IST; 4h
 21min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 720 (sshd)
    Tasks: 1 (limit: 11406)
   Memory: 5.8M
   CGroup: /system.slice/sshd.service
   └─720 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,
   chacha20-poly1305@openssh.com,aes256-ctr,
   aes256-cbc,aes128-gcm@openssh.com,aes128-ctr,
   aes128-cbc -oMACs=hmac-sha2-256-etm@openssh.com,hmac-sha>

... jegui.ilt.example.com systemd[1]: Starting OpenSSH server
 daemon...
```

```
... servera.lab.example.com sshd[720]: Server listening on 0.0.0.0
 port 22.
... servera.lab.example.com systemd[1]: Started OpenSSH server daemon.
... servera.lab.example.com sshd[720]: Server listening on :: port 22.
...output omitted...
... servera.lab.example.com sshd[1380]: pam_unix(sshd:session):
 session opened for user student 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
arp-ethers.service           disabled
atd.service                  enabled
auditd.service               enabled
auth-rpcgss-module.service   static
autovt@.service              enabled
blk-availability.service     disabled
chrony-dnssrv@.service       static
chrony-wait.service          disabled
chronyd.service              enabled
```

Press **q** to exit the command.

7. Exit from `servera`.

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

Finish

On `workstation`, run the **`lab services-identify finish`** script to complete this exercise.

```
[student@workstation ~]$ lab services-identify finish
```

This concludes the guided exercise.

CONTROLLING SYSTEM SERVICES

OBJECTIVES

After completing this section, you should be able to control system daemons and network services, using **systemctl**.

STARTING AND STOPPING SERVICES

Services need to be stopped or started manually for a number of reasons: perhaps the service needs to be updated; the configuration file may need to be changed; or a service may need to be uninstalled; or an administrator may manually start an infrequently used service.

To start a service, first verify that it is not running with **systemctl status**. Then, use the **systemctl start** command as the root user (using **sudo** if necessary). The example below shows how to start the sshd.service service:

```
[root@host ~]# systemctl start sshd.service
```

The systemd service looks for **.service** files for service management in commands in the absence of the service type with the service name. Thus the above command can be executed as:

```
[root@host ~]# systemctl start sshd
```

To stop a currently running service, use the **stop** argument with the **systemctl** command. The example below shows how to stop the sshd.service service:

```
[root@host ~]# systemctl stop sshd.service
```

RESTARTING AND RELOADING SERVICES

During a restart of a running service, the service is stopped and then started. On the restart of service, the process ID changes and a new process ID gets associated during the startup. To restart a running service, use the **restart** argument with the **systemctl** command. The example below shows how to restart the sshd.service service:

```
[root@host ~]# systemctl restart sshd.service
```

Some services have the ability to reload their configuration files without requiring a restart. This process is called a *service reload*. Reloading a service does not change the process ID associated with various service processes. To reload a running service, use the **reload** argument with the **systemctl** command. The example below shows how to reload the sshd.service service after configuration changes:

```
[root@host ~]# systemctl reload sshd.service
```

In case you are not sure whether the service has the functionality to reload the configuration file changes, use the **reload-or-restart** argument with the **systemctl** command. The command reloads the configuration changes if the reloading functionality is available. Otherwise, the command restarts the service to implements the new configuration changes:

```
[root@host ~]# systemctl reload-or-restart sshd.service
```

LISTING UNIT DEPENDENCIES

Some services require that other services be running first, creating dependencies on the other services. Other services are not started at boot time but rather only on demand. In both cases, ***systemd*** and **systemctl** start services as needed whether to resolve the dependency or to start an infrequently used service. For example, if the CUPS print service is not running and a file is placed into the print spool directory, then the system will start 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
```

To completely stop printing services on a system, 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...
```

MASKING AND UNMASKING SERVICES

At times, a system may have different services installed that are conflicting with each other. For example, there are multiple methods to manage mail servers (`postfix` and `sendmail`, for example). Masking a service prevents an administrator from accidentally starting a service that conflicts with others. Masking creates a link in the configuration directories to the **/dev/null** file which prevents the service from starting.

```
[root@host ~]# systemctl mask sendmail.service
Created symlink /etc/systemd/system/sendmail.service → /dev/null.
```

```
[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
A disabled service can be started manually or by other unit files but it does not start automatically at boot. A masked service does not start manually or automatically.

ENABLING SERVICES TO START OR STOP AT BOOT

Starting a service on a running system does not guarantee that the service automatically starts when the system reboots. Similarly, stopping a service on a running system does not keep it from starting again when the system reboots. Creating links in the `systemd` configuration directories enables the service to start at boot. The **`systemctl`** commands creates and removes these links.

To start a service at boot, use the **`systemctl enable`** command.

```
[root@root ~]# systemctl enable sshd.service
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service → /usr/
lib/systemd/system/sshd.service.
```

The above command creates a symbolic link from the service unit file, usually in the **`/usr/lib/systemd/system`** directory, to the location on disk where `systemd` looks for files, which is 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, execute both the **`systemctl start`** and **`systemctl enable`** commands.

To disable the service from starting automatically, use the following command, which removes the symbolic link created while enabling a service. Note that disabling a service does not stop the service.

```
[root@host ~]# systemctl disable sshd.service
Removed /etc/systemd/system/multi-user.target.wants/sshd.service.
```

To verify whether the service is enabled or disable, use the **`systemctl is-enabled`** command.

SUMMARY OF `systemctl` COMMANDS

Services can be started and stopped on a running system and enabled or disabled for an automatic start at boot time.

Useful Service Management Commands

TASK	COMMAND
View detailed information about a unit state.	`systemctl status` ***UNIT***
Stop a service on a running system.	`systemctl stop` ***UNIT***
Start a service on a running system.	`systemctl start` ***UNIT***
Restart a service on a running system.	`systemctl restart` ***UNIT***
Reload the configuration file of a running service.	`systemctl reload` ***UNIT***
Completely disable a service from being started, both manually and at boot.	`systemctl mask` ***UNIT***
Make a masked service available.	`systemctl unmask` ***UNIT***
Configure a service to start at boot time.	`systemctl enable` ***UNIT***
Disable a service from starting at boot time.	`systemctl disable` ***UNIT***
List units required and wanted by the specified unit.	`systemctl list-dependencies` ***UNIT***

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 8.0 Configuring basic system settings* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#managing-system-services_managing-services-with-systemd

GUIDED EXERCISE

CONTROLLING SYSTEM SERVICES

In this exercise, you will use **systemctl** to stop, start, restart, reload, enable, and disable a systemd-managed service.

OUTCOMES

You should be able to use the **systemctl** command to control systemd-managed services.

BEFORE YOU BEGIN

Log in as the student user on workstation using student as the password.

From workstation, run the **lab services-control start** command. The command runs a start script that determines whether the host, servera, is reachable on the network. The script also ensures that the sshd and chronyd services are running on servera.

```
[student@workstation ~]$ lab services-control start
```

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 therefore a password is not required.

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

2. Execute the **systemctl restart** and **systemctl reload** commands on the sshd service. Observe the different results of executing these commands.

   2.1. Display the status of the sshd service. Note the process ID of the sshd daemon.

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled;
 vendor preset: enabled)
   Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min
 ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 759 (sshd)
    Tasks: 1 (limit: 11407)
   Memory: 5.9M
```

   Press **q** to exit the command.

   2.2. Restart the sshd service and view the status. The process ID of the daemon must change.

```
[student@servera ~]$ sudo systemctl restart sshd
[sudo] password for student: student
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled;
 vendor preset: enabled)
   Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min
 ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1132 (sshd)
    Tasks: 1 (limit: 11407)
   Memory: 5.9M
```

In the preceding output, notice that the process ID changed from 759 to 1132 (on your system, the numbers likely will be different). Press **q** to exit the command.

2.3. Reload the sshd service and view the status. The process ID of the daemon must not change and connections are not interrupted.

```
[student@servera ~]$ sudo systemctl reload sshd
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled;
 vendor preset: enabled)
   Active: active (running) since Wed 2019-02-06 23:50:42 EST; 9min
 ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1132 (sshd)
    Tasks: 1 (limit: 11407)
   Memory: 5.9M
```

Press **q** to exit the command.

3. Verify that the chronyd service is running.

```
[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 Wed 2019-02-06 23:50:38 EST; 1h 25min ago
  ...output omitted...
```

Press **q** to exit the command.

4. Stop the chronyd service and view the status.

```
[student@servera ~]$ sudo systemctl stop chronyd
[student@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 2019-02-07 01:20:34 EST; 44s ago
  ...output omitted...
... servera.lab.example.com chronyd[710]: System clock wrong by 1.349113
 seconds, adjustment started
... servera.lab.example.com systemd[1]: Stopping NTP client/server...
... servera.lab.example.com systemd[1]: Stopped NTP client/server.
```

Press **q** to exit the command.

5. Determine if the chronyd service is enabled to start at system boot.

```
[student@server ~]$ systemctl is-enabled chronyd
enabled
```

6. Reboot servera, then view the status of the chronyd service.

```
[student@servera ~]$ sudo 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.

```
[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; enabled; vendor
 preset: enabled)
   Active: active (running) since Thu 2019-02-07 01:48:26 EST; 5min ago
   ...output omitted...
```

Press **q** to exit the command.

7. Disable the chronyd service so that it does not start at system boot, then view the status of the service.

```
[student@servera ~]$ sudo systemctl disable chronyd
[sudo] password for student: student
Removed /etc/systemd/system/multi-user.target.wants/chronyd.service.
[student@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 2019-02-07 01:48:26 EST; 5min ago
   ...output omitted...
```

Press **q** to exit the command.

8. Reboot servera, then view the status of the chronyd service.

```
[student@servera ~]$ sudo 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.

```
[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. Exit from servera.

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

Finish

On workstation, run the **lab services-control finish** script to complete this exercise.

```
[student@workstation ~]$ lab services-control finish
```

This concludes the guided exercise.

SELECTING THE BOOT TARGET

OBJECTIVES

After completing this section, you should be able to:

- Describe the Red Hat Enterprise Linux boot process.
- Set the default target used when booting.
- Boot a system to a non-default target.

DESCRIBING THE RED HAT ENTERPRISE LINUX 8 BOOT PROCESS

Modern computer systems are complex combinations of hardware and software. Starting from an undefined, powered-down state to a running system with a login prompt requires a large number of pieces of hardware and software to work together. The following list gives a high-level overview of the tasks involved for a physical **x86_64** system booting Red Hat Enterprise Linux 8. The list for **x86_64** virtual machines is roughly the same, but the hypervisor handles some of the hardware-specific steps in software.

- The machine is powered on. The system firmware, either modern UEFI or older BIOS, runs a *Power On Self Test (POST)* and starts to initialize some of the hardware.

  Configured using the system BIOS or UEFI configuration screens that you typically reach by pressing a specific key combination, such as **F2**, early during the boot process.

- The system firmware searches for a bootable device, either configured in the UEFI boot firmware or by searching for a *Master Boot Record (MBR)* on all disks, in the order configured in the BIOS.

  Configured using the system BIOS or UEFI configuration screens that you typically reach by pressing a specific key combination, such as **F2**, early during the boot process.

- The system firmware reads a boot loader from disk and then passes control of the system to the boot loader. On a Red Hat Enterprise Linux 8 system, the boot loader is the *GRand Unified Bootloader version 2 (GRUB2)*.

  Configured using the **`grub2-install`** command, which installs GRUB2 as the boot loader on the disk.

- GRUB2 loads its configuration from the **`/boot/grub2/grub.cfg`** file and displays a menu where you can select which kernel to boot.

  Configured using the **`/etc/grub.d/`** directory, the **`/etc/default/grub`** file, and the **`grub2-mkconfig`** command to generate the **`/boot/grub2/grub.cfg`** file.

- After you select a kernel, or the timeout expires, the boot loader loads the kernel and *initramfs* from disk and places them in memory. An **`initramfs`** is an archive containing the kernel modules for all the hardware required at boot, initialization scripts, and more. On Red Hat Enterprise Linux 8, the **`initramfs`** contains an entire usable system by itself.

Configured using the **/etc/dracut.conf.d/** directory, the **dracut** command, and the **lsinitrd** command to inspect the **initramfs** file.

- The boot loader hands control over to the kernel, passing in any options specified on the kernel command line in the boot loader, and the location of the **initramfs** in memory.

  Configured using the **/etc/grub.d/** directory, the **/etc/default/grub** file, and the **grub2-mkconfig** command to generate the **/boot/grub2/grub.cfg** file.

- The kernel initializes all hardware for which it can find a driver in the **initramfs**, then executes **/sbin/init** from the **initramfs** as PID 1. On Red Hat Enterprise Linux 8, **/sbin/init** is a link to **systemd**.

  Configured using the kernel **init=** command-line parameter.

- The systemd instance from the **initramfs** executes all units for the initrd.target target. This includes mounting the root file system on disk on to the **/sysroot** directory.

  Configured using **/etc/fstab**

- The kernel switches (pivots) the root file system from **initramfs** to the root file system in **/sysroot**. systemd then re-executes itself using the copy of systemd installed on the disk.

- systemd looks for a default target, either passed in from the kernel command line or configured on the system, then starts (and stops) units to comply with the configuration for that target, solving dependencies between units automatically. In essence, a systemd target is a set of units that the system should activate to reach the desired state. These targets typically start a text-based login or a graphical login screen.

  Configured using **/etc/systemd/system/default.target** and **/etc/systemd/system/**.

REBOOTING AND SHUTTING DOWN

To power off or reboot a running system from the command line, you can use the **systemctl** command.

systemctl poweroff stops all running services, unmounts all file systems (or remounts them read-only when they cannot be unmounted), and then powers down the system.

systemctl reboot stops all running services, unmounts all file systems, and then reboots the system.

You can also use the shorter version of these commands, **poweroff** and **reboot**, which are symbolic links to their **systemctl** equivalents.

> **NOTE**
> **systemctl halt** and **halt** are also available to stop the system, but unlike **poweroff**, these commands do not power off the system; they bring a system down to a point where it is safe to power it off manually.

SELECTING A SYSTEMD TARGET

A systemd target is a set of systemd units that the system should start to reach a desired state. The following table lists the most important targets.

Commonly Used Targets

TARGET	PURPOSE
`graphical.target`	System supports multiple users, graphical- and text-based logins.
`multi-user.target`	System supports multiple users, text-based logins only.
`rescue.target`	**`sulogin`** prompt, basic system initialization completed.
`emergency.target`	**`sulogin`** prompt, **`initramfs`** pivot complete, and system root mounted on **/** read only.

A target can be a part of another target. For example, the `graphical.target` includes `multi-user.target`, which in turn depends on `basic.target` and others. You can view these dependencies with the following command.

```
[user@host ~]$ systemctl list-dependencies graphical.target | grep target
graphical.target
* └─multi-user.target
*   ├─basic.target
*   | ├─paths.target
*   | ├─slices.target
*   | ├─sockets.target
*   | ├─sysinit.target
*   | | ├─cryptsetup.target
*   | | ├─local-fs.target
*   | | └─swap.target
```

To list the available targets, use the following command.

```
[user@host ~]$ systemctl list-units --type=target --all
  UNIT                      LOAD      ACTIVE   SUB    DESCRIPTION
  ------------------------------------------------------------------------------
  basic.target              loaded    active   active Basic System
  cryptsetup.target         loaded    active   active Local Encrypted Volumes
  emergency.target          loaded    inactive dead   Emergency Mode
  getty-pre.target          loaded    inactive dead   Login Prompts (Pre)
  getty.target              loaded    active   active Login Prompts
  graphical.target          loaded    inactive dead   Graphical Interface
```

Selecting a Target at Runtime

On a running system, administrators can switch to a different target using the **`systemctl isolate`** command.

```
[root@host ~]# systemctl isolate multi-user.target
```

Isolating a target stops all services not required by that target (and its dependencies), and starts any required services not yet started.

Not all targets can be isolated. You can only isolate targets that have `AllowIsolate=yes` set in their unit files. For example, you can isolate the graphical target, but not the cryptsetup target.

```
[user@host ~]$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
...output omitted...
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
[user@host ~]$ systemctl cat cryptsetup.target
# /usr/lib/systemd/system/cryptsetup.target
...output omitted...
[Unit]
Description=Local Encrypted Volumes
Documentation=man:systemd.special(7)
```

Setting a Default Target

When the system starts, `systemd` activates the `default.target` target. Normally the default target in **`/etc/systemd/system/`** is a symbolic link to either `graphical.target` or `multi-user.target`. Instead of editing this symbolic link by hand, the **`systemctl`** command provides two subcommands to manage this link: **`get-default`** and **`set-default`**.

```
[root@host ~]# systemctl get-default
multi-user.target
[root@host ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/
graphical.target.
[root@host ~]# systemctl get-default
graphical.target
```

Selecting a Different Target at Boot Time

To select a different target at boot time, append the **`systemd.unit=`*`target`*`.target`** option to the kernel command line from the boot loader.

For example, to boot the system into a rescue shell where you can change the system configuration with almost no services running, append the following option to the kernel command line from the boot loader.

```
systemd.unit=rescue.target
```

This configuration change only affects a single boot, making it a useful tool for troubleshooting the boot process.

To use this method of selecting a different target, use the following procedure:

1. Boot or reboot the system.
2. Interrupt the boot loader menu countdown by pressing any key (except **Enter** which would initiate a normal boot).
3. Move the cursor to the kernel entry that you want to start.
4. Press **e** to edit the current entry.
5. Move the cursor to the line that starts with **linux**. This is the kernel command line.
6. Append **systemd.unit=*target*.target**. For example, **systemd.unit=emergency.target**.
7. Press **Ctrl+x** to boot with these changes.

REFERENCES

info grub2 (*GNU GRUB manual*)

bootup(7), **dracut.bootup**(7), **lsinitrd**(1), **systemd.target**(5), **systemd.special**(7), **sulogin**(8), and **systemctl**(1) man pages

For more information, refer to the *Managing services with systemd* chapter in the *Configuring basic system settings* guide at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/#managing-services-with-systemd

GUIDED EXERCISE

SELECTING THE BOOT TARGET

In this exercise, you will determine the default target into which a system boots, and boot that system into other targets.

OUTCOMES

You should be able to update the system default target and use a temporary target from the boot loader.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab boot-selecting start`** command. This command runs a start script that prepares `workstation` for the exercise.

```
[student@workstation ~]$ lab boot-selecting start
```

1. On `workstation`, open a terminal and confirm that the default target is `graphical.target`.

```
[student@workstation ~]$ systemctl get-default
graphical.target
```

2. On `workstation`, switch to the `multi-user` target manually without rebooting. Use the **sudo** command and if prompted, use `student` as the password.

```
[student@workstation ~]$ sudo systemctl isolate multi-user.target
[sudo] password for student: student
```

3. Access a text-based console. Use the **`Ctrl`**+**`Alt`**+**`F1`** key sequence using the relevant button or menu entry. Log in as `root` using `redhat` as the password.

> **NOTE**
> Reminder: If you are using the terminal through a webpage you can click the Show Keyboard icon under your web browser's url bar and then to the right of the machine's IP address.

```
workstation login: root
Password: redhat
[root@workstation ~]#
```

4. Configure `workstation` to automatically boot into the `multi-user` target, and then reboot `workstation` to verify. When done, change the default `systemd` target back to the `graphical` target.

4.1. Use the **`systemctl set-default`** command to set the default target.

```
[root@workstation ~]# systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/
systemd/system/multi-user.target.
```

4.2. Reboot `workstation`.

```
[root@workstation ~]# systemctl reboot
```

Notice that after reboot the system presents a text-based console and not a graphical login anymore.

4.3. Log in as `root` using `redhat` as the password.

```
workstation login: root
Password: redhat
Last login: Thu Mar 28 14:50:53 on tty1
[root@workstation ~]#
```

4.4. Set the default `systemd` target back to the `graphical` target.

```
[root@workstation ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/
systemd/system/graphical.target.
```

This concludes the first part of the exercise where you practice setting the default `systemd` target.

5. In this second part of the exercise, you are practicing the rescue mode.

   Access the boot loader by rebooting `workstation` again. From within the boot loader menu, boot into the `rescue` target.

   5.1. Initiate the reboot.

   ```
   [root@workstation ~]# systemctl reboot
   ```

   5.2. When the boot loader menu appears, press any key to interrupt the countdown (except **Enter**, which would initiate a normal boot).

   5.3. Use the cursor keys to highlight the default boot loader entry.

   5.4. Press **e** to edit the current entry.

   5.5. Using the cursor keys, navigate to the line that starts with **linux**.

   5.6. Press **End** to move the cursor to the end of the line.

   5.7. Append **systemd.unit=rescue.target** to the end of the line.

   5.8. Press **Ctrl**+**x** to boot using the modified configuration.

   5.9. Log in to rescue mode. The `root` password is `redhat` You may need to hit enter to get a clean prompt.

   ```
   Give root password for maintenance
   (or press Control-D to continue): redhat
   [root@workstation ~]#
   ```

6. Confirm that in rescue mode, the root file system is in read/write mode.

   ```
   [root@workstation ~]# mount
   ...output omitted...
   /dev/vda1 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
   ...output omitted...
   ```

7. Press **Ctrl**+**d** to continue with the boot process.

   The system presents a graphical login. Log in as `student` using `student` as the password.

Finish

On `workstation`, run the **lab boot-selecting finish** script to complete this exercise.

```
[student@workstation ~]$ lab boot-selecting finish
```

This concludes the guided exercise.

RESETTING THE ROOT PASSWORD

OBJECTIVES

After completing this section, you should be able to log in to a system and change the `root` password when the current `root` password has been lost.

RESETTING THE ROOT PASSWORD FROM THE BOOT LOADER

One task that every system administrator should be able to accomplish is resetting a lost `root` password. If the administrator is still logged in, either as an unprivileged user but with full **sudo** access, or as `root`, this task is trivial. When the administrator is not logged in, this task becomes slightly more involved.

Several methods exist to set a new `root` password. A system administrator could, for example, boot the system using a Live CD, mount the root file system from there, and edit **/etc/shadow**. In this section, we explore a method that does not require the use of external media.

NOTE
On Red Hat Enterprise Linux 6 and earlier, administrators can boot the system into runlevel 1 to get a `root` prompt. The closest analogs to runlevel 1 on a Red Hat Enterprise Linux 8 machine are the rescue and emergency targets, both of which require the `root` password to log in.

On Red Hat Enterprise Linux 8, it is possible to have the scripts that run from the **initramfs** pause at certain points, provide a `root` shell, and then continue when that shell exits. This is mostly meant for debugging, but you can also use this method to reset a lost `root` password.

To access that `root` shell, follow these steps:

1. Reboot the system.
2. Interrupt the boot loader countdown by pressing any key, except **Enter**.
3. Move the cursor to the kernel entry to boot.
4. Press **e** to edit the selected entry.
5. Move the cursor to the kernel command line (the line that starts with **linux**).
6. Append **rd.break**. With that option, the system breaks just before the system hands control from the **initramfs** to the actual system.
7. Press **Ctrl**+**x** to boot with the changes.

At this point, the system presents a `root` shell, with the actual root file system on the disk mounted read-only on **/sysroot**. Because troubleshooting often requires modification to the root file system, you need to change the root file system to read/write. The following step shows how the **remount,rw** option to the **mount** command remounts the file system with the new option (**rw**) set.

NOTE

Prebuilt images may place multiple **console=** arguments to the kernel to support a wide array of implementation scenarios. Those **console=** arguments indicate the devices to use for console output. The caveat with **rd.break** is that even though the system sends the kernel messages to all the consoles, the prompt ultimately uses whichever console is given last. If you do not get your prompt, you may want to temporarily reorder the **console=** arguments when you edit the kernel command line from the boot loader.

IMPORTANT

The system has not yet enabled SELinux, so any file you create does not have an SELinux context. Some tools, such as the **passwd** command, first create a temporary file, then move it in place of the file they are intended to edit, effectively creating a new file without an SELinux context. For this reason, when you use the **passwd** command with **rd.break**, the **/etc/shadow** file does not get an SELinux context.

To reset the `root` password from this point, use the following procedure:

1. Remount **/sysroot** as read/write.

```
switch_root:/# mount -o remount,rw /sysroot
```

2. Switch into a **chroot** jail, where **/sysroot** is treated as the root of the file-system tree.

```
switch_root:/# chroot /sysroot
```

3. Set a new `root` password.

```
sh-4.4# passwd root
```

4. Make sure that all unlabeled files, including **/etc/shadow** at this point, get relabeled during boot.

```
sh-4.4# touch /.autorelabel
```

5. Type **exit** twice. The first command exits the **chroot** jail, and the second command exits the **initramfs** debug shell.

At this point, the system continues booting, performs a full SELinux relabel, and then reboots again.

INSPECTING LOGS

Looking at the logs of previously failed boots can be useful. If the system journals are persistent across reboots, you can use the **journalctl** tool to inspect those logs.

Remember that by default, the system journals are kept in the **/run/log/journal** directory, which means the journals are cleared when the system reboots. To store journals in the **/**

var/log/journal directory, which persists across reboots, set the Storage parameter to **persistent** in **/etc/systemd/journald.conf**.

```
[root@host ~]# vim /etc/systemd/journald.conf
...output omitted...
[Journal]
Storage=persistent
...output omitted...
[root@host ~]# systemctl restart systemd-journald.service
```

To inspect the logs of a previous boot, use the **-b** option of **journalctl**. Without any arguments, the **-b** option only displays messages since the last boot. With a negative number as an argument, it displays the logs of previous boots.

```
[root@host ~]# journalctl -b -1 -p err
```

This command shows all messages rated as an error or worse from the previous boot.

REPAIRING SYSTEMD BOOT ISSUES

To troubleshoot service startup issues at boot time, Red Hat Enterprise Linux 8 makes the following tools available.

Enabling the Early Debug Shell

By enabling the debug-shell service with **systemctl enable debug-shell.service**, the system spawns a root shell on **TTY9** (**Ctrl**+**Alt**+**F9**) early during the boot sequence. This shell is automatically logged in as root, so that administrators can debug the system while the operating system is still booting.

WARNING
Do not forget to disable the debug-shell.service service when you are done debugging, because it leaves an unauthenticated root shell open to anyone with local console access.

Using the Emergency and Rescue Targets

By appending either **systemd.unit=rescue.target** or **systemd.unit=emergency.target** to the kernel command line from the boot loader, the system spawns into a rescue or emergency shell instead of starting normally. Both of these shells require the root password.

The emergency target keeps the root file system mounted read-only, while the rescue target waits for sysinit.target to complete, so that more of the system is initialized, such as the logging service or the file systems. The root user at this point can not make changes to /etc/fstab until the drive is remounted in a read write state **mount -o remount,rw /**

Administrators can use these shells to fix any issues that prevent the system from booting normally; for example, a dependency loop between services, or an incorrect entry in **/etc/fstab**. Exiting from these shells continues with the regular boot process.

Identifying Stuck Jobs

During startup, `systemd` spawns a number of jobs. If some of these jobs cannot complete, they block other jobs from running. To inspect the current job list, administrators can use the **`systemctl list-jobs`** command. Any jobs listed as running must complete before the jobs listed as waiting can continue.

REFERENCES

`dracut.cmdline`(7), **`systemd-journald`**(8), **`journald.conf`**(5), **`journalctl`**(1), and **`systemctl`**(1) man pages

GUIDED EXERCISE

RESETTING THE ROOT PASSWORD

In this exercise, you will reset the `root` password on a system.

OUTCOMES

You should be able to reset a lost `root` password.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab boot-resetting start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also resets the `root` password to a random string and sets a higher timeout for the GRUB2 menu.

```
[student@workstation ~]$ lab boot-resetting start
```

1. Reboot `servera`, and interrupt the countdown in the boot-loader menu.

   1.1. Locate the icon for the `servera` console, as appropriate for your classroom environment. Open the console.

   Send a **`Ctrl`**+**`Alt`**+**`Del`** to your system using the relevant button or menu entry.

   1.2. When the boot-loader menu appears, press any key to interrupt the countdown, except **`Enter`**.

2. Edit the default boot-loader entry, in memory, to abort the boot process just after the kernel mounts all the file systems, but before it hands over control to `systemd`.

   2.1. Use the cursor keys to highlight the default boot-loader entry.

   2.2. Press **e** to edit the current entry.

   2.3. Use the cursor keys to navigate to the line that starts with **`linux`**.

   2.4. Press **End** to move the cursor to the end of the line.

   2.5. Append **`rd.break`** to the end of the line.

   2.6. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

3. At the **`switch_root`** prompt, remount the **`/sysroot`** file system read/write, then use **`chroot`** to go into a **`chroot`** jail at **`/sysroot`**.

```
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
```

4. Change the `root` password back to `redhat`.

```
sh-4.4# passwd root
Changing password for user root.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

5. Configure the system to automatically perform a full SELinux relabel after boot. This is necessary because the **passwd** command recreates the **/etc/shadow** file without an SELinux context.

```
sh-4.4# touch /.autorelabel
```

6. Type **exit** twice to continue booting your system as usual. The system runs an SELinux relabel, then reboots again by itself. When the system is up, verify your work by logging in as `root` at the console. Use `redhat` as the password.

Finish

On `workstation`, run the **lab boot-resetting finish** script to complete this exercise.

```
[student@workstation ~]$ lab boot-resetting finish
```

This concludes the guided exercise.

REPAIRING FILE SYSTEM ISSUES AT BOOT

OBJECTIVES

After completing this section, you should be able to manually repair file-system configuration or corruption issues that stop the boot process.

DIAGNOSING AND FIXING FILE SYSTEM ISSUES

Errors in **`/etc/fstab`** and corrupt file systems can stop a system from booting. In most cases, `systemd` drops to an emergency repair shell that requires the `root` password.

The following table lists some common errors and their results.

Common File System Issues

PROBLEM	RESULT
Corrupt file system	`systemd` attempts to repair the file system. If the problem is too severe for an automatic fix, the system drops the user to an emergency shell.
Nonexistent device or UUID referenced in **`/etc/fstab`**	`systemd` waits for a set amount of time, waiting for the device to become available. If the device does not become available, the system drops the user to an emergency shell after the timeout.
Nonexistent mount point in **`/etc/fstab`**	The system drops the user to an emergency shell.
Incorrect mount option specified in **`/etc/fstab`**	The system drops the user to an emergency shell.

In all cases, administrators can also use the emergency target to diagnose and fix the issue, because no file systems are mounted before the emergency shell is displayed.

NOTE

When using the emergency shell to address file-system issues, do not forget to run **`systemctl daemon-reload`** after editing **`/etc/fstab`**. Without this reload, `systemd` may continue using the old version.

The **`nofail`** option in an entry in the **`/etc/fstab`** file permits the system to boot even if the mount of that file system is not successful. *Do not* use this option under normal circumstances. With **`nofail`**, an application can start with its storage missing, with possibly severe consequences.

REFERENCES

`systemd-fsck`(8), **`systemd-fstab-generator`**(8), and **`systemd.mount`**(5) man pages

GUIDED EXERCISE

REPAIRING FILE SYSTEM ISSUES AT BOOT

In this exercise, you will recover a system from a misconfiguration in **/etc/fstab** that causes the boot process to fail.

OUTCOMES

You should be able to diagnose **/etc/fstab** issues and use emergency mode to recover the system.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **lab boot-repairing start** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also introduces a file-system issue, sets a higher timeout for the GRUB2 menu, and reboots `servera`.

```
[student@workstation ~]$ lab boot-repairing start
```

1. Access the `servera` console and notice that the boot process is stuck early on.

   1.1. Locate the icon for the `servera` console, as appropriate for your classroom environment. Open the console.

   Notice that a start job does not seem to complete. Take a minute to speculate about a possible cause for this behavior.

   1.2. To reboot, send a **Ctrl+Alt+Del** to your system using the relevant button or menu entry. With this particular boot problem, this key sequence may not immediately abort the running job, and you may have to wait for it to time out before the system reboots.

   If you wait for the task to time out without sending a **Ctrl+Alt+Del**, the system eventually spawns an emergency shell by itself.

   1.3. When the boot-loader menu appears, press any key to interrupt the countdown, except **Enter**.

2. Looking at the error from the previous boot, it appears that at least parts of the system are still functioning. Because you know the `root` password, `redhat`, attempt an emergency boot.

   2.1. Use the cursor keys to highlight the default boot loader entry.

   2.2. Press **e** to edit the current entry.

   2.3. Use the cursor keys to navigate to the line that starts with **linux**.

2.4. Press **End** to move the cursor to the end of the line.

2.5. Append **`systemd.unit=emergency.target`** to the end of the line.

2.6. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

3. Log in to emergency mode. The `root` password is `redhat`.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@servera ~]#
```

4. Determine which file systems are currently mounted.

```
[root@servera ~]# mount
...output omitted...
/dev/vda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
...output omitted...
```

Notice that the root file system is mounted read-only.

5. Remount the root file system read/write.

```
[root@servera ~]# mount -o remount,rw /
```

6. Use the **`mount -a`** command to attempt to mount all the other file systems. With the **`--all`** (**`-a`**) option, the command mounts all the file systems listed in **`/etc/fstab`** that are not yet mounted.

```
[root@servera ~]# mount -a
mount: /RemoveMe: mount point does not exist.
```

7. Edit **`/etc/fstab`** to fix the issue.

7.1. Remove or comment out the incorrect line.

```
[root@servera ~]# vim /etc/fstab
...output omitted...
# /dev/sdz1   /RemoveMe   xfs   defaults   0 0
```

7.2. Update `systemd` for the system to register the new **`/etc/fstab`** configuration.

```
[root@servera ~]# systemctl daemon-reload
[root@servera ~]#
```

8. Verify that your **/etc/fstab** is now correct by attempting to mount all entries.

```
[root@servera ~]# mount -a
[root@servera ~]#
```

9. Reboot the system and wait for the boot to complete. The system should now boot normally.

```
[root@servera ~]# systemctl reboot
```

Finish

On workstation, run the **lab boot-repairing finish** script to complete this exercise.

```
[student@workstation ~]$ lab boot-repairing finish
```

This concludes the guided exercise.

▷ LAB

CONTROLLING SERVICES AND DAEMONS

PERFORMANCE CHECKLIST

In this lab, you will configure several services to be enabled or disabled, running or stopped, based on a specification that is provided to you.

OUTCOMES

You should be able to enable, disable, start, and stop services.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab services-review start`** command. The command runs a start script that determines whether the host, `serverb`, is reachable on the network. The script also ensures that the `psacct` and `rsyslog` services are configured appropriately on `serverb`.

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

1. On `serverb`, 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 `serverb` before evaluating the lab.

Evaluation

On `workstation`, run the **`lab services-review grade`** script to confirm success on this lab.

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

Finish

On `workstation`, run the **`lab services-review finish`** script to complete this lab.

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

This concludes the lab.

▷ SOLUTION

CONTROLLING SERVICES AND DAEMONS

PERFORMANCE CHECKLIST

In this lab, you will configure several services to be enabled or disabled, running or stopped, based on a specification that is provided to you.

OUTCOMES

You should be able to enable, disable, start, and stop services.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab services-review start`** command. The command runs a start script that determines whether the host, `serverb`, is reachable on the network. The script also ensures that the `psacct` and `rsyslog` services are configured appropriately on `serverb`.

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

1. On `serverb`, start the `psacct` service.

   1.1. Use the **ssh** command to log in to `serverb` as the `student` user.

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

   1.2. Use the **`systemctl`** command to verify the status of the `psacct` service. Notice that `psacct` is stopped and disabled to start at boot time.

   ```
   [student@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.

   ```
   [student@serverb ~]$ sudo systemctl start psacct
   [sudo] password for student: student
   [student@serverb ~]$
   ```

1.4. Verify that the psacct service is running.

```
[student@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.

```
[student@serverb ~]$ sudo 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.

```
[student@serverb ~]$ systemctl is-enabled psacct
enabled
```

3. Stop the rsyslog service.

3.1. Use the **systemctl** command to verify the status of the rsyslog service. Notice that the rsyslog service is running and enabled to start at boot time.

```
[student@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 Fri 2019-02-08 10:16:00 IST; 2h 34min
 ago
  ...output omitted...
```

Press **q** to exit the command.

3.2. Stop the rsyslog service.

```
[student@serverb ~]$ sudo systemctl stop rsyslog
[sudo] password for student: student
[student@serverb ~]$
```

3.3. Verify that the rsyslog service is stopped.

```
[student@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.

```
[student@serverb ~]$ sudo systemctl disable rsyslog
Removed /etc/systemd/system/syslog.service.
Removed /etc/systemd/system/multi-user.target.wants/rsyslog.service.
```

   4.2. Verify that the `rsyslog` is disabled to start at system boot.

```
[student@serverb ~]$ systemctl is-enabled rsyslog
disabled
```

5. Reboot `serverb` before evaluating the lab.

```
[student@serverb ~]$ sudo systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On `workstation`, run the **`lab services-review grade`** script to confirm success on this lab.

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

Finish

On `workstation`, run the **`lab services-review finish`** script to complete this lab.

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

SUMMARY

In this chapter, you learned:

- Use the **`systemctl status`** command to determine the status of system daemons and network services started by `systemd`.
- The **`systemctl list-dependencies`** command lists all service units upon which a specific service unit depends.
- **`systemctl reboot`** and **`systemctl poweroff`** reboot and power down a system, respectively.
- **`systemctl isolate`** ***`desired`***`.`**`target`** switches to a new target at runtime.
- **`systemctl get-default`** and **`systemctl set-default`** can be used to query and set the default target.
- Use **`rd.break`** on the kernel command-line to interrupt the boot process before control is handed over from the **`initramfs`**. The root file system is mounted read-only under **`/sysroot`**.
- The `emergency.target` target can be used to diagnose and fix file system issues.

CHAPTER 10

MANAGING NETWORKING

GOAL

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

OBJECTIVES

- Test and inspect current network configuration with command-line utilities.
- Manage network settings and devices using **`nmcli`**.
- Modify network settings by editing the configuration files.
- Configure a server's static host name and its name resolution, and test the results.

SECTIONS

- Validating Network Configuration (and Guided Exercise)
- Configuring Networking from the Command Line (and Guided Exercise)
- Editing Network Configuration Files (and Guided Exercise)
- Configuring Host Names and Name Resolution (and Guided Exercise)

LAB

Managing Networking

VALIDATING NETWORK CONFIGURATION

OBJECTIVES

After completing this section, you should be able to test and inspect current network configuration with command-line utilities.

GATHERING NETWORK INTERFACE INFORMATION

Identifying Network Interfaces

The **`ip link`** command will list all network interfaces available on your system:

```
[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
```

In the preceding 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`.

To configure each network interface correctly, you need to know which one is connected to which network. In many cases, you will know the MAC address of the interface 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`.

Displaying 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,❶UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
 qlen 1000
    ❷link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    ❸inet 192.0.2.2/24 brd 192.0.2.255 scope global ens3
       valid_lft forever preferred_lft forever
    ❹inet6 2001:db8:0:1:5054:ff:fe00:b/64 scope global
```

```
      valid_lft forever preferred_lft forever
❺ inet6 fe80::5054:ff:fe00:b/64 scope link
      valid_lft forever preferred_lft forever
```

❶ An active interface is UP.
❷ The `link/ether` line specifies the hardware (MAC) address of the device.
❸ The `inet` line shows an IPv4 address, its network prefix length, and scope.
❹ The `inet6` line shows an IPv6 address, its network prefix length, and scope. This address is of *global* scope and is normally used.
❺ This `inet6` line shows that the interface has an IPv6 address of *link* scope that can only be used for communication on the local Ethernet link.

Displaying Performance Statistics

The **ip** command may also be used to show statistics about network performance. Counters for each network interface can be used to identify the presence of network issues. The counters record statistics for things like the number of received (RX) and transmitted (TX) packets, packet errors, and packets that were dropped.

```
[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
```

CHECKING CONNECTIVITY BETWEEN HOSTS

The **ping** command is used to test connectivity. The command continues to run until **Ctrl+c** is pressed unless options are given to limit the number of packets sent.

```
[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 **ping** in Red Hat Enterprise Linux. It communicates over IPv6 and takes IPv6 addresses, but otherwise works like **ping**.

```
[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
```

When you ping 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 is left out, the error *connect: Invalid argument* is displayed.

Pinging `ff02::1` can be useful for finding 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 ~]$ 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
```

Remember that IPv6 link-local addresses can be used by other hosts on the same link, just 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 ~]$
```

TROUBLESHOOTING ROUTING

Network routing is complex, and sometimes traffic does not behave as you might have expected. Here are some useful diagnosis tools.

Displaying the Routing Table

Use the **`ip`** command with the **`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
```

This shows the IPv4 routing table. All packets destined for the `10.0.0.0/8` network are sent directly to the destination through the device `ens4`. All packets destined for the `192.0.2.0/24`

network are sent directly to the destination through the device ens3. All other packets are sent to the default router located at 192.0.2.254, and also through device ens3.

Add the **-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
```

In this example, ignore the unreachable routes, which point at unused networks. That leaves three routes:

1. The 2001:db8:0:1::/64 network, using the ens3 interface (which presumably has an address on that network).
2. The fe80::/64 network, using the ens3 interface, for the link-local address. On a system with multiple interfaces, there is a route to fe80::/64 out each interface for each link-local address.
3. A default route to all networks on the IPv6 Internet (the ::/0 network) that do not have a more specific route on the system, through the router at 2001:db8:0:1::ffff, reachable with the ens3 device.

Tracing Routes Taken by Traffic

To trace the path that network traffic takes to reach a remote host through multiple routers, use either **traceroute** or **tracepath**. This can identify whether an issue is with one of your routers or an intermediate one. 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, however, the **traceroute** command is not installed by default.

```
[user@host ~]$ tracepath access.redhat.com
...output omitted...
 4:  71-32-28-145.rcmt.qwest.net                    48.853ms asymm  5
 5:  dcp-brdr-04.inet.qwest.net                    100.732ms asymm  7
 6:  206.111.0.153.ptr.us.xo.net                    96.245ms asymm  7
 7:  207.88.14.162.ptr.us.xo.net                    85.270ms asymm  8
 8:  ae1d0.cir1.atlanta6-ga.us.xo.net               64.160ms asymm  7
 9:  216.156.108.98.ptr.us.xo.net                  108.652ms
10:  bu-ether13.atlngamq46w-bcr00.tbone.rr.com     107.286ms asymm 12
...output omitted...
```

Each line in the output of **tracepath** represents a router or *hop* that the packet passes through between the source and the final destination. Additional information is provided as available,

including the *round trip timing (RTT)* and any changes in the *maximum transmission unit (MTU)* size. The `asymm` indication means traffic reached that router and returned from that router using different (*asymmetric*) routes. The routers shown are the ones used for outbound traffic, not the return traffic.

The **`tracepath6`** and **`traceroute -6`** commands are the equivalent to **`tracepath`** and **`traceroute`** for IPv6.

```
[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
```

TROUBLESHOOTING PORTS AND SERVICES

TCP services use sockets as end points for communication and are made up of an IP address, protocol, and port number. Services typically listen on standard ports while clients use a random available port. Well-known names for standard ports are listed in the **`/etc/services`** file.

The **`ss`** command is used to display socket statistics. The **`ss`** command is meant to replace the older tool **`netstat`**, part of the *net-tools* package, which may be more familiar to some system administrators but which 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                ❶*:ssh                        *:*
LISTEN     0      100          ❷127.0.0.1:smtp                     *:*
LISTEN     0      128                   *:36889                    *:*
ESTAB      0      0         ❸172.25.250.10:ssh        172.25.254.254:59392
LISTEN     0      128                  :::sunrpc                  :::*
LISTEN     0      128               ❹:::ssh                       :::*
LISTEN     0      100               ❺::1:smtp                     :::*
LISTEN     0      128                  :::34946                   :::*
```

❶ The port used for SSH is listening on all IPv4 addresses. The "*" is used to represent "all" when referencing IPv4 addresses or ports.
❷ The port used for SMTP is listening on the `127.0.0.1` IPv4 loopback interface.
❸ 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`.
❹ The port used for SSH is listening on all IPv6 addresses. The "::" syntax is used to represent all IPv6 interfaces.
❺ The port 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.

OPTION	DESCRIPTION
-t	Show TCP sockets.
-u	Show UDP sockets.
-l	Show only listening sockets.
-a	Show all (listening and established) sockets.
-p	Show the process using the sockets.
-A inet	Display active connections (but not listening sockets) for the `inet` address family. That is, ignore local UNIX domain sockets. For **ss**, both IPv4 and IPv6 connections are displayed. For **netstat**, only IPv4 connections are displayed. (**netstat -A inet6** displays IPv6 connections, and **netstat -46** displays IPv4 and IPv6 at the same time.)

REFERENCES

`ip-link`(8), `ip-address`(8), `ip-route`(8), `ip`(8), `ping`(8), `tracepath`(8), `traceroute`(8), `ss`(8), and `netstat`(8) man pages

For more information, refer to the *Configuring and Managing Networking* in *Red Hat Enterprise Linux 8.0* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

GUIDED EXERCISE

VALIDATING NETWORK CONFIGURATION

In this exercise, you will inspect the network configuration of one of your servers.

OUTCOMES

Identify the current network interfaces and basic network addresses.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab net-validate start`** command. The command runs a start script that determine if the host, `servera`, is reachable on the network.

```
[student@workstation ~]$ lab net-validate start
```

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.

> **IMPORTANT**
> Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names will vary according to the course platform and hardware in use.
>
> On your system now, locate the interface name (such as `ens06` or `en1p2`) associated with the Ethernet address `52:54:00:00:fa:0a`. Use this interface name to replace the en*X* placeholder used throughout this exercise.

Locate the network interface name associated with the Ethernet address `52:54:00:00:fa:0a`. Record or remember this name and use it to replace the en*X* placeholder in subsequent commands.

```
[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 addr
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
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
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
    inet 172.25.250.10/24 brd 172.25.250.255 scope global noprefixroute ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::3059:5462:198:58b2/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

4. Display the statistics for the en*X* 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 routing information.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev enX proto static metric 100
172.25.250.0/24 dev enX proto kernel scope link src 172.25.250.10 metric 100
```

6. Verify that the router is accessible.

```
[student@servera ~]$ ping -c3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.196 ms
64 bytes from 172.25.250.254: icmp_seq=2 ttl=64 time=0.436 ms
```

```
64 bytes from 172.25.250.254: icmp_seq=3 ttl=64 time=0.361 ms

--- 172.25.250.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 49ms
rtt min/avg/max/mdev = 0.196/0.331/0.436/0.100 ms
```

7. Show all the hops between the local system and `classroom.example.com`.

```
[student@servera ~]$ tracepath classroom.example.com
 1?: [LOCALHOST]                      pmtu 1500
 1:  workstation.lab.example.com                          0.270ms
 1:  workstation.lab.example.com                          0.167ms
 2:  classroom.example.com                                0.473ms 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. Exit from `servera`.

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

Finish

On `workstation`, run the **`lab net-validate finish`** script to finish this exercise.

```
[student@workstation ~]$ lab net-validate finish
```

This concludes the guided exercise.

CONFIGURING NETWORKING FROM THE COMMAND LINE

OBJECTIVES

After completing this section, you should be able to manage network settings and devices using the **nmcli** command.

DESCRIBING NETWORKMANAGER CONCEPTS

NetworkManager is a daemon that monitors and manages network settings. In addition to the daemon, there is a GNOME Notification Area applet providing network status information. Command-line and graphical tools talk to NetworkManager and save configuration files in the **/etc/sysconfig/network-scripts** directory.

- A *device* is a network interface.
- A *connection* is a collection of settings that can be configured for a device.
- Only one connection can be *active* for any one device at a time. Multiple connections may exist for use by different devices or to allow a configuration to be altered for the same device. If you need to temporarily change networking settings, instead of changing the configuration of a connection, you can change which connection is active for a device. For example, a device for a wireless network interface on a laptop might use different connections for the wireless network at a work site and for the wireless network at home.
- Each connection has a *name* or ID that identifies it.
- The **nmcli** utility is used to create and edit connection files from the command line.

VIEWING NETWORKING INFORMATION

The **nmcli dev 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     --
```

The **nmcli con show** command displays a list of all connections. To list only the active connections, add the **--active** option.

```
[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
```

ADDING A NETWORK CONNECTION

The **nmcli con add** command is used to add new network connections. The following example **nmcli con add** commands assume that the name of the network connection being added is not already in use.

The following command adds a new connection named eno2 for the interface eno2, which gets IPv4 networking information using DHCP and autoconnects on startup. It also gets IPv6 networking settings by listening for router advertisements on the local link. The name of the configuration file is based on the value of the **con-name** option, eno2, and is saved to the **/etc/sysconfig/network-scripts/ifcfg-eno2** file.

```
[root@host ~]# nmcli con add con-name eno2 type ethernet ifname eno2
```

The next example creates an eno2 connection for the eno2 device with a static IPv4 address, using the IPv4 address and network prefix `192.168.0.5/24` and default gateway `192.168.0.254`, but still autoconnects at startup and saves its configuration into the same file. Due to screen size limitations, terminate the first line with a shell \ escape and complete the command on the next line.

```
[root@host ~]# nmcli con add con-name eno2 type ethernet ifname eno2 \
ipv4.address 192.168.0.5/24 ipv4.gateway 192.168.0.254
```

This final example creates an eno2 connection for the eno2 device with static IPv6 and IPv4 addresses, using the IPv6 address and network prefix `2001:db8:0:1::c000:207/64` and default IPv6 gateway `2001:db8:0:1::1`, and the IPv4 address and network prefix `192.0.2.7/24` and default IPv4 gateway `192.0.2.1`, but still autoconnects at startup and saves its configuration into **/etc/sysconfig/network-scripts/ifcfg-eno2**. Due to screen size limitations, terminate the first line with a shell \ escape and complete the command on the next line.

```
[root@host ~]# nmcli con add con-name eno2 type ethernet ifname eno2 \
ipv6.address 2001:db8:0:1::c000:207/64 ipv6.gateway 2001:db8:0:1::1 \
ipv4.address 192.0.2.7/24 ipv4.gateway 192.0.2.1
```

CONTROLLING NETWORK CONNECTIONS

The **nmcli con up *name*** command activates the connection *name* on the network interface it is bound to. Note that the command takes the name of a *connection*, not the name of the network interface. Remember that the **nmcli con show** command displays the names of *all* available connections.

```
[root@host ~]# nmcli con up static-ens3
```

The **nmcli dev disconnect *device*** command disconnects the network interface *device* and brings it down. This command can be abbreviated **nmcli dev dis *device***:

```
[root@host ~]# nmcli dev dis ens3
```

IMPORTANT

Use **nmcli dev dis *device*** to deactivate a network interface.

The **nmcli con down *name*** command is normally not the best way to deactivate a network interface because it brings down the connection. However, by default, most wired system connections are configured with **autoconnect** enabled. This activates the connection as soon as its network interface is available. Since the connection's network interface is still available, **nmcli con down *name*** brings the interface down, but then NetworkManager immediately brings it up again unless the connection is entirely disconnected from the interface.

MODIFYING NETWORK CONNECTION SETTINGS

NetworkManager connections have two kinds of settings. There are *static* connection properties, configured by the administrator and stored in the configuration files in **/etc/sysconfig/network-scripts/ifcfg-***. There may also be *active* connection data, which the connection gets from a DHCP server and which are not stored persistently.

To list the current settings for a connection, run the **nmcli con show *name*** command, where *name* is the name of the connection. Settings in lowercase are static properties that the administrator can change. Settings in all caps are active settings in temporary use for this instance of the connection.

```
[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...
```

The **`nmcli con mod`** ***`name`*** command is used to change the settings for a connection. These changes are also saved in the **`/etc/sysconfig/network-scripts/ifcfg-`*`name`*** file for the connection. Available settings are documented in the `nm-settings`(5) man page.

To set the IPv4 address to `192.0.2.2/24` and default gateway to `192.0.2.254` for the connection `static-ens3`:

```
[root@host ~]# nmcli con mod static-ens3 ipv4.address 192.0.2.2/24 \
ipv4.gateway 192.0.2.254
```

To set the IPv6 address to `2001:db8:0:1::a00:1/64` and default gateway to `2001:db8:0:1::1` for the connection `static-ens3`:

```
[root@host ~]# nmcli con mod static-ens3 ipv6.address 2001:db8:0:1::a00:1/64 \
ipv6.gateway 2001:db8:0:1::1
```

IMPORTANT

If a connection that gets its IPv4 information from a DHCPv4 server is being changed to get it from static configuration files only, the setting **`ipv4.method`** should also be changed from `auto` to `manual`.

Likewise, if a connection that gets its IPv6 information by SLAAC or a DHCPv6 server is being changed to get it from static configuration files only, the setting **`ipv6.method`** should also be changed from `auto` or `dhcp` to `manual`.

Otherwise, the connection may hang or not complete successfully when it is activated, or it may get an IPv4 address from DHCP or an IPv6 address from DHCPv6 or SLAAC in addition to the static address.

A number of settings may have multiple values. A specific value can be added to the list or deleted from the list for a setting by adding a + or - symbol to the start of the setting name.

DELETING A NETWORK CONNECTION

The **`nmcli con del`** ***`name`*** command deletes the connection named *name* from the system, disconnecting it from the device and removing the file **`/etc/sysconfig/network-scripts/ifcfg-`***`name`*.

```
[root@host ~]# nmcli con del static-ens3
```

WHO CAN MODIFY NETWORK SETTINGS?

The `root` user can make any necessary network configuration changes with **`nmcli`**.

However, regular users that are logged in on the local console can also make many network configuration changes to the system. They have to log in at the system's keyboard to either a text-based virtual console or the graphical desktop environment to get this control. The logic behind this is that if someone is physically present at the computer's console, it's likely being used as a workstation or laptop and they may need to configure, activate, and deactivate wireless or wired network interfaces at will. By contrast, if the system is a server in the datacenter, generally the only users logging in locally to the machine itself should be administrators.

Regular users that log in using **`ssh`** do not have access to change network permissions without becoming `root`.

You can use the **`nmcli gen permissions`** command to see what your current permissions are.

SUMMARY OF COMMANDS

The following table is a list of key **`nmcli`** commands discussed in this section.

COMMAND	PURPOSE
`nmcli dev status`	Show the NetworkManager status of all network interfaces.
`nmcli con show`	List all connections.
`nmcli con show` ***`name`***	List the current settings for the connection *name*.

COMMAND	PURPOSE
`nmcli con add con-name name`	Add a new connection named *name*.
`nmcli con mod name`	Modify the connection *name*.
`nmcli con reload`	Reload the configuration files (useful after they have been edited by hand).
`nmcli con up name`	Activate the connection *name*.
`nmcli dev dis dev`	Deactivate and disconnect the current connection on the network interface *dev*.
`nmcli con del name`	Delete the connection *name* and its configuration file.

REFERENCES

`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

CONFIGURING NETWORKING FROM THE COMMAND LINE

In this exercise, you will configure network settings using **nmcli**.

OUTCOMES

You should be able to convert a system from DHCP to static configuration.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **lab net-configure start** command. The command runs a start script that determine if the host, `servera`, is reachable on the network.

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

> **NOTE**
> If prompted by the **sudo** command for `student`'s password, enter `student` as the password.

1. Use the **ssh** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to `servera`.

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

2. Locate network interface names.

> **IMPORTANT**
> Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names will vary according to the course platform and hardware in use.
>
> On your system now, locate the interface name (such as `ens06` or `en1p2`) associated with the Ethernet address `52:54:00:00:fa:0a`. Use this interface name to replace the en*X* placeholder used throughout this exercise.

Locate the network interface name associated with the Ethernet address `52:54:00:00:fa:0a`. Record or remember this name and use it to replace the en*X* placeholder in subsequent commands.

```
[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. View network settings using **`nmcli`**.

3.1. Show all connections.

```
[student@servera ~]$ nmcli con show
NAME                UUID                                  TYPE
 DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet
 enX
```

3.2. Display only the active connection.

Your network interface name should appear under DEVICE, and the name of the connection active for that device is listed on the same line under NAME. This exercise assumes that the active connection is `Wired connection 1`.

If the name of the active connection is different, use that instead of `Wired connection 1` for the rest of this exercise.

```
[student@servera ~]$ nmcli con show --active
NAME                UUID                                  TYPE
 DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet
  enX
```

3.3. Display all configuration settings for the active connection.

```
[student@servera ~]$ nmcli con show "Wired connection 1"
connection.id:                Wired connection 1
connection.uuid:              03da038a-3257-4722-a478-53055cc90128
connection.stable-id:         --
connection.type:              802-3-ethernet
connection.interface-name:    --
connection.autoconnect:       yes
...output omitted...
ipv4.method:                  manual
ipv4.dns:                     172.25.250.254
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:                   Wired connection 1
GENERAL.UUID:                   03da038a-3257-4722-a478-53055cc90128
GENERAL.DEVICES:                enX
GENERAL.STATE:                  activated
GENERAL.DEFAULT:                yes
GENERAL.DEFAULT6:               no
GENERAL.SPEC-OBJECT:            --
GENERAL.VPN:                    no
GENERAL.DBUS-PATH:              /org/freedesktop/NetworkManager/
ActiveConnection/1
GENERAL.CON-PATH:               /org/freedesktop/NetworkManager/
Settings/1
GENERAL.ZONE:                   --
GENERAL.MASTER-PATH:            --
IP4.ADDRESS[1]:                 172.25.250.10/24
IP4.GATEWAY:                    172.25.250.254
IP4.ROUTE[1]:                   dst = 172.25.250.0/24, nh = 0.0.0.0, mt =
 100
IP4.ROUTE[2]:                   dst = 0.0.0.0/0, nh = 172.25.250.254, mt
 = 100
IP4.DNS[1]:                     172.25.250.254
IP6.ADDRESS[1]:                 fe80::3059:5462:198:58b2/64
IP6.GATEWAY:                    --
IP6.ROUTE[1]:                   dst = fe80::/64, nh = ::, mt = 100
IP6.ROUTE[2]:                   dst = ff00::/8, nh = ::, mt = 256,
 table=255
```

Press **q** to exit the command.

3.4. Show device status.

```
[student@servera ~]$ nmcli dev status
DEVICE  TYPE      STATE      CONNECTION
enX     ethernet  connected  Wired connection 1
lo      loopback  unmanaged  --
```

3.5. Display the settings for the en*X* device.

```
[student@servera ~]$ nmcli dev show enX
GENERAL.DEVICE:                 enX
GENERAL.TYPE:                   ethernet
GENERAL.HWADDR:                 52:54:00:00:FA:0A
GENERAL.MTU:                    1500
GENERAL.STATE:                  100 (connected)
GENERAL.CONNECTION:             Wired connection 1
GENERAL.CON-PATH:               /org/freedesktop/NetworkManager/
ActiveConnection/1
WIRED-PROPERTIES.CARRIER:       on
IP4.ADDRESS[1]:                 172.25.250.10/24
IP4.GATEWAY:                    172.25.250.254
```

```
IP4.ROUTE[1]:                    dst = 172.25.250.0/24, nh = 0.0.0.0, mt =
 100
IP4.ROUTE[2]:                    dst = 0.0.0.0/0, nh = 172.25.250.254, mt
 = 100
IP4.DNS[1]:                      172.25.250.254
IP6.ADDRESS[1]:                  fe80::3059:5462:198:58b2/64
IP6.GATEWAY:                     --
IP6.ROUTE[1]:                    dst = fe80::/64, nh = ::, mt = 100
IP6.ROUTE[2]:                    dst = ff00::/8, nh = ::, mt = 256,
 table=255
```

4. Create a static connection with the same IPv4 address, network prefix, and default gateway. Name the new connection *static-addr*.

```
[student@servera ~]$ sudo nmcli con add con-name "static-addr" ifname enX \
type ethernet ipv4.method manual \
ipv4.address 172.25.250.10/24 ipv4.gateway 172.25.250.254
Connection 'static-addr' (15aa3901-555d-40cb-94c6-cea6f9151634) successfully
 added.
```

5. Modify the new connection to add the DNS setting.

```
[student@servera ~]$ sudo nmcli con mod "static-addr" ipv4.dns
 172.25.250.254
```

6. Display and activate the new connection.

6.1. View all connections.

```
[student@servera ~]$ nmcli con show
NAME                UUID                                  TYPE
 DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet
 enX
static-addr         15aa3901-555d-40cb-94c6-cea6f9151634  ethernet  --
```

6.2. View the active connection.

```
[student@servera ~]$ nmcli con show --active
NAME                UUID                                  TYPE
 DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet
 enX
```

6.3. Activate the new `static-addr` connection.

```
[student@servera ~]$ sudo nmcli con up "static-addr"
Connection successfully activated (D-Bus active path: /org/
freedesktop/NetworkManager/ActiveConnection/2)
```

6.4. Verify the new active connection.

```
[student@servera ~]$ nmcli con show --active
NAME         UUID                                  TYPE      DEVICE
static-addr  15aa3901-555d-40cb-94c6-cea6f9151634  ethernet  enX
```

7. Configure the original connection so that it does not start at boot, and verify that the static connection is used when the system reboots.

7.1. Disable the original connection from autostarting at boot.

```
[student@servera ~]$ sudo nmcli con mod "Wired connection 1" \
connection.autoconnect no
```

7.2. Reboot the system.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

7.3. View the active connection.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ nmcli con show --active
NAME         UUID                                  TYPE      DEVICE
static-addr  15aa3901-555d-40cb-94c6-cea6f9151634  ethernet  enX
```

8. Test connectivity using the new network addresses.

8.1. Verify the IP address.

```
[student@servera ~]$ ip addr show enX
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
    inet 172.25.250.10/24 brd 172.25.250.255 scope global
 noprefixroute enX
       valid_lft forever preferred_lft forever
    inet6 fe80::6556:cdd9:ce15:1484/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

8.2. Verify the default gateway.

```
[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
```

8.3. Ping the DNS address.

```
[student@servera ~]$ ping -c3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.225 ms
64 bytes from 172.25.250.254: icmp_seq=2 ttl=64 time=0.314 ms
64 bytes from 172.25.250.254: icmp_seq=3 ttl=64 time=0.472 ms
```

```
--- 172.25.250.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 46ms
rtt min/avg/max/mdev = 0.225/0.337/0.472/0.102 ms
```

8.4. Exit from servera.

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

Finish

On workstation, run the **lab net-configure finish** script to finish this exercise.

```
[student@workstation ~]$ lab net-configure finish
```

This concludes the guided exercise.

EDITING NETWORK CONFIGURATION FILES

OBJECTIVES

After completing this section, you should be able to modify network configuration by editing configuration files.

DESCRIBING CONNECTION CONFIGURATION FILES

By default, changes made with **`nmcli con mod`** ***`name`*** are automatically saved to **`/etc/sysconfig/network-scripts/ifcfg-`*****`name`**. That file can also be manually edited with a text editor. After doing so, run **`nmcli con reload`** so that NetworkManager reads the configuration changes.

For backward-compatibility reasons, the directives saved in that file have different names and syntax than the `nm-settings`(5) names. The following table maps some of the key setting names to **`ifcfg-*`** directives.

Comparison of nm-settings and ifcfg-* Directives

NMCLI CON MOD	IFCFG-* FILE	EFFECT
`ipv4.method manual`	**`BOOTPROTO=none`**	IPv4 addresses configured statically.
`ipv4.method auto`	**`BOOTPROTO=dhcp`**	Looks for configuration settings from a DHCPv4 server. If static addresses are also set, will not bring those up until we have information from DHCPv4.
`ipv4.addresses "192.0.2.1/24 192.0.2.254"`	**`IPADDR0=192.0.2.1`** **`PREFIX0=24`** **`GATEWAY0=192.0.2.254`**	Sets static IPv4 address, network prefix, and default gateway. If more than one is set for the connection, then instead of 0, the **`ifcfg-*`** directives end with 1, 2, 3 and so on.
`ipv4.dns 8.8.8.8`	**`DNS0=8.8.8.8`**	Modify **`/etc/resolv.conf`** to use this **`nameserver`**.
`ipv4.dns-search example.com`	**`DOMAIN=example.com`**	Modify **`/etc/resolv.conf`** to use this domain in the **`search`** directive.
`ipv4.ignore-auto-dns true`	**`PEERDNS=no`**	Ignore DNS server information from the DHCP server.

NMCLI CON MOD	IFCFG-* FILE	EFFECT
ipv6.method manual	**IPV6_AUTOCONF=no**	IPv6 addresses configured statically.
ipv6.method auto	**IPV6_AUTOCONF=yes**	Configures network settings using SLAAC from router advertisements.
ipv6.method dhcp	**IPV6_AUTOCONF=no DHCPV6C=yes**	Configures network settings by using DHCPv6, but not SLAAC.
ipv6.addresses "2001:db8::a/64 2001:db8::1"	**IPV6ADDR=2001:db8::a/64 IPV6_DEFAULTGW=2001:db8**	Sets static IPv6 address, network prefix, and default gateway. If more than one address is set for the connection, IPV6_SECONDARIES takes a double-quoted list of space-delimited address/prefix definitions.
ipv6.dns ...	**DNS0= ...**	Modify **/etc/resolv.conf** to use this nameserver. Exactly the same as IPv4.
ipv6.dns-search example.com	**DOMAIN=example.com**	Modify **/etc/resolv.conf** to use this domain in the **search** directive. Exactly the same as IPv4.
ipv6.ignore-auto-dns true	**IPV6_PEERDNS=no**	Ignore DNS server information from the DHCP server.
connection.autoconnect yes	**ONBOOT=yes**	Automatically activate this connection at boot.
connection.id ens3	**NAME=ens3**	The name of this connection.
connection.interface-name ens3	**DEVICE=ens3**	The connection is bound to the network interface with this name.
802-3-ethernet.mac-address . . .	**HWADDR= ...**	The connection is bound to the network interface with this MAC address.

MODIFYING NETWORK CONFIGURATION

It is also possible to 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 named **/etc/sysconfig/network-scripts/ifcfg-*name***, where

name refers to the name of the device or connection that the configuration file controls. The following are standard variables found in the file used for static or dynamic IPv4 configuration.

IPv4 Configuration Options for `ifcfg` File

STATIC	*DYNAMIC*	*EITHER*
BOOTPROTO=none	**BOOTPROTO=dhcp**	**DEVICE=ens3**
IPADDR0=172.25.250.10		**NAME="static-ens3"**
PREFIX0=24		**ONBOOT=yes**
GATEWAY0=172.25.250.254		**UUID=f3e8(...)ad3e**
DEFROUTE=yes		**USERCTL=yes**
DNS1=172.25.254.254		

In the static settings, variables for IP address, prefix, and gateway have a number at the end. This allows multiple sets of values to be assigned to the interface. The DNS variable also has a number used to specify the order of lookup when multiple servers are specified.

After modifying the configuration files, run **`nmcli con reload`** to make NetworkManager read the configuration changes. The interface still needs to be restarted for changes to take effect.

```
[root@host ~]# nmcli con reload
[root@host ~]# nmcli con down "static-ens3"
[root@host ~]# nmcli con up "static-ens3"
```

REFERENCES

`nmcli`(1) man page

For more information, refer to the *Configuring and Managing Networking* in *Red Hat Enterprise Linux 8.0* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

GUIDED EXERCISE

EDITING NETWORK CONFIGURATION FILES

In this exercise, you will manually modify network configuration files and ensure that the new settings take effect.

OUTCOMES

You should be able to add an additional network address to each system.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab net-edit start`** command. The command runs a start script that determine if the hosts, `servera` and `serverb`, are reachable on the network.

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

1. Use the **ssh** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to `servera`.

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

2. Locate network interface names.

> **IMPORTANT**
> Network interface names are determined by their bus type and the detection order of devices during boot. Your network interface names will vary according to the course platform and hardware in use.
>
> On your system now, locate the interface name (such as `ens06` or `en1p2`) associated with the Ethernet address `52:54:00:00:fa:0a`. Use this interface name to replace the en*X* placeholder used throughout this exercise.

Locate the network interface name associated with the Ethernet address `52:54:00:00:fa:0a`. Record or remember this name and use it to replace the en*X* placeholder in subsequent commands. The active connection is also named `Wired connection 1` (and therefore is managed by the file **`/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1`**).

If you have done previous exercises in this chapter and this was true for your system, it should be true for this exercise as well.

```
[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
[student@servera ~]$ nmcli con show --active
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
[student@servera ~]$ ls /etc/sysconfig/network-scripts/ifcfg-
Wired_connection_1
```

3. Edit the **/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1** file on servera to add an additional address of **10.0.1.1/24**.

   3.1. Append an entry to the file to specify the IPv4 address.

```
[student@servera ~]$ echo "IPADDR1=10.0.1.1" | \
sudo tee -a /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
[sudo] password for student: student
IPADDR1=10.0.1.1
```

   3.2. Append an entry to the file to specify the network prefix.

```
[student@servera ~]$ echo "PREFIX1=24" | \
sudo tee -a /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
PREFIX1=24
```

4. Activate the new address.

4.1. Reload the configuration changes.

```
[student@servera ~]$ sudo nmcli con reload
```

4.2. Restart the connection with the new settings.

```
[student@servera ~]$ sudo nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/
freedesktop/NetworkManager/ActiveConnection/3)
```

4.3. Verify the new IP address.

```
[student@servera ~]$ ip addr show enX
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
    inet 172.25.250.10/24 brd 172.25.250.255 scope global
 noprefixroute enX
       valid_lft forever preferred_lft forever
    inet 10.0.1.1/24 brd 10.0.1.255 scope global noprefixroute enX
       valid_lft forever preferred_lft forever
    inet6 fe80::4bf3:e1d9:3076:f8d7/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

4.4. Exit from `servera` to return to `workstation` as `student` user.

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

5. As the `student` user on `serverb`, edit the **`/etc/sysconfig/network-scripts/ifcfg-Wired_connection_1`** file to add an additional address of **`10.0.1.2/24`**, then load the new configuration.

5.1. From `workstation`, use the **`ssh`** command to log in to `serverb` as the `student` user.

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

5.2. Modify the **`ifcfg-Wired_connection_1`** file to add the second IPv4 address and network prefix.

```
[student@serverb ~]$ echo "IPADDR2=10.0.1.2" | \
sudo tee -a /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
[sudo] password for student: student
IPADDR2=10.0.1.2
[student@serverb ~]$ echo "PREFIX2=24" | \
sudo tee -a /etc/sysconfig/network-scripts/ifcfg-Wired_connection_1
PREFIX2=24
```

5.3. Reload the configuration changes.

```
[student@serverb ~]$ sudo nmcli con reload
```

5.4. Bring up the connection with the new settings.

```
[student@serverb ~]$ sudo nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/
freedesktop/NetworkManager/ActiveConnection/4)
```

5.5. Verify the new IP address.

```
[student@serverb ~]$ ip addr show enX
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:0b brd ff:ff:ff:ff:ff:ff
    inet 172.25.250.11/24 brd 172.25.250.255 scope global
 noprefixroute enX
       valid_lft forever preferred_lft forever
    inet 10.0.1.2/24 brd 10.0.1.255 scope global noprefixroute enX
       valid_lft forever preferred_lft forever
    inet6 fe80::74c:3476:4113:463f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

6. Test connectivity using the new network addresses.

6.1. From `serverb`, ping the new address of `servera`.

```
[student@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=0.342 ms
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.188 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.317 ms

--- 10.0.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 35ms
rtt min/avg/max/mdev = 0.188/0.282/0.342/0.068 ms
```

6.2. Exit from `serverb` to return to `workstation`.

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

6.3. From `workstation`, use the **ssh** command to access `servera` as the `student` user to ping the new address of `serverb`.

```
[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=0.269 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.338 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=0.361 ms

--- 10.0.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 48ms
rtt min/avg/max/mdev = 0.269/0.322/0.361/0.044 ms
```

Finish

On `workstation`, run the **`lab net-edit finish`** script to finish this exercise.

```
[student@workstation ~]$ lab net-edit finish
```

This concludes the guided exercise.

CONFIGURING HOST NAMES AND NAME RESOLUTION

OBJECTIVES

After completing this section, you should be able to configure a server's static host name and its name resolution and test the results.

CHANGING THE SYSTEM HOST NAME

The **hostname** command displays or temporarily modifies the system's fully qualified host name.

```
[root@host ~]# hostname
host@example.com
```

A static host name may be specified in the **/etc/hostname** file. The **hostnamectl** command is used to modify this file and may be used to view the status of the system's fully qualified host name. If this file does not exist, the host name is set by a reverse DNS query once the interface has an IP address assigned.

```
[root@host ~]# hostnamectl set-hostname host@example.com
[root@host ~]# hostnamectl status
   Static hostname: host.example.com
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 73ab164e278e48be9bf80e80714a8cd5
           Boot ID: 6b1cbc4177164ef58c0e9ed4adb2904f
    Virtualization: kvm
  Operating System: Red Hat Enterprise Linux 8.0 beta (Ootpa)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:8.0:beta
            Kernel: Linux 4.18.0-60.el8.x86_64
      Architecture: x86-64
[root@host ~]# cat /etc/hostname
host@example.com
```

IMPORTANT

In Red Hat Enterprise Linux 7 and later, the static host name is stored in **/etc/hostname**. Red Hat Enterprise Linux 6 and earlier stores the host name as a variable in the **/etc/sysconfig/network** file.

CONFIGURING NAME RESOLUTION

The *stub resolver* is used to convert host names to IP addresses or the reverse. It determines where to look based on the configuration of the **/etc/nsswitch.conf** file. By default, the contents of the **/etc/hosts** file are checked first.

```
[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 can be used to test host name resolution using the **/etc/hosts** file.

If an entry is not found in the **/etc/hosts** file, by default the stub resolver tries to look up the hostname by using a DNS nameserver. The **/etc/resolv.conf** file controls how this query is performed:

- `search`: a list of domain names to try with a short host name. Both this and `domain` should not be set in the same file; if they are, the last instance wins. See `resolv.conf`(5) for details.
- `nameserver`: the IP address of a nameserver to query. Up to three nameserver directives may be given to provide backups if one is down.

```
[root@host ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager updates the **/etc/resolv.conf** file using DNS settings in the connection configuration files. Use the **nmcli** to modify the connections.

```
[root@host ~]# nmcli con mod ID ipv4.dns IP
[root@host ~]# nmcli con down ID
[root@host ~]# nmcli con up ID
[root@host ~]# cat /etc/sysconfig/network-scripts/ifcfg-ID
...output omitted...
DNS1=8.8.8.8
...output omitted...
```

The default behavior of **nmcli con mod ID ipv4.dns IP** is to replace any previous DNS settings with the new IP list provided. A **+** or **-** symbol in front of the **ipv4.dns** argument adds or removes an individual entry.

```
[root@host ~]# nmcli con mod ID +ipv4.dns IP
```

To add the DNS server with IPv6 IP address `2001:4860:4860::8888` to the list of nameservers to use with the connection `static-ens3`:

```
[root@host ~]# nmcli con mod static-ens3 +ipv6.dns 2001:4860:4860::8888
```

NOTE

Static IPv4 and IPv6 DNS settings all end up as `nameserver` directives in **`/etc/resolv.conf`**. You should ensure that there is, at minimum, an IPv4-reachable name server listed (assuming a dual-stack system). It is better to have at least one name server using IPv4 and a second using IPv6 in case you have network issues with either your IPv4 or IPv6 networking.

Testing DNS Name Resolution

The **`host`** ***`HOSTNAME`*** command can be used to test DNS server connectivity.

```
[root@host ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
[root@host ~]# host 172.25.254.254
254.254.25.172.in-addr.arpa domain name pointer classroom.example.com.
```

IMPORTANT

DHCP automatically rewrites the **`/etc/resolv.conf`** file as interfaces are started unless you specify PEERDNS=**no** in the relevant interface configuration files. Set this using the **`nmcli`** command.

```
[root@host ~]# nmcli con mod "static-ens3" ipv4.ignore-auto-dns yes
```

REFERENCES

`nmcli`(1), `hostnamectl`(1), `hosts`(5), `getent`(1), `host`(1), and `resolv.conf`(5) man pages

For more information, refer to the *Configuring and Managing Networking* in *Red Hat Enterprise Linux 8.0* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_networking/

GUIDED EXERCISE

CONFIGURING HOST NAMES AND NAME RESOLUTION

In this exercise, you will manually configure the system's static host name, **/etc/hosts** file, and DNS name resolver.

OUTCOMES

You should be able to set a customized host name and configure name resolution settings.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **lab net-hostnames start** command. The command runs a start script that determine if the host, `servera`, is reachable on the network.

```
[student@workstation ~]$ lab net-hostnames start
```

1. Use the **ssh** command to log in to `servera` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to `servera`.

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

2. View the current host name settings.

   2.1. Display the current host name.

```
[student@servera ~]$ hostname
servera.lab.example.com
```

   2.2. Display the host name status.

```
[student@servera ~]$ hostnamectl status
   Static hostname: n/a
Transient hostname: servera.lab.example.com
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 73ab164e278e48be9bf80e80714a8cd5
           Boot ID: 76b13a300c944ab49445af778cb8f749
    Virtualization: kvm
  Operating System: Red Hat Enterprise Linux 8.0 (Ootpa)
```

```
     CPE OS Name: cpe:/o:redhat:enterprise_linux:8.0:GA
          Kernel: Linux 4.18.0-80.el8.x86_64
    Architecture: x86-64
```

3. Set a static host name to match the current transient host name.

   3.1. Change the host name and host name configuration file.

```
[student@servera ~]$ sudo hostnamectl set-hostname
 servera.lab.example.com
[sudo] password for student: student
[student@servera ~]$
```

   3.2. View the configuration file providing the host name at network start.

```
[student@servera ~]$ cat /etc/hostname
servera.lab.example.com
```

   3.3. Display the host name status.

```
[student@servera ~]$ hostnamectl status
   Static hostname: servera.lab.example.com
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 73ab164e278e48be9bf80e80714a8cd5
           Boot ID: 76b13a300c944ab49445af778cb8f749
    Virtualization: kvm
  Operating System: Red Hat Enterprise Linux 8.0 (Ootpa)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:8.0:GA
            Kernel: Linux 4.18.0-80.el8.x86_64
      Architecture: x86-64
```

4. Temporarily change the host name.

4.1. Change the host name.

```
[student@servera ~]$ sudo hostname testname
```

4.2. Display the current host name.

```
[student@servera ~]$ hostname
testname
```

4.3. View the configuration file providing the host name at network start.

```
[student@servera ~]$ cat /etc/hostname
servera.lab.example.com
```

4.4. Reboot the system.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

4.5. From `workstation` log in to `servera` as `student` user.

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

4.6. Display the current host name.

```
[student@servera ~]$ hostname
servera.lab.example.com
```

5. Add a local nickname for the classroom server.

5.1. Look up the IP address of the classroom.example.com.

```
[student@servera ~]$ host classroom.example.com
classroom.example.com has address 172.25.254.254
```

5.2. Modify **/etc/hosts** so that the additional name of **class** can be used to access the IP address 172.25.254.254.

```
[student@servera ~]$ sudo vim /etc/hosts
[student@servera ~]$ 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 classroom class
172.25.254.254 content.example.com content
...content omitted...
```

5.3. Look up the IP address of class.

```
[student@servera ~]$ host class
Host class not found: 2(SERVFAIL)
[student@servera ~]$ getent hosts class
172.25.254.254    classroom.example.com class
```

5.4. Ping class.

```
[student@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=64 time=0.397 ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=2
 ttl=64 time=0.447 ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=3
 ttl=64 time=0.470 ms

--- classroom.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.397/0.438/0.470/0.030 ms
```

5.5. Exit from servera.

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

Finish

On `workstation`, run the **`lab net-hostnames finish`** script to finish this exercise.

```
[student@workstation ~]$ lab net-hostnames finish
```

This concludes the guided exercise.

▷ LAB

MANAGING NETWORKING

PERFORMANCE CHECKLIST

In this lab, you will configure networking settings on a Red Hat Enterprise Linux server.

OUTCOMES

You should be able to configure two static IPv4 addresses for the primary network interface.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation` run the **`lab net-review start`** command. The command runs a start script that determine if the host, `serverb`, is reachable on the network.

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

1. Use the **`ssh`** command to log in to `serverb` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to `serverb`.
2. Use the **`sudo -i`** command to switch to the `root` user. If prompted, use `student` as the password.
3. Create a new connection with a static network connection using the settings in the table.

PARAMETER	SETTING
Connection name	lab
Interface name	en*X* (might vary, use the interface that has `52:54:00:00:fa:0b` as its MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

4. Configure the new connection to be autostarted. Other connections should not start automatically.
5. Modify the new connection so that it also uses the address 10.0.1.1/24.
6. Configure the **`hosts`** file so that 10.0.1.1 can be referenced as **`private`**.
7. Reboot the system.
8. From `workstation` use the **`ping`** command to verify that `serverb` is initialized.

Evaluation

On workstation, run the **lab net-review grade** script to confirm success on this lab.

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

Finish

On workstation, run the **lab net-review finish** script to finish this lab.

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

This concludes the lab.

▷ SOLUTION

MANAGING NETWORKING

PERFORMANCE CHECKLIST

In this lab, you will configure networking settings on a Red Hat Enterprise Linux server.

OUTCOMES

You should be able to configure two static IPv4 addresses for the primary network interface.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation` run the **`lab net-review start`** command. The command runs a start script that determine if the host, `serverb`, is reachable on the network.

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

1. Use the **ssh** command to log in to `serverb` as the `student` user. The systems are configured to use SSH keys for authentication, so a password is not required to log in to `serverb`.

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

2. Use the **`sudo -i`** command to switch to the `root` user. If prompted, use `student` as the password.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3. Create a new connection with a static network connection using the settings in the table.

PARAMETER	SETTING
Connection name	lab
Interface name	en*X* (might vary, use the interface that has `52:54:00:00:fa:0b` as its MAC address)
IP address	172.25.250.11/24
Gateway address	172.25.250.254
DNS address	172.25.250.254

Determine the interface name and the current active connection's name. The solution assumes that the interface name is en*X* and the connection name is `Wired connection 1`.

```
[root@serverb ~]# 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:0b brd ff:ff:ff:ff:ff:ff
[root@serverb ~]# nmcli con show --active
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  03da038a-3257-4722-a478-53055cc90128  ethernet  enX
```

Create the new `lab` connection profile based on the information in the table described in the instructions. Associate the profile with your network interface name listed in the output of the previous **`ip link`** command.

```
[root@serverb ~]# nmcli con add con-name lab ifname enX type ethernet \
ipv4.method manual \
ipv4.address 172.25.250.11/24 ipv4.gateway 172.25.250.254
[root@serverb ~]# nmcli con mod "lab" ipv4.dns 172.25.250.254
```

4. Configure the new connection to be autostarted. Other connections should not start automatically.

```
[root@serverb ~]# nmcli con mod "lab" connection.autoconnect yes
[root@serverb ~]# nmcli con mod "Wired connection 1" connection.autoconnect
 no
```

5. Modify the new connection so that it also uses the address 10.0.1.1/24.

```
[root@serverb ~]# nmcli con mod "lab" +ipv4.addresses 10.0.1.1/24
```

Or alternately:

```
[root@serverb ~]# echo "IPADDR1=10.0.1.1" \
>> /etc/sysconfig/network-scripts/ifcfg-lab
[root@serverb ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-
lab
```

6. Configure the **`hosts`** file so that 10.0.1.1 can be referenced as **`private`**.

```
[root@serverb ~]# echo "10.0.1.1 private" >> /etc/hosts
```

7. Reboot the system.

```
[root@serverb ~]# systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

8. From `workstation` use the **`ping`** command to verify that `serverb` is initialized.

```
[student@workstation ~]$ ping -c3 serverb
PING serverb.lab.example.com (172.25.250.11) 56(84) bytes of data.
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=1 ttl=64
 time=0.478 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
 time=0.504 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64
 time=0.513 ms

--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 78ms
rtt min/avg/max/mdev = 0.478/0.498/0.513/0.023 ms
[student@workstation ~]$
```

Evaluation

On workstation, run the **`lab net-review grade`** script to confirm success on this lab.

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

Finish

On `workstation`, run the **`lab net-review finish`** script to finish this lab.

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

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The TCP/IP network model is a simplified, four-layered set of abstractions that describes how different protocols interoperate in order for computers to send traffic from one machine to another over the Internet.
- IPv4 is the primary network protocol used 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, using both protocols in parallel.
- NetworkManager is a daemon that monitors and manages network configuration.
- The **`nmcli`** command is a command-line tool for configuring network settings with NetworkManager.
- The system's static host name is stored in the **`/etc/hostname`** file. The **`hostnamectl`** command is used to modify or view the status of the system's host name and related settings. The **`hostname`** command displays or temporarily modifies the system's host name.

CHAPTER 11

ANALYZING AND STORING LOGS

GOAL

Locate and accurately interpret logs of system events for troubleshooting purposes.

OBJECTIVES

- Describe the basic logging architecture used by Red Hat Enterprise Linux to record events.
- Interpret events in relevant syslog files to troubleshoot problems or review system status.
- Find and interpret entries in the system journal to troubleshoot problems or review system status.
- Configure the system journal to preserve the record of events when a server is rebooted.
- Maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

SECTIONS

- Describing System Log Architecture (and Quiz)
- Reviewing Syslog Files (and Guided Exercise)
- Reviewing System Journal Entries (and Guided Exercise)
- Preserving the System Journal (and Guided Exercise)
- Maintaining Accurate Time (and Guided Exercise)

LAB

Analyzing and Storing Logs

DESCRIBING SYSTEM LOG ARCHITECTURE

OBJECTIVES

After completing this section, you should be able to describe the basic logging architecture used by Red Hat Enterprise Linux to record events.

SYSTEM LOGGING

Processes and the operating system kernel record a log of events that happen. These logs are used to audit the system and troubleshoot problems.

Many systems record logs of events in text files which are kept in the **`/var/log`** directory. These logs can be inspected using normal text utilities such as **`less`** and **`tail`**.

A standard logging system based on the *Syslog* protocol is built into Red Hat Enterprise Linux. Many programs use this system to record events and organize them into log files. The `systemd-journald` and `rsyslog` services handle the syslog messages in Red Hat Enterprise Linux 8.

The `systemd-journald` service is at the heart of the operating system event logging architecture. It collects event messages from many sources including the kernel, output from the early stages of the boot process, standard output and standard error from daemons as they start up and run, and syslog events. It then restructures them into a standard format, and writes them into a structured, indexed system journal. By default, this journal is stored on a file system that does not persist across reboots.

However, the `rsyslog` service reads syslog messages received by `systemd-journald` from the journal as they arrive. It then processes the syslog events, recording them to its log files or forwarding them to other services according to its own configuration.

The `rsyslog` service sorts and writes syslog messages to the log files that do persist across reboots in **`/var/log`**. The `rsyslog` service sorts the log messages to specific log files based on the type of program that sent each message, or *facility*, and the priority of each syslog message.

In addition to syslog message files, the **`/var/log`** directory contains log files from other services on the system. The following table lists some useful files in the **`/var/log`** directory.

Selected System Log Files

LOG FILE	TYPE OF MESSAGES STORED
`/var/log/messages`	Most syslog messages are logged here. Exceptions include messages related to authentication and email processing, scheduled job execution, and those which are purely debugging-related.
`/var/log/secure`	Syslog messages related to security and authentication events.
`/var/log/maillog`	Syslog messages related to the mail server.
`/var/log/cron`	Syslog messages related to scheduled job execution.

LOG FILE	TYPE OF MESSAGES STORED
`/var/log/boot.log`	Non-syslog console messages related to system startup.

NOTE

Some applications do not use ***syslog*** to manage their log messages, although typically, they do place their log files in a subdirectory of /var/log. For example, the Apache Web Server saves log messages to files in a subddirectory of the **`/var/log`** directory.

REFERENCES

`systemd-journald.service`(8), `rsyslogd`(8), and `rsyslog.conf`(5) man pages

For more information refer to the *Using the log files to troubleshoot problems* section in the *Red Hat Enterprise Linux 8.0 Configuring basic system settings Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#Troubleshoot-log-files_getting-started-with-system-administration

QUIZ

DESCRIBING SYSTEM LOG ARCHITECTURE

Choose the correct answer to the following questions:

1. **Which of these log files stores most syslog messages, with the exception of those that are related to authentication, mail, scheduled jobs, and debugging?**
   a. `/var/log/maillog`
   b. `/var/log/boot.log`
   c. `/var/log/messages`
   d. `/var/log/secure`

2. **Which log file stores syslog messages related to security and authentication operations in the system?**
   a. `/var/log/maillog`
   b. `/var/log/boot.log`
   c. `/var/log/messages`
   d. `/var/log/secure`

3. **Which service sorts and organizes syslog messages into files in `/var/log`?**
   a. `rsyslog`
   b. `systemd-journald`
   c. `auditd`
   d. `tuned`

4. **Which directory accommodates the human-readable syslog files?**
   a. `/sys/kernel/debug`
   b. `/var/log/journal`
   c. `/run/log/journal`
   d. `/var/log`

5. **Which file stores syslog messages related to the mail server?**
   a. `/var/log/lastlog`
   b. `/var/log/maillog`
   c. `/var/log/tallylog`
   d. `/var/log/boot.log`

6. Which file stores syslog messages related to the scheduled jobs?
   a. `/var/log/cron`
   b. `/var/log/tallylog`
   c. `/var/log/spooler`
   d. `/var/log/secure`

7. What file stores console messages related to system startup?
   a. `/var/log/messages`
   b. `/var/log/cron`
   c. `/var/log/boot.log`
   d. `/var/log/secure`

SOLUTION

DESCRIBING SYSTEM LOG ARCHITECTURE

Choose the correct answer to the following questions:

1. **Which of these log files stores most syslog messages, with the exception of those that are related to authentication, mail, scheduled jobs, and debugging?**
   a. `/var/log/maillog`
   b. `/var/log/boot.log`
   c. **`/var/log/messages`**
   d. `/var/log/secure`

2. **Which log file stores syslog messages related to security and authentication operations in the system?**
   a. `/var/log/maillog`
   b. `/var/log/boot.log`
   c. `/var/log/messages`
   d. **`/var/log/secure`**

3. **Which service sorts and organizes syslog messages into files in `/var/log`?**
   a. `rsyslog`
   b. `systemd-journald`
   c. `auditd`
   d. `tuned`

4. **Which directory accommodates the human-readable syslog files?**
   a. `/sys/kernel/debug`
   b. `/var/log/journal`
   c. `/run/log/journal`
   d. **`/var/log`**

5. **Which file stores syslog messages related to the mail server?**
   a. `/var/log/lastlog`
   b. **`/var/log/maillog`**
   c. `/var/log/tallylog`
   d. `/var/log/boot.log`

6. Which file stores syslog messages related to the scheduled jobs?
   a. `/var/log/cron`
   b. `/var/log/tallylog`
   c. `/var/log/spooler`
   d. `/var/log/secure`

7. What file stores console messages related to system startup?
   a. `/var/log/messages`
   b. `/var/log/cron`
   c. `/var/log/boot.log`
   d. `/var/log/secure`

REVIEWING SYSLOG FILES

OBJECTIVES

After completing this section, you should be able to interpret events in relevant syslog files to troubleshoot problems or review system status.

LOGGING EVENTS TO THE SYSTEM

Many programs use the `syslog` protocol to log events to the system. Each log message is categorized by a facility (the type of message) and a priority (the severity of the message). Available facilities are documented in the `rsyslog.conf`(5) man page.

The following table lists the standard eight syslog priorities from highest to lowest.

Overview of Syslog Priorities

CODE	PRIORITY	SEVERITY
0	emerg	System is unusable
1	alert	Action must be taken immediately
2	crit	Critical condition
3	err	Non-critical error condition
4	warning	Warning condition
5	notice	Normal but significant event
6	info	Informational event
7	debug	Debugging-level message

The `rsyslog` service uses the facility and priority of log messages to determine how to handle them. This is configured by rules in the **`/etc/rsyslog.conf`** file and any file in the **`/etc/rsyslog.d`** directory that has a file name extension of **`.conf`**. Software packages can easily add rules by installing an appropriate file in the **`/etc/rsyslog.d`** directory.

Each rule that controls how to sort syslog messages is a line in one of the configuration files. The left side of each line indicates the facility and severity of the syslog messages the rule matches. The right side of each line indicates what file to save the log message in (or where else to deliver the message). An asterisk (**`*`**) is a wildcard that matches all values.

For example, the following line would record messages sent to the `authpriv` facility at any priority to the file **`/var/log/secure`**:

```
authpriv.*                  /var/log/secure
```

Log messages sometimes match more than one rule in **rsyslog.conf**. In such cases, one message is stored in more than one log file. To limit messages stored, the key word **none** in the priority field indicates that no messages for the indicated facility should be stored in the given file.

Instead of logging syslog messages to a file, they can also be printed to the terminals of all logged-in users. The **rsyslog.conf** file has a setting to print all the syslog messages with the **emerg** priority to the terminals of all logged-in users.

SAMPLE RULES OF RSYSLOG

```
#### RULES ####

# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog


# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

NOTE

The syslog subsystem has many more features beyond the scope of this course. For those who wish to explore further, consult the `rsyslog.conf`(5) man page and the extensive HTML documentation in **/usr/share/doc/rsyslog/html/index.html** contained in the *rsyslog-doc* package, available from the AppStream repository in Red Hat Enterprise Linux 8.

LOG FILE ROTATION

The **logrotate** tool rotates log files to keep them from taking up too much space in the file system containing the **/var/log** directory. When a log file is rotated, it is renamed with an extension indicating the date it was rotated. For example, the old **/var/log/messages** file may

become **/var/log/messages-20190130** if it is rotated on 2019-01-30. Once the old log file is rotated, a new log file is created and the service that writes to it is notified.

After a certain number of rotations, typically after four weeks, the oldest log file is discarded to free disk space. A scheduled job runs the **logrotate** program daily to see if any logs need to be rotated. Most log files are rotated weekly, but **logrotate** rotates some faster, or slower, or when they reach a certain size.

Configuration of **logrotate** is not covered in this course. For more information, see the **logrotate**(8) man page.

ANALYZING A SYSLOG ENTRY

Log messages start with the oldest message on top and the newest message at the end of the log file. The `rsyslog` service uses a standard format while recording entries in log files. The following example explains the anatomy of a log message in the **/var/log/secure** log file.

```
❶Feb 11 20:11:48 ❷localhost ❸sshd[1433]: ❹Failed password for student from
 172.25.0.10 port 59344 ssh2
```

❶ The time stamp when the log entry was recorded
❷ The host from which the log message was sent
❸ The program or process name and PID number that sent the log message
❹ The actual message sent

MONITORING LOGS

Monitoring one or more log files for events is helpful to reproduce problems and issues. The **tail -f** ***/path/to/file*** command outputs the last 10 lines of the file specified and continues to output new lines in the file as they get written.

For example, to monitor for failed login attempts, run the **tail** command in one terminal and then in another terminal, run the **ssh** command as the `root` user while a user tries to log in to the system.

In the first terminal, run the following **tail** command:

```
[root@host ~]# tail -f /var/log/secure
```

In the second terminal, run the following **ssh** command:

```
[root@host ~]# ssh root@localhost
root@localhost's password: redhat
...output omitted...
[root@host ~]#
```

Return to the first terminal and view the logs.

```
...output omitted...
Feb 10 09:01:13 host sshd[2712]: Accepted password for root from 172.25.254.254
 port 56801 ssh2
Feb 10 09:01:13 host sshd[2712]: pam_unix(sshd:session): session opened for user
 root by (uid=0)
```

SENDING SYSLOG MESSAGES MANUALLY

The **logger** command can send messages to the **rsyslog** service. By default, it sends the message to the **user** facility with the **notice** priority (**user.notice**) unless specified otherwise with the **-p** option. It is useful to test any change to the rsyslog service configuration.

To send a message to the **rsyslog** service that gets recorded in the **/var/log/boot.log** log file, execute the following **logger** command:

```
[root@host ~]# logger -p local7.notice "Log entry created on host"
```

REFERENCES

logger(1), tail(1), rsyslog.conf(5), and logrotate(8) man pages

rsyslog Manual

- **/usr/share/doc/rsyslog/html/index.html** provided by the *rsyslog-doc* package

For more information refer to the *Using the log files to troubleshoot problems* section in the *Red Hat Enterprise Linux 8.0 Configuring basic system settings Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#Troubleshoot-log-files_getting-started-with-system-administration

GUIDED EXERCISE

REVIEWING SYSLOG FILES

In this exercise, you will reconfigure `rsyslog` to write specific log messages to a new file.

OUTCOMES

You should be able to configure the `rsyslog` service to write all log messages with the **debug** priority to the **/var/log/messages-debug** log file.

BEFORE YOU BEGIN

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

On `workstation`, run **lab log-configure start** to start the exercise. This script ensures that the environment is setup correctly.

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

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

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

2. Configure `rsyslog` on `servera` to log all messages with the **debug** priority, or higher, for any service into the new **/var/log/messages-debug** log file by adding the `rsyslog` configuration file **/etc/rsyslog.d/debug.conf**.

   2.1. Use the **sudo -i** command to switch to the `root` user. Specify `student` as the password for the `student` user if asked while running the **sudo -i** command.

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

   2.2. Create the **/etc/rsyslog.d/debug.conf** file with the necessary entries to redirect all log messages having the **debug** priority to **/var/log/messages-debug**. You may use the **vim /etc/rsyslog.d/debug.conf** command to create the file with the following content.

```
*.debug /var/log/messages-debug
```

   This configuration line catches syslog messages with any facility and a **debug** or above priority level. The `rsyslog` service write those syslog messages to the **/var/log/messages-debug** file. The wildcard (*****) in the **facility** or **priority** fields of the configuration line indicates any facility or priority.

2.3. Restart the `rsyslog` service.

```
[root@servera ~]# systemctl restart rsyslog
```

3. Verify that all the log messages with the **debug** priority appears in the **/var/log/messages-debug** file.

3.1. Use the **logger** command with the **-p** option to generate a log message with the **user** facility and the **debug** priority.

```
[root@servera ~]# logger -p user.debug "Debug Message Test"
```

3.2. Use the **tail** command to view the last ten log messages from the **/var/log/messages-debug** file and confirm that you see the **Debug Message Test** message among the other log messages.

```
[root@servera ~]# tail /var/log/messages-debug
Feb 13 18:22:38 servera systemd[1]: Stopping System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25176]: [origin software="rsyslogd"
 swVersion="8.37.0-9.el8" x-pid="25176" x-info="http://
www.rsyslog.com"] exiting on signal 15.
Feb 13 18:22:38 servera systemd[1]: Stopped System Logging Service.
Feb 13 18:22:38 servera systemd[1]: Starting System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25410]: environment variable TZ is
 not set, auto correcting this to TZ=/etc/localtime  [v8.37.0-9.el8
 try http://www.rsyslog.com/e/2442 ]
Feb 13 18:22:38 servera systemd[1]: Started System Logging Service.
Feb 13 18:22:38 servera rsyslogd[25410]: [origin software="rsyslogd"
 swVersion="8.37.0-9.el8" x-pid="25410" x-info="http://
www.rsyslog.com"] start
Feb 13 18:27:58 servera student[25416]: Debug Message Test
```

3.3. Exit both the `root` and `student` users' shells on `servera` to return to the `student` user's shell on `workstation`.

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

Finish

On `workstation`, run **lab log-configure finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

```
[student@workstation ~]$ lab log-configure finish
```

This concludes the guided exercise.

REVIEWING SYSTEM JOURNAL ENTRIES

OBJECTIVES

After completing this section, you should be able to find and interpret entries in the system journal to troubleshoot problems or review system status.

FINDING EVENTS

The `systemd-journald` service stores logging data in a structured, indexed binary file called the journal. This data includes extra information about the log event. For example, for syslog events this includes the facility and the priority of the original message.

> **IMPORTANT**
> In Red Hat Enterprise Linux 8, the **`/run/log`** directory stores the system journal by default. The contents of the **`/run/log`** directory get cleared after a reboot. You can change this setting, and how to do so is discussed later in this chapter.

To retrieve log messages from the journal, use the **`journalctl`** command. You can use this command to view all messages in the journal, or to search for specific events based on a wide range of options and criteria. If you run the command as `root`, you have full access to the journal. Regular users can also use this command, but might be restricted from seeing certain messages.

```
[root@host ~]# journalctl
...output omitted...
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Stopped target Sockets.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Closed D-Bus User Message Bus
 Socket.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Closed Multimedia System.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Reached target Shutdown.
Feb 21 17:46:25 host.lab.example.com systemd[24263]: Starting Exit the Session...
Feb 21 17:46:25 host.lab.example.com systemd[24268]: pam_unix(systemd-
user:session): session c>
Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopped User Manager for UID
 1001.
Feb 21 17:46:25 host.lab.example.com systemd[1]: user-runtime-dir@1001.service:
 Unit not neede>
Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopping /run/user/1001 mount
 wrapper...
Feb 21 17:46:25 host.lab.example.com systemd[1]: Removed slice User Slice of UID
 1001.
Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopped /run/user/1001 mount
 wrapper.
Feb 21 17:46:36 host.lab.example.com sshd[24434]: Accepted publickey for root from
 172.25.250.>
Feb 21 17:46:37 host.lab.example.com systemd[1]: Started Session 20 of user root.
Feb 21 17:46:37 host.lab.example.com systemd-logind[708]: New session 20 of user
 root.
```

```
Feb 21 17:46:37 host.lab.example.com sshd[24434]: pam_unix(sshd:session): session
 opened for u>
Feb 21 18:01:01 host.lab.example.com CROND[24468]: (root) CMD (run-parts /etc/
cron.hourly)
Feb 21 18:01:01 host.lab.example.com run-parts[24471]: (/etc/cron.hourly) starting
 0anacron
Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished
 0anacron
```

The **journalctl** command highlights important log messages: messages at **notice** or **warning** priority are in bold text while messages at the **error** priority or higher are in red text.

The key to successfully using the journal for troubleshooting and auditing is to limit journal searches to show only relevant output.

By default, **journalctl -n** shows the last 10 log entries. You can adjust this with an optional argument that specifies how many log entries to display. For the last five log entries, run the following **journalctl** command:

```
[root@host ~]# journalctl -n 5
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:01:01 +07.
 --
...output omitted...
Feb 21 17:46:37 host.lab.example.com systemd-logind[708]: New session 20 of user
 root.
Feb 21 17:46:37 host.lab.example.com sshd[24434]: pam_unix(sshd:session): session
 opened for u>
Feb 21 18:01:01 host.lab.example.com CROND[24468]: (root) CMD (run-parts /etc/
cron.hourly)
Feb 21 18:01:01 host.lab.example.com run-parts[24471]: (/etc/cron.hourly) starting
 0anacron
Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished
 0anacron
lines 1-6/6 (END) q
```

Similar to the **tail -f** command, the **journalctl -f** command outputs the last 10 lines of the system journal and continues to output new journal entries as they get written to the journal. To exit the **journalctl -f** process, use the **Ctrl+C** key combination.

```
[root@host ~]# journalctl -f
-- Logs begin at Wed 2019-02-20 16:01:17 +07. --
...output omitted...
Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished
 0anacron
Feb 21 18:22:42 host.lab.example.com sshd[24437]: Received disconnect from
 172.25.250.250 port 48710:11: disconnected by user
Feb 21 18:22:42 host.lab.example.com sshd[24437]: Disconnected from user root
 172.25.250.250 port 48710
Feb 21 18:22:42 host.lab.example.com sshd[24434]: pam_unix(sshd:session): session
 closed for user root
Feb 21 18:22:42 host.lab.example.com systemd-logind[708]: Session 20 logged out.
 Waiting for processes to exit.
Feb 21 18:22:42 host.lab.example.com systemd-logind[708]: Removed session 20.
```

```
Feb 21 18:22:43 host.lab.example.com sshd[24499]: Accepted
 publickey for root from 172.25.250.250 port 48714 ssh2: RSA
 SHA256:1UGybTe52L2jzEJa1HLVKn9QUCKrTv3ZzxnMJol1Fro
Feb 21 18:22:44 host.lab.example.com systemd-logind[708]: New session 21 of user
 root.
Feb 21 18:22:44 host.lab.example.com systemd[1]: Started Session 21 of user root.
Feb 21 18:22:44 host.lab.example.com sshd[24499]: pam_unix(sshd:session): session
 opened for user root by (uid=0)
^C
```

To help troubleshoot problems, you might want to filter the output of the journal based on the priority of the journal entries. The **`journalctl -p`** takes either the name or the number of a priority level and shows the journal entries for entries at that priority and above. The **`journalctl`** command understands the **`debug`**, **`info`**, **`notice`**, **`warning`**, **`err`**, **`crit`**, **`alert`**, and **`emerg`** priority levels.

Run the following **`journalctl`** command to list journal entries at the **`err`** priority or higher:

```
[root@host ~]# journalctl -p err
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:01:01 +07.
 --
..output omitted...
Feb 20 16:01:17 host.lab.example.com kernel: Detected CPU family 6 model 13
 stepping 3
Feb 20 16:01:17 host.lab.example.com kernel: Warning: Intel Processor - this
 hardware has not undergone testing by Red Hat and might not be certif>
Feb 20 16:01:20 host.lab.example.com smartd[669]: DEVICESCAN failed: glob(3)
 aborted matching pattern /dev/discs/disc*
Feb 20 16:01:20 host.lab.example.com smartd[669]: In the system's table of devices
 NO devices found to scan
lines 1-5/5 (END) q
```

When looking for specific events, you can limit the output to a specific time frame. The **`journalctl`** command has two options to limit the output to a specific time range, the **`--since`** and **`--until`** options. Both options take a time argument in the format *"YYYY-MM-DD hh:mm:ss"* (the double-quotes are required to preserve the space in the option). If the date is omitted, the command assumes the current day, and if the time is omitted, the command assumes the whole day starting at 00:00:00. Both options take **`yesterday`**, **`today`**, and **`tomorrow`** as valid arguments in addition to the date and time field.

Run the following **`journalctl`** command to list all journal entries from today's records.

```
[root@host ~]# journalctl --since today
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:31:14 +07.
 --
...output omitted...
Feb 21 18:22:44 host.lab.example.com systemd-logind[708]: New session 21 of user
 root.
Feb 21 18:22:44 host.lab.example.com systemd[1]: Started Session 21 of user root.
Feb 21 18:22:44 host.lab.example.com sshd[24499]: pam_unix(sshd:session): session
 opened for user root by (uid=0)
Feb 21 18:31:13 host.lab.example.com systemd[1]: Starting dnf makecache...
Feb 21 18:31:14 host.lab.example.com dnf[24533]: Red Hat Enterprise Linux 8.0
 AppStream (dvd)    637 kB/s | 2.8 kB     00:00
```

```
Feb 21 18:31:14 host.lab.example.com dnf[24533]: Red Hat Enterprise Linux 8.0
 BaseOS (dvd)       795 kB/s | 2.7 kB     00:00
Feb 21 18:31:14 host.lab.example.com dnf[24533]: Metadata cache created.
Feb 21 18:31:14 host.lab.example.com systemd[1]: Started dnf makecache.
```

Run the following **journalctl** command to list all journal entries ranging from **2019-02-10 20:30:00** to **2019-02-13 12:00:00**.

```
[root@host ~]# journalctl --since "2019-02-10 20:30:00" \
--until "2019-02-13 12:00:00"
...output omitted...
```

You can also specify all entries since a time relative to the present. For example, to specify all entries in the last hour, you can use the following command:

```
[root@host ~]# journalctl --since "-1 hour"
...output omitted...
```

NOTE

You can use other, more sophisticated time specifications with the **--since** and **--until** options. For some examples, see the `systemd.time`(7) man page.

In addition to the visible content of the journal, there are fields attached to the log entries that can only be seen when verbose output is turned on. Any displayed extra field can be used to filter the output of a journal query. This is useful to reduce the output of complex searches for certain events in the journal.

```
[root@host ~]# journalctl -o verbose
-- Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:31:14 +07.
 --
...output omitted...
Thu 2019-02-21 18:31:14.509128 +07...
    PRIORITY=6
    _BOOT_ID=4409bbf54680496d94e090de9e4a9e23
    _MACHINE_ID=73ab164e278e48be9bf80e80714a8cd5
    SYSLOG_FACILITY=3
    SYSLOG_IDENTIFIER=systemd
    _UID=0
    _GID=0
    CODE_FILE=../src/core/job.c
    CODE_LINE=826
    CODE_FUNC=job_log_status_message
    JOB_TYPE=start
    JOB_RESULT=done
    MESSAGE_ID=39f53479d3a045ac8e11786248231fbf
    _TRANSPORT=journal
    _PID=1
    _COMM=systemd
    _EXE=/usr/lib/systemd/systemd
    _CMDLINE=/usr/lib/systemd/systemd --switched-root --system --deserialize 18
    _CAP_EFFECTIVE=3ffffffffff
```

```
_SELINUX_CONTEXT=system_u:system_r:init_t:s0
_SYSTEMD_CGROUP=/init.scope
_SYSTEMD_UNIT=init.scope
_SYSTEMD_SLICE=-.slice
UNIT=dnf-makecache.service
MESSAGE=Started dnf makecache.
_HOSTNAME=host.lab.example.com
INVOCATION_ID=d6f90184663f4309835a3e8ab647cb0e
_SOURCE_REALTIME_TIMESTAMP=1550748674509128
```

The following list gives the common fields of the system journal that can be used to search for lines relevant to a particular process or event.

- _COMM is the name of the command
- _EXE is the path to the executable for the process
- _PID is the PID of the process
- _UID is the UID of the user running the process
- _SYSTEMD_UNIT is the systemd unit that started the process

More than one of the system journal fields can be combined to form a granular search query with the **`journalctl`** command. For example, the following **`journalctl`** command shows all journal entries related to the **`sshd.service`** `systemd` unit from a process with PID 1182.

```
[root@host ~]# journalctl _SYSTEMD_UNIT=sshd.service _PID=1182
Apr 03 19:34:27 host.lab.example.com sshd[1182]: Accepted password for root
 from ::1 port 52778 ssh2
Apr 03 19:34:28 host.lab.example.com sshd[1182]: pam_unix(sshd:session): session
 opened for user root by (uid=0)
...output omitted...
```

NOTE

For a list of commonly used journal fields, consult the `systemd.journal-fields`(7) man page.

REFERENCES

`journalctl`(1), `systemd.journal-fields`(7), and `systemd.time`(7) man pages

For more information refer to the *Using the log files to troubleshoot problems* section in the *Red Hat Enterprise Linux 8.0 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#Troubleshoot-log-files_getting-started-with-system-administration

GUIDED EXERCISE

REVIEWING SYSTEM JOURNAL ENTRIES

In this exercise, you will search the system journal for entries recording events that match specific criteria.

OUTCOMES

You should be able to search the system journal for entries recording events based on different criteria.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab log-query start`** to start the exercise. This script ensures that the environment is setup correctly.

```
[student@workstation ~]$ lab log-query start
```

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

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

2. Use the **`_PID=1`** match with the **`journalctl`** command to display only log events originating from the `systemd` process running with the process identifier of 1 on `servera`. To quit **`journalctl`**, press **`q`**.

```
[student@servera ~]$ journalctl _PID=1
...output omitted...
Feb 13 13:21:08 localhost systemd[1]: Found device /dev/disk/by-uuid/
cdf61ded-534c-4bd6-b458-cab18b1a72ea.
Feb 13 13:21:08 localhost systemd[1]: Started dracut initqueue hook.
Feb 13 13:21:08 localhost systemd[1]: Found device /dev/disk/by-
uuid/44330f15-2f9d-4745-ae2e-20844f22762d.
Feb 13 13:21:08 localhost systemd[1]: Reached target Initrd Root Device.
lines 1-5/5 (END) q
[student@servera ~]$
```

NOTE

The **`journalctl`** command may produce a different output on your system.

3. Use the **_UID=81** match with the **journalctl** command to display all log events originating from a system service started with the user identifier of 81 on servera. To quit **journalctl** press **q**.

```
[student@servera ~]$ journalctl _UID=81
...output omitted...
Feb 22 01:29:09 servera.lab.example.com dbus-daemon[672]: [system]
 Activating via systemd: service name='org.freedesktop.nm_dispatcher'>
Feb 22 01:29:09 servera.lab.example.com dbus-daemon[672]: [system]
 Successfully activated service 'org.freedesktop.nm_dispatcher'
lines 1-5/5 (END) q
[student@servera ~]$
```

4. Use the **-p warning** option with the **journalctl** command to display log events with priority **warning** and above on servera. To quit **journalctl** press **q**.

```
[student@servera ~]$ journalctl -p warning
...output omitted...
Feb 13 13:21:07 localhost kernel: Detected CPU family 6 model 13 stepping 3
Feb 13 13:21:07 localhost kernel: Warning: Intel Processor - this hardware
 has not undergone testing by Red Hat and might not >
Feb 13 13:21:07 localhost kernel: acpi PNP0A03:00: fail to add MMCONFIG
 information, can't access extended PCI configuration s>
Feb 13 13:21:07 localhost rpc.statd[288]: Running as root.  chown /var/lib/
nfs/statd to choose different user
Feb 13 13:21:07 localhost rpc.idmapd[293]: Setting log level to 0
...output omitted...
Feb 13 13:21:13 servera.lab.example.com rsyslogd[1172]: environment variable
 TZ is not set, auto correcting this to TZ=/etc/lo>
Feb 13 14:51:42 servera.lab.example.com systemd[1]: cgroup compatibility
 translation between legacy and unified hierarchy sett>
Feb 13 17:15:37 servera.lab.example.com rsyslogd[25176]: environment
 variable TZ is not set, auto correcting this to TZ=/etc/l>
Feb 13 18:22:38 servera.lab.example.com rsyslogd[25410]: environment
 variable TZ is not set, auto correcting this to TZ=/etc/l>
Feb 13 18:47:55 servera.lab.example.com rsyslogd[25731]: environment
 variable TZ is not set, auto correcting this to TZ=/etc/l>
lines 1-17/17 (END) q
```

5. Display all log events recorded in the past 10 minutes from the current time on servera.

   5.1. Use the **--since** option with the **journalctl** command to display all log events recorded in the past 10 minutes on servera. To quit **journalctl** press **q**.

```
[student@servera ~]$ journalctl --since "-10min"
...output omitted...
Feb 13 22:31:01 servera.lab.example.com CROND[25890]: (root) CMD (run-
parts /etc/cron.hourly)
Feb 13 22:31:01 servera.lab.example.com run-parts[25893]: (/etc/
cron.hourly) starting 0anacron
Feb 13 22:31:01 servera.lab.example.com run-parts[25899]: (/etc/
cron.hourly) finished 0anacron
```

```
Feb 13 22:31:41 servera.lab.example.com sshd[25901]: Bad protocol
 version identification 'brain' from 172.25.250.254 port 37450
Feb 13 22:31:42 servera.lab.example.com sshd[25902]: Accepted
 publickey for root from 172.25.250.254 port 37452 ssh2: RSA SHA2>
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Started /run/
user/0 mount wrapper.
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Created slice User
 Slice of UID 0.
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Starting User
 Manager for UID 0...
Feb 13 22:31:42 servera.lab.example.com systemd[1]: Started Session
 118 of user root.
Feb 13 22:31:42 servera.lab.example.com systemd-logind[712]: New
 session 118 of user root.
Feb 13 22:31:42 servera.lab.example.com systemd[25906]:
 pam_unix(systemd-user:session): session opened for user root by
 (uid=0)
...output omitted...
lines 1-32/84 39% q
```

6. Use the **--since** option and the **_SYSTEMD_UNIT="sshd.service"** match with the **journalctl** command to display all the log events originating from the sshd service recorded since **09:00:00** this morning on servera. To quit **journalctl** press **q**.

```
[student@servera ~]$ journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"
...output omitted...
Feb 13 13:21:12 servera.lab.example.com sshd[727]: Server listening on
 0.0.0.0 port 22.
Feb 13 13:21:12 servera.lab.example.com sshd[727]: Server listening on ::
 port 22.
Feb 13 13:22:07 servera.lab.example.com sshd[1238]: Accepted publickey for
 student from 172.25.250.250 port 50590 ssh2: RSA SH>
Feb 13 13:22:07 servera.lab.example.com sshd[1238]: pam_unix(sshd:session):
 session opened for user student by (uid=0)
Feb 13 13:22:08 servera.lab.example.com sshd[1238]: pam_unix(sshd:session):
 session closed for user student
Feb 13 13:25:47 servera.lab.example.com sshd[1289]: Accepted publickey for
 root from 172.25.250.254 port 37194 ssh2: RSA SHA25>
Feb 13 13:25:47 servera.lab.example.com sshd[1289]: pam_unix(sshd:session):
 session opened for user root by (uid=0)
Feb 13 13:25:47 servera.lab.example.com sshd[1289]: pam_unix(sshd:session):
 session closed for user root
Feb 13 13:25:48 servera.lab.example.com sshd[1316]: Accepted publickey for
 root from 172.25.250.254 port 37196 ssh2: RSA SHA25>
Feb 13 13:25:48 servera.lab.example.com sshd[1316]: pam_unix(sshd:session):
 session opened for user root by (uid=0)
Feb 13 13:25:48 servera.lab.example.com sshd[1316]: pam_unix(sshd:session):
 session closed for user root
Feb 13 13:26:07 servera.lab.example.com sshd[1355]: Accepted publickey for
 student from 172.25.250.254 port 37198 ssh2: RSA SH>
Feb 13 13:26:07 servera.lab.example.com sshd[1355]: pam_unix(sshd:session):
 session opened for user student by (uid=0)
```

```
Feb 13 13:52:28 servera.lab.example.com sshd[1473]: Accepted publickey for
 root from 172.25.250.254 port 37218 ssh2: RSA SHA25>
Feb 13 13:52:28 servera.lab.example.com sshd[1473]: pam_unix(sshd:session):
 session opened for user root by (uid=0)
...output omitted...
lines 1-32 q
```

7. Log out of `servera`.

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

Finish

On `workstation`, run **`lab log-query finish`** to complete this exercise. This script ensures that the environment is restored back to the clean state.

```
[student@workstation ~]$ lab log-query finish
```

This concludes the guided exercise.

PRESERVING THE SYSTEM JOURNAL

OBJECTIVES

After completing this section, you should be able to configure the system journal to preserve the record of events when a server is rebooted.

STORING THE SYSTEM JOURNAL PERMANENTLY

By default, the system journals are kept in the **`/run/log/journal`** directory, which means the journals are cleared when the system reboots. You can change the configuration settings of the `systemd-journald` service in the **`/etc/systemd/journald.conf`** file to make the journals persist across reboot.

The `Storage` parameter in the **`/etc/systemd/journald.conf`** file defines whether to store system journals in a volatile manner or persistently across reboot. Set this parameter to **`persistent`**, **`volatile`**, or **`auto`** as follows:

- **`persistent`**: stores journals in the **`/var/log/journal`** directory which persists across reboots.

  If the **`/var/log/journal`** directory does not exist, the `systemd-journald` service creates it.

- **`volatile`**: stores journals in the volatile **`/run/log/journal`** directory.

  As the **`/run`** file system is temporary and exists only in the runtime memory, data stored in it, including system journals, do not persist across reboot.

- **`auto`**: ***rsyslog*** determines whether to use persistent or volatile storage. If the **`/var/log/journal`** directory exists, then ***rsyslog*** uses persistent storage, otherwise it uses volatile storage.

  This is the default action if the `Storage` parameter is not set.

The advantage of persistent system journals is that the historic data is available immediately at boot. However, even with a persistent journal, not all data is kept forever. The journal has a built-in log rotation mechanism that triggers monthly. In addition, by default, the journals are not allowed to get larger than 10% of the file system it is on, or leave less than 15% of the file system free. These values can be tuned for both the runtime and persistent journals in **`/etc/systemd/journald.conf`**. The current limits on the size of the journal are logged when the `systemd-journald` process starts. The following command output shows the journal entries that reflect the current size limits:

```
[user@host ~]$ journalctl | grep -E 'Runtime|System journal'
Feb 25 13:01:46 localhost systemd-journald[147]: Runtime journal (/run/log/
journal/ae06db7da89142138408d77efea9229c) is 8.0M, max 91.4M, 83.4M free.
Feb 25 13:01:48 remotehost.lab.example.com systemd-journald[548]: Runtime journal
 (/run/log/journal/73ab164e278e48be9bf80e80714a8cd5) is 8.0M, max 91.4M, 83.4M
 free.
Feb 25 13:01:48 remotehost.lab.example.com systemd-journald[548]: System journal
 (/var/log/journal/73ab164e278e48be9bf80e80714a8cd5) is 8.0M, max 3.7G, 3.7G free.
Feb 25 13:01:48 remotehost.lab.example.com systemd[1]: Starting Tell Plymouth To
 Write Out Runtime Data...
Feb 25 13:01:48 remotehost.lab.example.com systemd[1]: Started Tell Plymouth To
 Write Out Runtime Data.
```

NOTE

In the **grep** above, the pipe (**|**) symbol acts as an *or* indicator. That is, **grep** matches any line containing either the **Runtime** string or the **System** string from the **journalctl** output. This fetches the current size limits on the volatile (**Runtime**) journal store as well the persistent (**System**) journal store.

Configuring Persistent System Journals

To configure the `systemd-journald` service to preserve system journals persistently across reboot, set `Storage` to **persistent** in the **/etc/systemd/journald.conf** file. Run the text editor of your choice as the superuser to edit the **/etc/systemd/journald.conf** file.

```
[Journal]
Storage=persistent
...output omitted...
```

After editing the configuration file, restart the `systemd-journald` service to bring the configuration changes into effect.

```
[root@host ~]# systemctl restart systemd-journald
```

If the `systemd-journald` service successfully restarts, you can see that the **/var/log/journal** directory is created and contains one or more subdirectories. These subdirectories have hexadecimal characters in their long names and contain ***.journal** files. The ***.journal** files are the binary files that store the structured and indexed journal entries.

```
[root@host ~]# ls /var/log/journal
73ab164e278e48be9bf80e80714a8cd5
[root@host ~]# ls /var/log/journal/73ab164e278e48be9bf80e80714a8cd5
system.journal  user-1000.journal
```

While the system journals persist across reboot, you get an extensive number of entries in the output of the **journalctl** command that includes entries from the current system boot as well as the previous ones. To limit the output to a specific system boot, use the **-b** option with the **journalctl** command. The following **journalctl** command retrieves the entries limited to the first system boot:

```
[root@host ~]# journalctl -b 1
...output omitted...
```

The following **journalctl** command retrieves the entries limited to the second system boot. The following argument is meaningful only if the system has been rebooted for more than twice:

```
[root@host ~]# journalctl -b 2
```

The following **journalctl** command retrieves the entries limited to the current system boot:

```
[root@host ~]# journalctl -b
```

NOTE

When debugging a system crash with a persistent journal, it is usually required to limit the journal query to the reboot before the crash happened. The **-b** option can be accompanied by a negative number indicating how many prior system boots the output should include. For example, **journalctl -b -1** limits the output to only the previous boot.

REFERENCES

`systemd-journald.conf`(5), `systemd-journald`(8) man pages

For more information refer to the *Using the log files to troubleshoot problems* section in the *Red Hat Enterprise Linux 8.0 Configuring basic system settings Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#Troubleshoot-log-files_getting-started-with-system-administration

GUIDED EXERCISE

PRESERVING THE SYSTEM JOURNAL

In this exercise, you will configure the system journal to preserve its data after a reboot.

OUTCOMES

You should be able to configure the system journal to preserve its data after a reboot.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab log-preserve start`** to start the exercise. This script ensures that the environment is set up correctly.

```
[student@workstation ~]$ lab log-preserve start
```

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

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

2. As the superuser, confirm that the **`/var/log/journal`** directory does not exist. Use the **`ls`** command to list the **`/var/log/journal`** directory contents. Use **`sudo`** to elevate the `student` user privileges. Use `student` as the password if asked.

```
[student@servera ~]$ sudo ls /var/log/journal
[sudo] password for student: student
ls: cannot access '/var/log/journal': No such file or directory
```

Since the **`/var/log/journal`** directory does not exist, `systemd-journald` service is not preserving its journals.

3. Configure the `systemd-journald` service on `servera` to preserve journals across a reboot.

   3.1. Uncomment the **`Storage=auto`** line in the **`/etc/systemd/journald.conf`** file and set `Storage` to **`persistent`**. You may use the **`sudo vim /etc/systemd/journald.conf`** command to edit the configuration file. Type **`/ Storage=auto`** from **`vim`** command mode to search for the **`Storage=auto`** line.

```
...output omitted...
[Journal]
Storage=persistent
...output omitted...
```

3.2. Use the **systemctl** command to restart the systemd-journald service to bring the configuration changes into effect.

```
[student@servera ~]$ sudo systemctl restart systemd-journald.service
```

4. Confirm that the systemd-journald service on servera preserves its journals such that the journals persist across reboots.

4.1. Use the **systemctl reboot** command to restart servera.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

Notice that the SSH connection was terminated as soon as you restarted the servera system.

4.2. Open an SSH session to servera again.

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

4.3. Use the **ls** command to confirm that the **/var/log/journal** directory exists. The **/var/log/journal** directory contains a subdirectory with a long hexadecimal name. The journal files are found in that directory. The subdirectory name on your system will be different.

```
[student@servera ~]$ sudo ls /var/log/journal
[sudo] password for student: student
73ab164e278e48be9bf80e80714a8cd5
[student@servera ~]$ sudo ls /var/log/
journal/73ab164e278e48be9bf80e80714a8cd5
system.journal  user-1000.journal
```

4.4. Log out of servera.

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

Finish

On workstation, run **lab log-preserve finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

```
[student@workstation ~]$ lab log-preserve finish
```

This concludes the guided exercise.

MAINTAINING ACCURATE TIME

OBJECTIVES

After completing this section, you should be able to maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

SETTING LOCAL CLOCKS AND TIME ZONES

Correct synchronized system time is critical for log file analysis across multiple systems. The *Network Time Protocol* (*NTP*) is a standard way for machines to provide and obtain correct time information on the Internet. A machine may get accurate time information from public NTP services on the Internet, such as the NTP Pool Project. A high-quality hardware clock to serve accurate time to local clients is another option.

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

```
[user@host ~]$ timedatectl
               Local time: Fri 2019-04-05 16:10:29 CDT
           Universal time: Fri 2019-04-05 21:10:29 UTC
                 RTC time: Fri 2019-04-05 21:10:29
                Time zone: America/Chicago (CDT, -0500)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

A database of time zones is available and can be listed with the **`timedatectl list-timezones`** command.

```
[user@host ~]$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
...
```

Time zone names are based on the public time zone database that IANA maintains. Time zones are named based on continent or ocean, then typically but not always the largest city within the time zone region. For example, most of the US Mountain time zone is America/Denver.

Selecting the correct name can be non-intuitive in cases where localities inside the time zone have different daylight saving time rules. For example, in the USA, much of the state of Arizona (US Mountain time) does not have a daylight saving time adjustment at all and is in the time zone America/Phoenix.

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

The superuser can change the system setting to update the current time zone using the **timedatectl set-timezone** command. The following **timedatectl** command updates the current time zone to **America/Phoenix**.

```
[root@host ~]# timedatectl set-timezone America/Phoenix
[root@host ~]# timedatectl
               Local time: Fri 2019-04-05 14:12:39 MST
           Universal time: Fri 2019-04-05 21:12:39 UTC
                 RTC time: Fri 2019-04-05 21:12:39
                Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

NOTE

Should you need to use the Coordinated Universal Time (UTC) on a particular server, set its time zone to UTC. The **tzselect** command does not include the name of the UTC time zone. Use the **timedatectl set-timezone UTC** command to set the system's current time zone to **UTC**.

Use the **timedatectl set-time** command to change the system's current time. The time is specified in the *"YYYY-MM-DD hh:mm:ss"* format, where either date or time can be omitted. The following **timedatectl** command changes the time to **09:00:00**.

```
[root@host ~]# timedatectl set-time 9:00:00
[root@host ~]# timedatectl
               Local time: Fri 2019-04-05 09:00:27 MST
           Universal time: Fri 2019-04-05 16:00:27 UTC
                 RTC time: Fri 2019-04-05 16:00:27
                Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

The **timedatectl set-ntp** command enables or disables NTP synchronization for automatic time adjustment. The option requires either a **true** or **false** argument to turn it on or off. The following **timedatectl** command turns on NTP synchronization.

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

NOTE

In Red Hat Enterprise Linux 8, the **`timedatectl set-ntp`** command will adjust whether or not `chronyd` NTP service is operating. Other Linux distributions might use this setting to adjust a different NTP or SNTP service.

Enabling or disabling NTP using other utilities in Red Hat Enterprise Linux, such as in the graphical GNOME ***Settings*** application, also updates this setting.

CONFIGURING AND MONITORING CHRONYD

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

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

The *stratum* of the NTP time source determines its quality. The stratum determines the number of hops the machine is away from a high-performance reference clock. The reference clock is a **`stratum 0`** time source. An NTP server directly attached to it is a **`stratum 1`**, while a machine synchronizing time from the NTP server is a **`stratum 2`** time source.

The *server* and *peer* are the two categories of time sources that you can in the **`/etc/chrony.conf`** configuration file. The server is one stratum above the local NTP server, and the peer is at the same stratum level. More than one server and more than one peer can be specified, one per line.

The first argument of the **`server`** line is the IP address or DNS name of the NTP server. Following the server IP address or name, a series of options for the server can be listed. It is recommended to use the **`iburst`** option, because after the service starts, four measurements are taken in a short time period for a more accurate initial clock synchronization.

The following **`server classroom.example.com iburst`** line in the **`/etc/chrony.conf`** file causes the `chronyd` service to use the `classroom.example.com` NTP time source.

```
# Use public servers from the pool.ntp.org project.
...output omitted...
server classroom.example.com iburst
...output omitted...
```

After pointing **`chronyd`** to the local time source, `classroom.example.com`, you should restart the service.

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

The **`chronyc`** command acts as a client to the `chronyd` service. After setting up NTP synchronization, you should verify that the local system is seamlessly using the NTP server to synchronize the system clock using the **`chrony sources`** command. For more verbose output with additional explanations about the output, use the **`chronyc sources -v`** command.

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

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too variable.
||                                                 .- xxxx [ yyyy ] +/- zzzz
||                                                /   xxxx = adjusted offset,
||         Log2(Polling interval) -.             |    yyyy = measured offset,
||                                  \            |    zzzz = estimated error.
||                                   |           |
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* classroom.example.com         8   6    17    23   -497ns[-7000ns] +/-  956us
```

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

REFERENCES

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

NTP Pool Project
http://www.pool.ntp.org/

Time Zone Database
http://www.iana.org/time-zones

GUIDED EXERCISE

MAINTAINING ACCURATE TIME

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

OUTCOMES

You should be able to:

- Change the time zone on a server.
- Configure the server to synchronize its time with an NTP time source.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab log-maintain start`** to start the exercise. This script ensures that the time synchronization is disabled on the `servera` system to provide you with the opportunity to manually update the settings on the system and enable the time synchronization.

```
[student@workstation ~]$ lab log-maintain start
```

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

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

2. For the sake of the activity, pretend that the `servera` system is relocated to Haiti and so you need to update the time zone appropriately. Use **sudo** to elevate the privileges of the `student` user while running the **`timedatectl`** command to update the time zone. Use `student` as the password if asked.

   2.1. Use the **`tzselect`** command to determine the appropriate time zone for Haiti.

```
[student@servera ~]$ tzselect
Please identify a location so that time zone rules can be set
 correctly.
Please select a continent, ocean, "coord", or "TZ".
 1) Africa
 2) Americas
 3) Antarctica
 4) Asia
 5) Atlantic Ocean
 6) Australia
 7) Europe
```

```
 8) Indian Ocean
 9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the time zone using the Posix TZ format.
#? 2
Please select a country whose clocks agree with yours.
 1) Anguilla              19) Dominican Republic    37) Peru
 2) Antigua & Barbuda     20) Ecuador               38) Puerto Rico
 3) Argentina             21) El Salvador           39) St Barthelemy
 4) Aruba                 22) French Guiana         40) St Kitts &
 Nevis
 5) Bahamas               23) Greenland             41) St Lucia
 6) Barbados              24) Grenada               42) St Maarten
 (Dutch)
 7) Belize                25) Guadeloupe            43) St Martin
 (French)
 8) Bolivia               26) Guatemala             44) St Pierre &
 Miquelon
 9) Brazil                27) Guyana                45) St Vincent
10) Canada                28) Haiti                 46) Suriname
11) Caribbean NL          29) Honduras              47) Trinidad &
 Tobago
12) Cayman Islands        30) Jamaica               48) Turks & Caicos
 Is
13) Chile                 31) Martinique            49) United States
14) Colombia              32) Mexico                50) Uruguay
15) Costa Rica            33) Montserrat            51) Venezuela
16) Cuba                  34) Nicaragua             52) Virgin Islands
 (UK)
17) Curaçao               35) Panama                53) Virgin Islands
 (US)
18) Dominica              36) Paraguay
#? 28
The following information has been given:

 Haiti

Therefore TZ='America/Port-au-Prince' will be used.
Selected time is now: Tue Feb 19 00:51:05 EST 2019.
Universal Time is now: Tue Feb 19 05:51:05 UTC 2019.
Is the above information OK?
1) Yes
2) No
#? 1

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

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

Notice that the preceding **tzselect** command displayed the appropriate time zone for Haiti.

2.2. Use the **timedatectl** command to update the time zone on servera to **America/Port-au-Prince**.

```
[student@servera ~]$ sudo timedatectl set-timezone America/Port-au-
Prince
[sudo] password for student: student
```

2.3. Use the **timedatectl** command to verify that the time zone has been updated to **America/Port-au-Prince**.

```
[student@servera ~]$ timedatectl
           Local time: Tue 2019-02-19 01:16:29 EST
           Universal time: Tue 2019-02-19 06:16:29 UTC
                 RTC time: Tue 2019-02-19 06:16:29
                 Time zone: America/Port-au-Prince (EST, -0500)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: no
```

3. Configure the chronyd service on servera to synchronize the system time with the NTP time source classroom.example.com.

3.1. Edit the **/etc/chrony.conf** file to specify the classroom.example.com server as the NTP time source. You may use the **sudo vim /etc/chrony.conf** command to edit the configuration file. The following output shows the configuration line you must add to the configuration file:

```
...output omitted...
server classroom.example.com iburst
...output omitted...
```

The preceding line in the **/etc/chrony.conf** configuration file includes the **iburst** option to speed up initial time synchronization.

3.2. Use the **timedatectl** command to turn on the time synchronization on servera.

```
[student@servera ~]$ sudo timedatectl set-ntp yes
```

The preceding **timedatectl** command activates the NTP server with the changed settings in the **/etc/chrony.conf** configuration file. The preceding **timedatectl** command may activate either the chronyd or the ntpd service, based on what is currently installed on the system.

4. Verify that the time settings on `servera` are currently configured to synchronize with the `classroom.example.com` time source in the classroom environment.

4.1. Use the **`timedatectl`** command to verify that the `servera` currently has the time synchronization enabled.

```
[student@servera ~]$ timedatectl
               Local time: Tue 2019-02-19 01:52:17 EST
           Universal time: Tue 2019-02-19 06:52:17 UTC
                 RTC time: Tue 2019-02-19 06:52:17
                Time zone: America/Port-au-Prince (EST, -0500)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

NOTE

If the preceding output shows that the clock is not synchronized, wait for two seconds and re-run the **`timedatectl`** command. It takes a few seconds to successfully synchronize the time settings with the time source.

4.2. Use the **`chronyc`** command to verify that the `servera` system is currently synchronizing its time settings with the `classroom.example.com` time source.

```
[student@servera ~]$ chronyc sources -v
210 Number of sources = 1

  .-- Source mode  '^' = server, '=' = peer, '#' = local clock.
 / .- Source state '*' = current synced, '+' = combined , '-' = not
 combined,
| /   '?' = unreachable, 'x' = time may be in error, '~' = time too
 variable.
||                                                 .- xxxx [ yyyy ]
 +/- zzzz
||      Reachability register (octal) -.           |  xxxx = adjusted
 offset,
||      Log2(Polling interval) --.      |          |  yyyy = measured
 offset,
```

```
||                                 \     |          |  zzzz = estimated
 error.
||                                  |    |           \
MS Name/IP address         Stratum Poll Reach LastRx Last sample

===============================================================================
```

Notice that the preceding output shows an asterisk (*) in the source state (**S**) field for the `classroom.example.com` NTP time source. The asterisk indicates that the local system time is currently in successful synchronization with the NTP time source.

4.3. Log out of `servera`.

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

Finish

On `workstation`, run **`lab log-maintain finish`** to complete this exercise. This script ensures that the original time zone is restored along with all the original time settings on `servera`.

```
[student@workstation ~]$ lab log-maintain finish
```

This concludes the guided exercise.

▷ LAB

ANALYZING AND STORING LOGS

PERFORMANCE CHECKLIST

In this lab, you will change the time zone on an existing server and configure a new log file for all events related to authentication failures.

OUTCOMES

You should be able to:

- Update the time zone on an existing server.
- Configure a new log file to store all messages related to authentication failures.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab log-review start`** to start the exercise. This script records the current time zone of the `serverb` system and ensures that the environment is setup correctly.

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

1. From `workstation`, open an SSH session to `serverb` as `student`.
2. Pretend that the `serverb` system has been relocated to Jamaica and you must update the time zone appropriately. Use **`sudo`** to elevate the `student` user privileges for the **`timedatectl`** command to update the time zone. Use `student` as the password if asked.
3. Display the log events recorded in the previous 30 minutes on `serverb`.
4. Create the **`/etc/rsyslog.d/auth-errors.conf`** file, configured to have the `rsyslog` service write messages related to authentication and security issues to the new **`/var/log/auth-errors`** file. Use the **`authpriv`** facility and the **`alert`** priority in the configuration file.

Evaluation

On `workstation`, run the **`lab log-review grade`** command to confirm success of this exercise.

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

Finish

On `workstation`, run **`lab log-review finish`** to complete this lab. This script ensures that the original time zone is restored along with all the original time settings on `serverb`.

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

This concludes the guided exercise.

▷ SOLUTION

ANALYZING AND STORING LOGS

PERFORMANCE CHECKLIST

In this lab, you will change the time zone on an existing server and configure a new log file for all events related to authentication failures.

OUTCOMES

You should be able to:

- Update the time zone on an existing server.
- Configure a new log file to store all messages related to authentication failures.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab log-review start`** to start the exercise. This script records the current time zone of the `serverb` system and ensures that the environment is setup correctly.

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

1. From `workstation`, open an SSH session to `serverb` as `student`.

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

2. Pretend that the `serverb` system has been relocated to Jamaica and you must update the time zone appropriately. Use **sudo** to elevate the `student` user privileges for the **`timedatectl`** command to update the time zone. Use `student` as the password if asked.

   2.1. Use the **`timedatectl`** command to view available time zones and determine the appropriate time zone for Jamaica.

```
[student@serverb ~]$ timedatectl list-timezones | grep Jamaica
America/Jamaica
```

   2.2. Use the **`timedatectl`** command to set the time zone of the `serverb` system to **`America/Jamaica`**.

```
[student@serverb ~]$ sudo timedatectl set-timezone America/Jamaica
[sudo] password for student: student
```

2.3. Use the **timedatectl** command to verify that the time zone is successfully set to **America/Jamaica**.

```
[student@serverb ~]$ timedatectl
                Local time: Tue 2019-02-19 11:12:46 EST
           Universal time: Tue 2019-02-19 16:12:46 UTC
                 RTC time: Tue 2019-02-19 16:12:45
                Time zone: America/Jamaica (EST, -0500)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

3. Display the log events recorded in the previous 30 minutes on serverb.

3.1. Use the **date** command to determine the time frame to view the journal entries.

```
[student@serverb ~]$ date
Fri Feb 22 07:31:05 EST 2019
[student@serverb ~]$ date -d "-30 minutes"
Fri Feb 22 07:01:31 EST 2019
```

3.2. Use the **journalctl** command **--since** and **--until** options to display log events recorded in the previous 30 minutes on serverb. To quit **journalctl**, press **q**.

```
[student@serverb ~]$ journalctl --since 07:01:00 --until 07:31:00
...output omitted...
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target
 Timers.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target
 Paths.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Starting D-Bus
 User Message Bus Socket.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Listening on D-
Bus User Message Bus Socket.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target
 Sockets.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target
 Basic System.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Reached target
 Default.
Feb 22 07:24:28 serverb.lab.example.com systemd[1138]: Startup finished
 in 123ms.
Feb 22 07:24:28 serverb.lab.example.com systemd[1]: Started User
 Manager for UID 1000.
Feb 22 07:24:28 serverb.lab.example.com sshd[1134]:
 pam_unix(sshd:session): session opened for user student by (uid=0)
Feb 22 07:26:56 serverb.lab.example.com systemd[1138]: Starting Mark
 boot as successful...
Feb 22 07:26:56 serverb.lab.example.com systemd[1138]: Started Mark
 boot as successful.
lines 1-36/36 (END) q
```

4. Create the **/etc/rsyslog.d/auth-errors.conf** file, configured to have the rsyslog service write messages related to authentication and security issues to the new **/var/log/auth-errors** file. Use the **authpriv** facility and the **alert** priority in the configuration file.

   4.1. Create the **/etc/rsyslog.d/auth-errors.conf** file to specify the new **/var/log/auth-errors** file as the destination for messages related to authentication and security issues. You may use the **sudo vim /etc/rsyslog.d/auth-errors.conf** command to create the configuration file.

```
authpriv.alert  /var/log/auth-errors
```

   4.2. Restart the rsyslog service so that the changes in the configuration file take effect.

```
[student@serverb ~]$ sudo systemctl restart rsyslog
```

   4.3. Use the **logger** command to write a new log message to the **/var/log/auth-errors** file. Apply the **-p authpriv.alert** option to generate a log message relevant to authentication and security issues.

```
[student@serverb ~]$ logger -p authpriv.alert "Logging test
 authpriv.alert"
```

   4.4. Use the **tail** command to confirm that the **/var/log/auth-errors** file contains the log entry with the **Logging test authpriv.alert** message.

```
[student@serverb ~]$ sudo tail /var/log/auth-errors
Feb 19 11:56:07 serverb student[6038]: Logging test authpriv.alert
```

   4.5. Log out of serverb.

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

Evaluation

On workstation, run the **lab log-review grade** command to confirm success of this exercise.

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

Finish

On workstation, run **lab log-review finish** to complete this lab. This script ensures that the original time zone is restored along with all the original time settings on serverb.

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

This concludes the guided exercise.

SUMMARY

In this chapter, you learned:

- The `systemd-journald` and `rsyslog` services capture and write log messages to the appropriate files.
- The **`/var/log`** directory contains log files.
- Periodic rotation of log files prevent them from filling up the file system space.
- The `systemd` journals are temporary and do not persist across reboot.
- The `chronyd` service helps to synchronize time settings with a time source.
- The time zone of the server can be updated based on its location.

CHAPTER 12

IMPLEMENTING ADVANCED STORAGE FEATURES

GOAL

Create and manage logical volumes containing file systems and swap spaces from the command line, and configure advanced storage features with Stratis and VDO.

OBJECTIVES

- Create and manage logical volumes from storage devices, and format them with file systems or prepare them with swap spaces.
- Add and remove storage assigned to volume groups, and non-destructively extend the size of a logical volume formatted with an XFS or ext4 file system.
- Manage multiple storage layers at once using Stratis local storage management.
- Optimize use of storage space by using VDO to compress and deduplicate data on storage devices.

SECTIONS

- Creating Logical Volumes (and Guided Exercise)
- Extending Logical Volumes (and Guided Exercise)
- Managing Layered Storage with Stratis (and Guided Exercise)
- Compressing and Deduplicating Storage with VDO (and Guided Exercise)

LAB

Implementing Advanced Storage Features

CREATING LOGICAL VOLUMES

OBJECTIVES

After completing this section, you should be able to:

- Describe logical volume management components and concepts.
- Implement LVM storage.
- Display LVM component information.

LOGICAL VOLUME MANAGEMENT (LVM) CONCEPTS

Logical volumes and logical volume management make it easier to manage disk space. If a file system that hosts a logical volume needs more space, it can be allocated to its logical volume from the free space in its volume group and the file system can be resized. If a disk starts to fail, a replacement disk can be registered as a physical volume with the volume group and the logical volume's extents can be migrated to the new disk.

LVM Definitions

Physical devices
: Physical devices are the storage devices used to save data stored in a logical volume. These are block devices and could be disk partitions, whole disks, RAID arrays, or SAN disks. A device must be initialized as an LVM physical volume in order to be used with LVM. The entire device will be used as a physical volume.

Physical volumes (PVs)
: You must initialize a device as a physical volume before using it in an LVM system. LVM tools segment physical volumes into *physical extents (PEs)*, which are small chunks of data that act as the smallest storage block on a physical volume.

Volume groups (VGs)
: Volume groups are storage pools made up of one or more physical volumes. This is the functional equivalent of a whole disk in basic storage. A PV can only be allocated to a single VG. A VG can consist of unused space and any number of logical volumes.

Logical volumes (LVs)
: Logical volumes are created from free physical extents in a volume group and provide the "storage" device used by applications, users, and the operating system. LVs are a collection of *logical extents (LEs)*, which map to physical extents, the smallest storage chunk of a PV. By default, each LE maps to one PE. Setting specific LV options changes this mapping; for example, mirroring causes each LE to map to two PEs.

IMPLEMENTING LVM STORAGE

Creating LVM storage requires several steps. The first step is to determine which physical devices to use. After a set of suitable devices have been assembled, they are initialized as physical volumes so that they are recognized as belonging to LVM. The physical volumes are then combined into a volume group. This creates a pool of disk space out of which logical volumes can be allocated.

Logical volumes created from the available space in a volume group can be formatted with a file system, activated as swap space, and mounted or activated persistently.

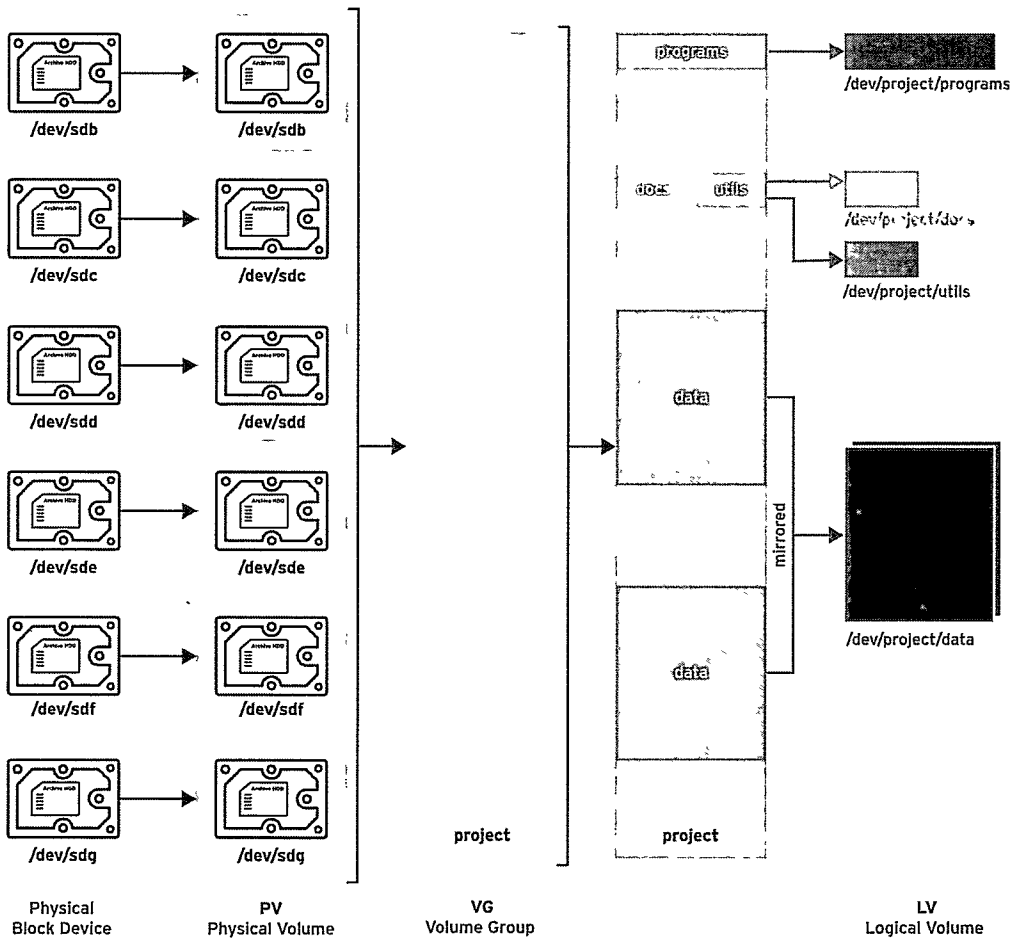


Figure 12.1: Logical volume management components

LVM provides a comprehensive set of command-line tools for implementing and managing LVM storage. These command-line tools can be used in scripts, making them suitable for automation.

> **IMPORTANT**
> The following examples use device **vdb** and its partitions to illustrate LVM commands. In practice, these examples would need to use the correct devices for the disk and disk partitions that are being used by the system. Use the `lsblk`, `blkid`, or `cat /proc/partitions` commands to identify the devices on your system.

Creating a Logical Volume

To create a logical volume, perform the following steps:

Prepare the physical device.

Use `parted`, `gdisk`, or `fdisk` to create a new partition for use with LVM. Always set the partition type to `Linux LVM` on LVM partitions; use `0x8e` for MBR partitions. If necessary, use `partprobe` to register the new partition with the kernel.

Alternatively, use a whole disk, a RAID array, or a SAN disk.

A physical device only needs to be prepared if there are none prepared already and a new physical volume is required to create or extend a volume group.

```
[root@host ~]# parted -s /dev/vdb mkpart primary 1MiB 769MiB
[root@host ~]# parted -s /dev/vdb mkpart primary 770MiB 1026MiB
[root@host ~]# parted -s /dev/vdb set 1 lvm on
[root@host ~]# parted -s /dev/vdb set 2 lvm on
```

Create a physical volume.

Use **pvcreate** to label the partition (or other physical device) as a physical volume. The **pvcreate** command divides the physical volume into physical extents (PEs) of a fixed size, for example, 4 MiB blocks. You can label multiple devices at the same time by using space-delimited device names as arguments to **pvcreate**.

```
[root@host ~]# pvcreate /dev/vdb2 /dev/vdb1
```

This labels the devices **/dev/vdb2** and **/dev/vdb1** as PVs, ready for allocation into a volume group.

A PV only needs to be created if there are no PVs free to create or extend a VG.

Create a volume group.

Use **vgcreate** to collect one or more physical volumes into a volume group. A volume group is the functional equivalent of a hard disk; you will create logical volumes from the pool of free physical extents in the volume group.

The **vgcreate** command-line consists of a volume group name followed by one or more physical volumes to allocate to this volume group.

```
[root@host ~]# vgcreate vg01 /dev/vdb2 /dev/vdb1
```

This creates a VG called vg01 that is the combined size, in PE units, of the two PVs **/dev/vdb2** and **/dev/vdb1**.

A VG only needs to be created if none already exist. Additional VGs may be created for administrative reasons to manage the use of PVs and LVs. Otherwise, existing VGs can be extended to accommodate new LVs when needed.

Create a logical volume.

Use **lvcreate** to create a new logical volume from the available physical extents in a volume group. At a minimum, the **lvcreate** command includes the **-n** option to set the LV name, either the **-L** option to set the LV size in bytes or the **-l** option to set the LV size in extents, and the name of the volume group hosting this logical volume.

```
[root@host ~]# lvcreate -n lv01 -L 700M vg01
```

This creates an LV called lv01, 700 MiB in size, in the VG vg01. This command will fail if the volume group does not have a sufficient number of free physical extents for the requested size.

Note also that the size will be rounded to a factor of the physical extent size if the size cannot match exactly.

You can specify the size using the **-L** option, which expects sizes in bytes, mebibytes (binary megabytes, 1048576 bytes), gibibytes (binary gigabytes), or similar. Alternatively, you can use the **-l** option, which expects sizes specified as a number of physical extents.

The following list provides some examples of creating LVs:

- **lvcreate -L 128M**: Size the logical volume to exactly 128 MiB.
- **lvcreate -l 128** : Size the logical volume to exactly 128 extents. The total number of bytes depends on the size of the physical extent block on the underlying physical volume.

> **IMPORTANT**
> Different tools display the logical volume name using either the traditional name, **/dev/*vgname/lvname***, or the kernel device mapper name, **/dev/mapper/*vgname-lvname***.

Add the file system.

Use **mkfs** to create an **XFS** file system on the new logical volume. Alternatively, create a file system based on your preferred file system, for example, **ext4**.

```
[root@host ~]# mkfs -t xfs /dev/vg01/lv01
```

To make the file system available across reboots, perform the following steps:

- Use **mkdir** to create a mount point.

```
[root@host ~]# mkdir /mnt/data
```

- Add an entry to the **/etc/fstab** file:

```
/dev/vg01/lv01  /mnt/data xfs  defaults 1 2
```

> **NOTE**
> Mounting a logical volume by name is equivalent to mounting by UUID because LVM finds its physical volumes based on a UUID even if you initially add them to the volume group by name.

- Run **mount /mnt/data** to mount the file system that you just added in **/etc/fstab**.

```
[root@host ~]# mount /mnt/data
```

Removing a Logical Volume

To remove *all* logical volume components, perform the following steps:

Prepare the file system.

Move all data that must be kept to another file system. Use the **umount** command to unmount the file system and then remove any **/etc/fstab** entries associated with this file system.

```
[root@host ~]# umount /mnt/data
```

> **WARNING**
> Removing a logical volume destroys any data stored on the logical volume. Back up or move your data *before* you remove the logical volume.

Remove the logical volume.

Use **lvremove** ***DEVICE_NAME*** to remove a logical volume that is no longer needed.

```
[root@host ~]# lvremove /dev/vg01/lv01
```

Unmount the LV file system before running this command. The command prompts for confirmation before removing the LV.

The LV's physical extents are freed and made available for assignment to existing or new LVs in the volume group.

Remove the volume group.

Use **vgremove** ***VG_NAME*** to remove a volume group that is no longer needed.

```
[root@host ~]# vgremove vg01
```

The VG's physical volumes are freed and made available for assignment to existing or new VGs on the system.

Remove the physical volumes.

Use **pvremove** to remove physical volumes that are no longer needed. Use a space-delimited list of PV devices to remove more than one at a time. This command deletes the PV metadata from the partition (or disk). The partition is now free for reallocation or reformatting.

```
[root@host ~]# pvremove /dev/vdb2 /dev/vdb1
```

REVIEWING LVM STATUS INFORMATION

Physical Volumes

Use **pvdisplay** to display information about physical volumes. To list information about all physical volumes, use the command without arguments. To list information about a specific physical volume, pass that device name to the command.

```
[root@host ~]# pvdisplay /dev/vdb1
  --- Physical volume ---
  PV Name               /dev/vdb1                    ❶
  VG Name               vg01                         ❷
```

```
  PV Size               768.00 MiB / not usable 4.00 MiB ❸
  Allocatable           yes
  PE Size               4.00 MiB                         ❹
  Total PE              191
  Free PE               16                               ❺
  Allocated PE          175
  PV UUID               JWzDpn-LG3e-n2oi-9Etd-VT2H-PMem-1ZXwP1
```

❶ **PV Name** maps to the device name.
❷ **VG Name** shows the volume group where the PV is allocated.
❸ **PV Size** shows the physical size of the PV, including any unusable space.
❹ **PE Size** is the physical extent size, which is the smallest size a logical volume can be allocated.

It is also the multiplying factor when calculating the size of any value reported in PE units, such as *Free PE*; for example: 26 PEs x 4 MiB (the *PE Size*) equals 104 MiB of free space. A logical volume size is rounded to a factor of PE units.

LVM sets the PE size automatically, although it is possible to specify it.

❺ **Free PE** shows how many PE units are available for allocation to new logical volumes.

Volume Groups

Use **vgdisplay** to display information about volume groups. To list information about all volume groups, use the command without arguments. To list information about a specific volume group, pass that VG name to the command.

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---
  VG Name               vg01                ❶
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               1
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               1016.00 MiB         ❷
  PE Size               4.00 MiB
  Total PE              254                 ❸
  Alloc PE / Size       175 / 700.00 MiB
  Free  PE / Size       79 / 316.00 MiB     ❹
  VG UUID               3snNw3-CF71-CcYG-Llk1-p6EY-rHEv-xfUSez
```

❶ **VG Name** is the name of the volume group.
❷ **VG Size** is the total size of the storage pool available for logical volume allocation.

❸ **Total PE** is the total size expressed in PE units.
❹ **Free PE / Size** shows how much space is free in the VG for allocating to new LVs or to extend existing LVs.

Logical Volumes

Use **lvdisplay** to display information about logical volumes. If you provide no argument to the command, it displays information about all LVs; if you provide an LV device name as an argument, the command displays information about that specific device.

```
[root@host ~]# lvdisplay /dev/vg01/lv01
  --- Logical volume ---
  LV Path                /dev/vg01/lv01          ❶
  LV Name                lv01
  VG Name                vg01                    ❷
  LV UUID                5IyRea-W8Zw-xLHk-3h2a-IuVN-YaeZ-i3IRrN
  LV Write Access        read/write
  LV Creation host, time host.lab.example.com, 2019-03-28 17:17:47 -0400
  LV Status              available
  # open                 1
  LV Size                700 MiB                 ❸
  Current LE             175                     ❹
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - current set to       256
  Block device           252:0
```

❶ **LV Path** shows the device name of the logical volume.

Some tools may report the device name as **/dev/mapper/*vgname-lvname***; both represent the same LV.

❷ **VG Name** shows the volume group that the LV is allocated from.
❸ **LV Size** shows the total size of the LV. Use file-system tools to determine the free space and used space for storage of data.
❹ **Current LE** shows the number of logical extents used by this LV. An LE usually maps to a physical extent in the VG, and therefore the physical volume.

REFERENCES

lvm(8), **pvcreate**(8), **vgcreate**(8), **lvcreate**(8), **pvremove**(8), **vgremove**(8), **lvremove**(8), **pvdisplay**(8), **vgdisplay**(8), **lvdisplay**(8), **fdisk**(8), **gdisk**(8), **parted**(8), **partprobe**(8), and **mkfs**(8) man pages

GUIDED EXERCISE

CREATING LOGICAL VOLUMES

In this lab, you will create a physical volume, volume group, logical volume, and an XFS file system. You will also persistently mount the logical volume file system.

OUTCOMES

You should be able to:

- Create physical volumes, volume groups, and logical volumes with LVM tools.
- Create new file systems on logical volumes and persistently mount them.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab lvm-creating start`** command. This command runs a start script that determines if the `servera` machine is reachable on the network. It also verifies that storage is available and that the appropriate software packages are installed.

```
[student@workstation ~]$ lab lvm-creating start
```

1. Use the **ssh** command to log in to `servera` 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 ~]$
```

2. Use the **sudo -i** command to switch to the `root` user. The password for the `student` user is `student`.

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

3. Create the physical resources.

   3.1. Use **parted** to create two 256 MiB partitions and set them to type Linux LVM.

```
[root@servera ~]# parted -s /dev/vdb mklabel gpt
[root@servera ~]# parted -s /dev/vdb mkpart primary 1MiB 257MiB
[root@servera ~]# parted -s /dev/vdb set 1 lvm on
[root@servera ~]# parted -s /dev/vdb mkpart primary 258MiB 514MiB
[root@servera ~]# parted -s /dev/vdb set 2 lvm on
```

3.2. Use **udevadm settle** for the system to register the new partitions.

```
[root@servera ~]# udevadm settle
```

4. Use **pvcreate** to add the two new partitions as PVs.

```
[root@servera ~]# pvcreate /dev/vdb1 /dev/vdb2
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
```

5. Use **vgcreate** to create a new VG named servera_01_vg built from the two PVs.

```
[root@servera ~]# vgcreate servera_01_vg /dev/vdb1 /dev/vdb2
  Volume group "servera_01_vg" successfully created
```

6. Use **lvcreate** to create a 400 MiB LV named servera_01_lv from the servera_01_vg VG.

```
[root@servera ~]# lvcreate -n servera_01_lv -L 400M servera_01_vg
  Logical volume "servera_01_lv" created.
```

This creates a device named **/dev/servera_01_vg/servera_01_lv** but without a file system on it.

7. Add a persistent file system.

7.1. Add an XFS file system on the `servera_01_lv` LV with the **mkfs** command.

```
[root@servera ~]# mkfs -t xfs /dev/servera_01_vg/servera_01_lv
...output omitted...
```

7.2. Create a mount point at **/data**.

```
[root@servera ~]# mkdir /data
```

7.3. Add the following line to the end of **/etc/fstab** on `servera`:

```
/dev/servera_01_vg/servera_01_lv    /data    xfs  defaults  1 2
```

7.4. Use **systemctl daemon-reload** to update `systemd` with the new **/etc/fstab** configuration.

```
[root@servera ~]# systemctl daemon-reload
```

7.5. Verify the **/etc/fstab** entry and mount the new `servera_01_lv` LV device with the **mount** command.

```
[root@servera ~]# mount /data
```

8. Test and review your work.

8.1. As a final test, copy some files to **/data** and verify how many were copied.

```
[root@servera ~]# cp -a /etc/*.conf /data
[root@servera ~]# ls /data | wc -l
34
```

You will verify that you still have the same number of files in the next guided exercise.

8.2. **parted /dev/vdb print** lists the partitions that exist on **/dev/vdb**.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

```
Number  Start   End     Size    File system  Name     Flags
 1      1049kB  269MB   268MB                primary  lvm
 2      271MB   539MB   268MB                primary  lvm
```

Notice the **Number** column, which contains the values **1** and **2**. These correspond to **/dev/vdb1** and **/dev/vdb2**, respectively. Also notice the **Flags** column, which indicates the partition type.

8.3. **pvdisplay** displays information about each of the physical volumes. Optionally, include the device name to limit details to a specific PV.

```
[root@servera ~]# pvdisplay /dev/vdb2
  --- Physical volume ---
  PV Name               /dev/vdb2
  VG Name               servera_01_vg
  PV Size               256.00 MiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
```

```
  Total PE              63
  Free PE               26
  Allocated PE          37
  PV UUID               2z0Cf3-99YI-w9ny-alEW-wWhL-S8RJ-M2rfZk
```

This shows that the PV is allocated to VG `servera_01_vg`, is 256 MiB in size (although 4 MiB is not usable), and the physical extent size (**`PE Size`**) is 4 MiB (the smallest allocatable LV size).

There are 63 PEs, of which 26 are free for allocation to LVs in the future and 37 are currently allocated to LVs. These translate to MiB values as follows:

- Total 252 MiB (63 PEs x 4 MiB); remember, 4 MiB is unusable.
- Free 104 MiB (26 PEs x 4 MiB)
- Allocated 148 MiB (37 PEs x 4 MiB)

8.4. **`vgdisplay`** ***`vgname`*** shows information about the volume group named `vgname`.

```
[root@servera ~]# vgdisplay servera_01_vg
```

Verify the following values:

- **`VG Size`** is **`504.00MiB`**.
- **`Total PE`** is **`126`**.
- **`Alloc PE / Size`** is **`100 / 400.00MiB`**.
- **`Free PE / Size`** is **`26 / 104.00MiB`**.

8.5. **`lvdisplay /dev/`*`vgname/lvname`*** displays information about the logical volume named `lvname`.

```
[root@servera ~]# lvdisplay /dev/servera_01_vg/servera_01_lv
```

Review the **`LV Path`**, **`LV Name`**, **`VG Name`**, **`LV Status`**, **`LV Size`**, and **`Current LE`** (logical extents, which map to physical extents).

8.6. The **`mount`** command shows all mounted devices and any mount options. It should include **`/dev/servera_01_vg/servera_01_lv`**.

> **NOTE**
> Many tools report the device mapper name instead, **`/dev/mapper/servera_01_vg-servera_01_lv`**; it is the same logical volume.

```
[root@servera ~]# mount
```

You should see (probably on the last line) **`/dev/mapper/servera_01_vg-servera_01_lv`** mounted on **`/data`** and the associated mount information.

8.7. **df -h** displays human-readable disk free space. Optionally, include the mount point to limit details to that file system.

```
[root@servera ~]# df -h /data
Filesystem                                  Size  Used Avail Use%
 Mounted on
/dev/mapper/servera_01_vg-servera_01_lv  395M   24M  372M   6% /data
```

Allowing for file-system metadata, these values are expected.

9. Log off from servera.

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

Finish

On workstation, run the **lab lvm-creating finish** script to finish this exercise. This script removes the storage configured on servera during the exercise.

```
[student@workstation ~]$ lab lvm-creating finish
```

This concludes the guided exercise.

EXTENDING LOGICAL VOLUMES

OBJECTIVES

After completing this section, you should be able to:

- Extend a volume group (VG) using **`pvcreate`** and **`vgextend`**, and use **`vgdisplay`** to verify the results.
- Reduce a VG using **`pvmove`** and **`vgreduce`**.
- Extend a logical volume (LV) using **`lvextend`**.
- Resize **XFS** file systems with **`xfs_growfs`**.
- Resize **`ext4`** file systems with **`resize2fs`**.

EXTENDING AND REDUCING A VOLUME GROUP

You can add more disk space to a volume group by adding additional physical volumes. This is called *extending the volume group*. Then, you can assign the new physical extents from the additional physical volumes to logical volumes.

You can remove unused physical volumes from a volume group. This is called *reducing the volume group*. First, use the **`pvmove`** command to move data from extents on one physical volume to extents on other physical volumes in the volume group. In this way, a new disk can be added to an existing volume group, data can be moved from an older or slower disk to a new disk, and the old disk removed from the volume group. You can perform these actions while the logical volumes in the volume group are in use.

IMPORTANT
The following examples use the device **vdb** and its partitions to illustrate LVM commands. In practice, use the appropriate devices for the disk and disk partitions on your own system.

Extending a Volume Group

To extend a volume group, perform the following steps:

Prepare the physical device and create the physical volume.

As with creating a new volume group, you must create and prepare a new partition for use as a physical volume if there are none prepared already.

```
[root@host ~]# parted -s /dev/vdb mkpart primary 1027MiB 1539MiB
[root@host ~]# parted -s /dev/vdb set 3 lvm on
[root@host ~]# pvcreate /dev/vdb3
```

A PV only needs to be created if there are no PVs free to extend the VG.

Extend the volume group.

Use **vgextend** to add the new physical volume to the volume group. Use the VG name and PV device name as arguments to **vgextend**.

```
[root@host ~]# vgextend vg01 /dev/vdb3
```

This extends the vg01 VG by the size of the **/dev/vdb3** PV.

Verify that the new space is available.

Use **vgdisplay** to confirm the additional physical extents are available. Inspect the **Free PE / Size** in the output. It should not be zero.

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---
  VG Name               vg01
...output omitted...
  Free  PE / Size       178 / 712.00 MiB
...output omitted...
```

Reducing a Volume Group

To reduce a volume group, perform the following steps:

Move the physical extents.

Use **pvmove *PV_DEVICE_NAME*** to relocate any physical extents from the physical volume you want to remove to other physical volumes in the volume group. The other physical volumes must have a sufficient number of free extents to accommodate this move. This is only possible if there are enough free extents in the VG and if all of those come from other PVs.

```
[root@host ~]# pvmove /dev/vdb3
```

This command moves the PEs from **/dev/vdb3** to other PVs with free PEs in the same VG.

> **WARNING**
> Before using **pvmove**, back up data stored on all logical volumes in the volume group. An unexpected power loss during the operation may leave the volume group in an inconsistent state. This could cause loss of data on logical volumes in the volume group.

Reduce the volume group.

Use **vgreduce *VG_NAME PV_DEVICE_NAME*** to remove a physical volume from a volume group.

```
[root@host ~]# vgreduce vg01 /dev/vdb3
```

This removes the **/dev/vdb3** PV from the vg01 VG and it can now be added to another VG. Alternatively, **pvremove** can be used to permanently stop using the device as a PV.

EXTENDING A LOGICAL VOLUME AND XFS FILE SYSTEM

One benefit of logical volumes is the ability to increase their size without experiencing downtime. Free physical extents in a volume group can be added to a logical volume to extend its capacity, which can then be used to extend the file system it contains.

Extending a Logical Volume

To extend a logical volume, perform the following steps:

Verify that the volume group has space available.

Use **vgdisplay** to verify that there are sufficient physical extents available.

```
[root@host ~]# vgdisplay vg01
  --- Volume group ---
  VG Name               vg01
...output omitted...
  Free  PE / Size       178 / 712.00 MiB
...output omitted...
```

Inspect the **Free PE / Size** in the output. Confirm that the volume group has sufficient free space for the LV extension. If insufficient space is available, then extend the volume group appropriately. See the section called "Extending and Reducing a Volume Group".

Extend the logical volume.

Use **lvextend***LV_DEVICE_NAME* to extend the logical volume to a new size.

```
[root@host ~]# lvextend -L +300M /dev/vg01/lv01
```

This increases the size of the logical volume lv01 by 300 MiB. Notice the plus sign (+) in front of the size, which means add this value to the existing size; otherwise, the value defines the final size of the LV.

As with **lvcreate**, different methods exist to specify the size: the **-l** option expects the number of physical extents as the argument. The **-L** option expects sizes in bytes, mebibytes, gibibytes, and similar.

The following list provides some examples of extending LVs.

Extending LVs Examples

COMMAND	RESULTS
lvextend -l 128	Resize the logical volume to *exactly* 128 extents in size.
lvextend -l +128	Add 128 extents to the current size of the logical volume.
lvextend -L 128M	Resize the logical volume to *exactly* 128 MiB.
lvextend -L +128M	Add 128 MiB to the current size of the logical volume.

COMMAND	RESULTS
`lvextend -l +50%FREE`	Add 50 percent of the current free space in the VG to the LV.

Extend the file system.

Use `xfs_growfs mountpoint` to expand the file system to occupy the extended LV. The target file system must be mounted when you use the `xfs_growfs` command. You can continue to use the file system while it is being resized.

```
[root@host ~]# xfs_growfs /mnt/data
```

> **NOTE**
> A common mistake is to run `lvextend` but to forget to run `xfs_growfs`. An alternative to running the two steps consecutively is to include the `-r` option with the `lvextend` command. This resizes the file system after the LV is extended, using `fsadm`(8). It works with a number of different file systems.

- Verify the new size of the mounted file system:

  `df -h /mountpoint`.

EXTENDING A LOGICAL VOLUME AND EXT4 FILE SYSTEM

The steps for extending an `ext4`-based logical volume are essentially the same as for an **XFS**-based LV, except for the step that resizes the file system. Review the section called "Extending a Logical Volume and XFS File System".

Verify that the volume group has space available.

Use `vgdisplay VGNAME` to verify that the volume group has a sufficient number of physical extents available.

Extend the logical volume.

Use `lvextend -l +extents /dev/vgname/lvname` to extend the logical volume */dev/vgname/lvname* by the *extents* value.

Extend the file system.

Use `resize2fs /dev/vgname/lvname` to expand the file system to occupy the new extended LV. The file system can be mounted and in use while the extension command is running. You can include the `-p` option to monitor the progress of the resize operation.

```
[root@host ~]# resize2fs /dev/vg01/lv01
```

> **NOTE**
> The primary difference between `xfs_growfs` and `resize2fs` is the argument passed to identify the file system. `xfs_growfs` takes the mount point and `resize2fs` takes the logical volume name.

EXTEND A LOGICAL VOLUME AND SWAP SPACE

Logical volumes formatted as swap space can be extended as well, however the process is different than the one for extending a file system, such as **ext4** or **XFS**. Logical volumes formatted with a file system can be extended dynamically with no downtime. Logical volumes formatted with swap space must be taken offline in order to extend them.

Verify the volume group has space available.

Use **vgdisplay** *vgname* to verify that a sufficient number of free physical extents are available.

Deactivate the swap space.

Use **swapoff -v /dev/*vgname/lvname*** to deactivate the swap space on the logical volume.

WARNING

Your system must have enough free memory or swap space to accept anything that needs to page in when the swap space on the logical volume is deactivated.

Extend the logical volume.

lvextend -l +*extents* /dev/*vgname/lvname* extends the logical volume */dev/vgname/lvname* by the *extents* value.

Format the logical volume as swap space.

mkswap /dev/*vgname/lvname* formats the entire logical volume as swap space.

Activate the swap space.

Use **swapon -va /dev/*vgname/lvname*** to activate the swap space on the logical volume.

REFERENCES

lvm(8), **pvcreate**(8), **pvmove**(8), **vgdisplay**(8), **vgextend**(8), **vgreduce**(8), **vgdisplay**(8), **vgextend**(8), **vgreduce**(8), **lvextend**(8), **fdisk**(8), **gdisk**(8), **parted**(8), **partprobe**(8), **xfs_growfs**(8), and **resize2fs**(8) **swapoff**(8) **swapon**(8) **mkswap**(8)man pages

GUIDED EXERCISE

EXTENDING LOGICAL VOLUMES

In this lab, you will extend the logical volume added in the previous practice exercise.

OUTCOMES

You should be able to:

- Extend the volume group to include an additional physical volume.
- Resize the logical volume while the file system is still mounted and in use.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

On `workstation`, run the **`lab lvm-extending start`** command. This command runs a start script that determines if the host `servera` is reachable on the network and ensures that the storage from the previous guided exercise is available.

```
[student@workstation ~]$ lab lvm-extending start
```

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

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

2. Use the **sudo -i** command to switch to `root` at the shell prompt.

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

3. Use **vgdisplay** to determine if the VG has sufficient free space to extend the LV to a total size of 700 MiB.

```
[root@servera ~]# vgdisplay servera_01_vg
  --- Volume group ---
  VG Name               servera_01_vg
  System ID
  Format                lvm2
...output omitted...
  VG Size               504.00 MiB
  PE Size               4.00 MiB
  Total PE              126
```

```
  Alloc PE / Size       100 / 400.00 MiB
  Free  PE / Size       26 / 104.00 MiB
  VG UUID               OBBAtU-2nBS-4SW1-khmF-yJzi-z7bD-DpCrAV
```

Only 104 MiB is available (26 PEs x 4 MiB extents) and you need at least 300 MiB to have 700 MiB in total. You need to extend the VG.

For later comparison, use **df** to record current disk free space:

```
[root@servera ~]# df -h /data
Filesystem                              Size  Used Avail Use% Mounted on
/dev/mapper/servera_01_vg-servera_01_lv 395M   24M  372M   6% /data
```

4. Create the physical resource.

   4.1. Use **parted** to create an additional partition of 512 MiB and set it to type Linux LVM.

```
[root@servera ~]# parted -s /dev/vdb mkpart primary 515MiB 1027MiB
[root@servera ~]# parted -s /dev/vdb set 3 lvm on
```

   4.2. Use **udevadm settle** for the system to register the new partition.

```
[root@servera ~]# udevadm settle
```

5. Use **pvcreate** to add the new partition as a PV.

```
[root@servera ~]# pvcreate /dev/vdb3
  Physical volume "/dev/vdb3" successfully created.
```

6. Extend the volume group.

   6.1. Use **vgextend** to extend the VG named `servera_01_vg`, using the new `/dev/vdb3` PV.

```
[root@servera ~]# vgextend servera_01_vg /dev/vdb3
  Volume group "servera_01_vg" successfully extended
```

   6.2. Use **vgdisplay** to inspect the `servera_01_vg` VG free space again. There should be plenty of free space now.

```
[root@servera ~]# vgdisplay servera_01_vg
  --- Volume group ---
  VG Name               servera_01_vg
  System ID
  Format                lvm2
...output omitted...
  VG Size               1012.00 MiB
  PE Size               4.00 MiB
```

```
  Total PE              253
  Alloc PE / Size       100 / 400.00 MiB
  Free  PE / Size       153 / 612.00 MiB
  VG UUID               OBBAtU-2nBS-4SW1-khmF-yJzi-z7bD-DpCrAV
```

612 MiB of free space is now available (153 PEs x 4 MiB extents).

7. Use **lvextend** to extend the existing LV to 700 MiB.

```
[root@servera ~]# lvextend -L 700M /dev/servera_01_vg/servera_01_lv
  Size of logical volume servera_01_vg/servera_01_lv changed from 400.00 MiB
 (100 extents) to 700.00 MiB (175 extents).
    Logical volume servera_01_vg/servera_01_lv successfully resized.
```

NOTE

The example specifies the exact size to make the final LV, but you could have specified the amount of additional space desired:

- **-L +300M** to add the new space using size in MiB.
- **-l 175** to specify the total number of extents (175 PEs x 4 MiB).
- **-l +75** to add the additional extents needed.

8. Use **xfs_growfs** to extend the XFS file system to the remainder of the free space on the LV.

```
[root@servera ~]# xfs_growfs /data
meta-data=/dev/mapper/servera_01_vg-servera_01_lv isize=512    agcount=4,
 agsize=25600 blks
...output omitted...
```

9. Use **df** and **ls | wc** to review the new file-system size and verify that the previously existing files are still present.

```
[root@servera ~]# df -h /data
Filesystem                              Size  Used Avail Use% Mounted on
/dev/mapper/servera_01_vg-servera_01_lv 695M   26M  670M   4% /data
[root@servera ~]# ls /data | wc -l
34
```

The files still exist and the file system approximates the specified size.

10. Log off from servera.

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

Finish

On workstation, run the **lab lvm-extending finish** command to finish this exercise. This script removes the storage configured on servera during the exercise.

```
[student@workstation ~]$ lab lvm-extending finish
```

This concludes the guided exercise.

MANAGING LAYERED STORAGE WITH STRATIS

OBJECTIVES

After completing this section, you should be able to manage multiple storage layers using Stratis local storage management.

DESCRIBING THE STRATIS ARCHITECTURE

The current local storage solution in Red Hat Enterprise Linux (RHEL) includes many stable and mature technologies, including the device mapper (dm), the logical volume manager (LVM) and the XFS file system. Features provided by these components include massively scalable file systems, snapshots, redundant (RAID) logical devices, multipathing, thin provisioning, caching, deduplication, and support for virtual machines and containers. Each storage stack layer (dm, LVM, and XFS) is managed using layer-specific commands and utilities, requiring that system administrators manage physical devices, fixed-size volumes, and file systems as separate storage components.

A new generation of storage management solutions appeared in recent years, referred to as *volume-managing file systems*, that dynamically and transparently manage the volume layer as file systems are created and sized. However, although the community development of these file systems was ongoing for years, none reached the level of feature support and stability required to become the primary local storage for Red Hat Enterprise Linux.

With RHEL 8, Red Hat introduces the Stratis storage management solution. Instead of developing from scratch, as other storage projects attempted, Stratis works with existing RHEL storage components. Stratis runs as a service that manages pools of physical storage devices, and transparently creates and manages volumes for the file systems being created. Because Stratis uses existing storage drivers and tools, all of the advanced storage features that you currently use in LVM, XFS, and the device mapper are also supported by Stratis.

In a volume-managed file system, file systems are built inside shared *pools* of disk devices using a concept known as *thin provisioning*. Stratis file systems do not have fixed sizes and no longer preallocate unused block space. Although the file system is still built on a hidden LVM volume, Stratis manages the underlying volume for you and can expand it when needed. The in-use size of a file system is seen as the amount of actual blocks in use by contained files. The space available to a file system is the amount of space still unused in the pooled devices on which it resides. Multiple file systems can reside in the same pool of disk devices, sharing the available space, but file systems can also reserve pool space to guarantee availability when needed.

Stratis uses stored metadata to recognize managed pools, volumes, and file systems. Therefore, file systems created by Stratis should never be reformatted or reconfigured manually; they should only be managed using Stratis tools and commands. Manually configuring Stratis file systems could cause the loss of that metadata and prevent Stratis from recognizing the file systems it has created.

You can create multiple pools with different sets of block devices. From each pool, you can create one or more file systems. Currently, you can create up to 2^{24} file systems per pool. The following diagram illustrates how the elements of the Stratis storage management solution are positioned.

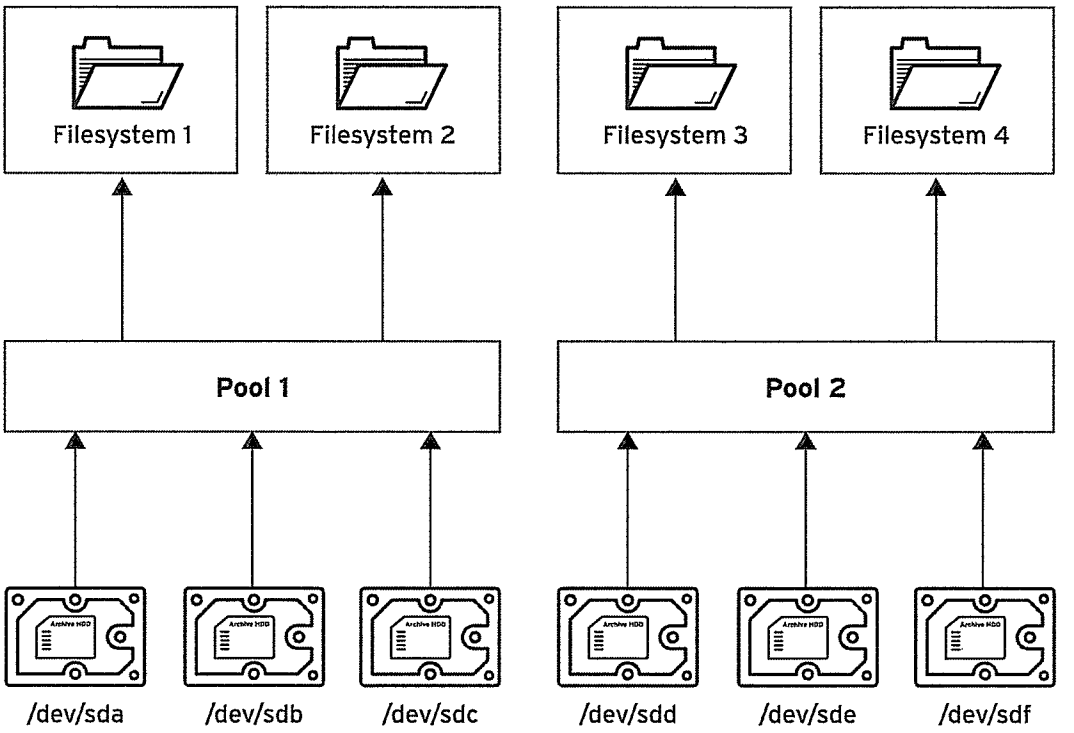


Figure 12.2: Elements of Stratis

A pool groups block devices into the *data tier* and optionally the *cache tier*. The data tier focuses on flexibility and integrity and the cache tier focuses on improved performance. Because the cache tier is intended to improve performance, you should use block devices that have higher input/output per second (IOPS), such as SSDs.

Describing the Simplified Storage Stack

Stratis simplifies many aspects of local storage provisioning and configuration across a range of Red Hat products. For example, in earlier versions of the ***Anaconda*** installer, system administrators had to layer each aspect of disk management over the other. Now, the installer uses Stratis, simplifying disk setup. Other products that use Stratis include ***Cockpit***, Red Hat Virtualization, and Red Hat Enterprise Linux Atomic Host. For all of these products, Stratis makes it simpler and less error prone to manage storage space and snapshots. Stratis allows easier integration with the higher-level management tools than using any CLI programmatically.

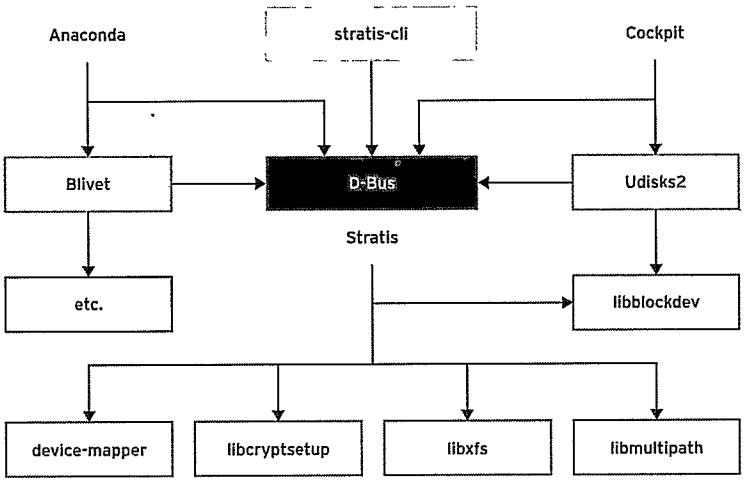


Figure 12.3: Stratis in the Linux storage management stack

Describing Stratis Layers

Internally, Stratis uses the `Backstore` subsystem to manage the block devices, and the `Thinpool` subsystem to manage the pools. The `Backstore` subsystem has a data tier that maintains the on-disk metadata on block devices, and detects and corrects data corruption. The cache tier uses high-performance block devices to act as a cache on top of the data tier. The `Thinpool` subsystem manages the thin-provisioned volumes associated with the Stratis file systems. This subsystem uses the `dm-thin` device mapper driver to replace LVM on the virtual volume sizing and management. `dm-thin` creates volumes with a large virtual size, formatted with XFS, but with a small physical size. As the physical size nears full, Stratis enlarges it automatically.

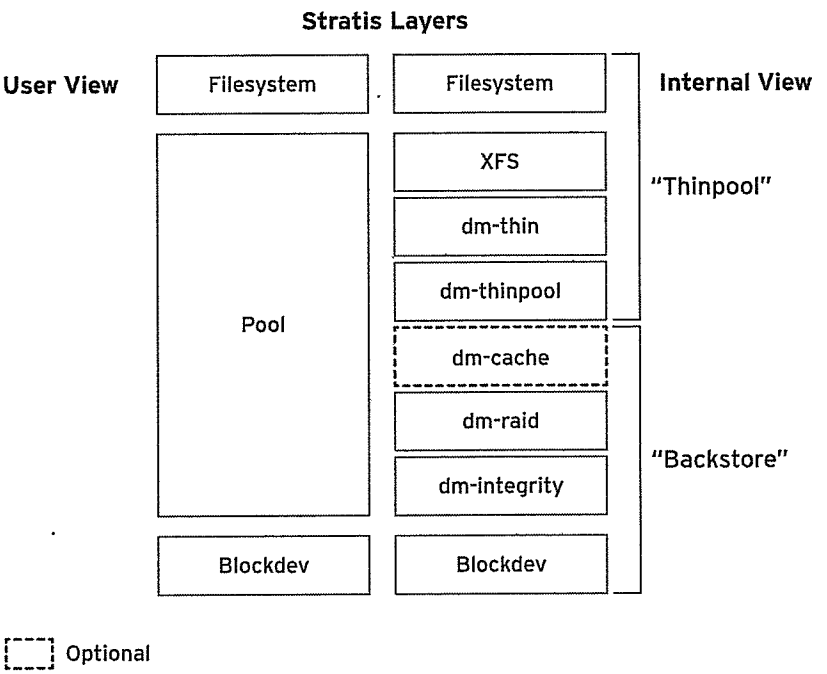


Figure 12.4: Stratis layers

Managing Thin-provisioned File Systems

To manage the thin-provisioned file systems using the Stratis storage management solution, install the *stratis-cli* and *stratisd* packages. The *stratis-cli* package provides the **stratis** command, which translates user requests to the `stratisd` service via the `D-Bus` API. The *stratisd* package provides the `stratisd` service, which implements the `D-Bus` interface, and manages and monitors the elements of Stratis, such as block devices, pools, and file systems. The `D-Bus` API is available if the `stratisd` service is running.

Install and activate Stratis using the usual tools:

- Install *stratis-cli* and *stratisd* using the **yum install** command.

```
[root@host ~]# yum install stratis-cli stratisd
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- Activate the `stratisd` service using the **systemctl** command.

```
[root@host ~]# systemctl enable --now stratisd
```

The following are common management operations performed using the Stratis storage management solution.

- Create pools of one or more block devices using the **stratis pool create** command.

```
[root@host ~]# stratis pool create pool1 /dev/vdb
```

  Each pool is a subdirectory under the **/stratis** directory.

- Use the **stratis pool list** command to view the list of available pools.

```
[root@host ~]# stratis pool list
Name     Total Physical Size  Total Physical Used
pool1                  5 GiB               52 MiB
```

- Use the **stratis pool add-data** command to add additional block devices to a pool.

```
[root@host ~]# stratis pool add-data pool1 /dev/vdc
```

- Use the **stratis blockdev list** command to view the block devices of a pool.

```
[root@host ~]# stratis blockdev list pool1
Pool Name  Device Node    Physical Size   State  Tier
pool1      /dev/vdb               5 GiB  In-use  Data
pool1      /dev/vdc               5 GiB  In-use  Data
```

- Use the **stratis filesystem create** command to create a dynamic and flexible file system from a pool.

```
[root@host ~]# stratis filesystem create pool1 filesystem1
```

The links to the Stratis file systems are in the **/stratis/*pool1*** directory.

- Stratis supports file-system snapshotting with the **stratis filesystem snapshot** command. Snapshots are independent of the source file systems.

```
[root@host ~]# stratis filesystem snapshot pool1 filesystem1 snapshot1
```

- Use the **stratis filesystem list** command to view the list of available file systems.

```
[root@host ~]# stratis filesystem list
...output omitted...
```

To ensure that the Stratis file systems are persistently mounted, edit **/etc/fstab** and specify the details of the file system. The following command displays the UUID of the file system that you should use in **/etc/fstab** to identify the file system.

```
[root@host ~]# lsblk --output=UUID /stratis/pool1/filesystem1
UUID
31b9363b-add8-4b46-a4bf-c199cd478c55
```

The following is an example entry in the **/etc/fstab** file to persistently mount a Stratis file system.

```
UUID=31b9...8c55 /dir1 xfs defaults,x-systemd.requires=stratisd.service 0 0
```

The **x-systemd.requires=stratisd.service** mount option delays mounting the file system until after `systemd` starts the `stratisd.service` during the boot process.

NOTE

Failure to use the **x-systemd.requires=stratisd.service** mount option in **/etc/fstab** for the Stratis file system will result in the machine booting to `emergency.target` on the next reboot.

REFERENCES

For more information, refer to the *Managing layered local storage with Stratis* chapter in the *Red Hat Enterprise Linux 8 Configuring and Managing File Systems Guide* at
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_file_systems/

Stratis Storage
https://stratis-storage.github.io/

What Stratis learned from ZFS, Btrfs, and Linux Volume Manager
https://opensource.com/article/18/4/stratis-lessons-learned

GUIDED EXERCISE

MANAGING LAYERED STORAGE WITH STRATIS

In this exercise, you will use the Stratis storage management solution to create pools, volumes, and file systems that work in cooperation.

OUTCOMES

You should be able to:

- Create a thin-provisioned file system using Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thin-provisioned file system.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab advstorage-stratis start`** to start the exercise. This script sets up the environment correctly and ensures that the additional disks on `servera` are clean.

```
[student@workstation ~]$ lab advstorage-stratis start
```

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

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

2. Switch to the `root` user.

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

3. Install the *stratisd* and *stratis-cli* packages using **yum**.

```
[root@servera ~]# yum install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

4. Activate the `stratisd` service using the **`systemctl`** command.

```
[root@servera ~]# systemctl enable --now stratisd
```

5. Ensure that the `stratispool1` Stratis pool exists with the block device `/dev/vdb`.

   5.1. Create a Stratis pool named `stratispool1` using the **`stratis pool create`** command.

```
[root@servera ~]# stratis pool create stratispool1 /dev/vdb
```

   5.2. Verify the availability of `stratispool1` using the **`stratis pool list`** command.

```
[root@servera ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
stratispool1                  5 GiB               52 MiB
```

   Note the size of the pool in the preceding output.

6. Expand the capacity of `stratispool1` using the `/dev/vdc` block device.

   6.1. Add the block device `/dev/vdc` to `stratispool1` using the **`stratis pool add-data`** command.

```
[root@servera ~]# stratis pool add-data stratispool1 /dev/vdc
```

   6.2. Verify the size of `stratispool1` using the **`stratis pool list`** command.

```
[root@servera ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
stratispool1                 10 GiB               56 MiB
```

   As shown above, the `stratispool1` pool size increased when you added the block device.

   6.3. Verify the block devices that are currently members of `stratispool1` using the **`stratis blockdev list`** command.

```
[root@servera ~]# stratis blockdev list stratispool1
Pool Name     Device Node    Physical Size   State  Tier
stratispool1  /dev/vdb               5 GiB  In-use  Data
stratispool1  /dev/vdc               5 GiB  In-use  Data
```

7. Add a thin-provisioned file system named `stratis-filesystem1` in the pool `stratispool1`. Mount the file system on **/stratisvol**. Create a file on the `stratis-filesystem1` file system named **file1** containing the text **Hello World!**.

7.1. Create the thin-provisioned file system `stratis-filesystem1` on `stratispool1` using the **stratis filesystem create** command. It may take up to a minute for the command to complete.

```
[root@servera ~]# stratis filesystem create stratispool1 stratis-
filesystem1
```

7.2. Verify the availability of `stratis-filesystem1` using the **stratis filesystem list** command.

```
[root@servera ~]# stratis filesystem list
Pool Name     Name                 Used     Created            Device
                                    UUID
stratispool1  stratis-filesystem1  546 MiB  Mar 29 2019 07:48  /
stratis/stratispool1/stratis-filesystem1  8714...e7db
```

Note the current usage of `stratis-filesystem1`. This usage of the file system increases on-demand in the following steps.

7.3. Create a directory named **/stratisvol** using the **mkdir** command.

```
[root@servera ~]# mkdir /stratisvol
```

7.4. Mount `stratis-filesystem1` on **/stratisvol** using the **mount** command.

```
[root@servera ~]# mount /stratis/stratispool1/stratis-filesystem1 /
stratisvol
```

7.5. Verify that the Stratis file system `stratis-filesystem1` is mounted on **/stratisvol** using the **mount** command.

```
[root@servera ~]# mount
...output omitted...
/dev/mapper/stratis-1-5c0e...12b9-thin-
fs-8714...e7db on /stratisvol type xfs
 (rw,relatime,seclabel,attr2,inode64,sunit=2048,swidth=2048,noquota)
```

7.6. Create the text file **/stratisvol/file1** using the **echo** command.

```
[root@servera ~]# echo "Hello World!" > /stratisvol/file1
```

8. Verify that the thin-provisioned file system `stratis-filesystem1` dynamically grows as the data on the file system grows.

8.1. View the current usage of `stratis-filesystem1` using the **`stratis filesystem list`** command.

```
[root@servera ~]# stratis filesystem list
Pool Name      Name                   Used      Created            Device
                                       UUID
stratispool1   stratis-filesystem1   546 MiB   Mar 29 2019 07:48   /
stratis/stratispool1/stratis-filesystem1   8714...e7db
```

8.2. Create a 2 GiB file on `stratis-filesystem1` using the **dd** command. It may take up to a minute for the command to complete.

```
[root@servera ~]# dd if=/dev/urandom of=/stratisvol/file2 bs=1M
 count=2048
```

8.3. Verify the usage of `stratis-filesystem1` using the **`stratis filesystem list`** command.

```
[root@servera ~]# stratis filesystem list
Pool Name      Name                   Used      Created            Device
                                        UUID
stratispool1   stratis-filesystem1   2.53 GiB   Mar 29 2019 07:48   /
stratis/stratispool1/stratis-filesystem1   8714...e7db
```

The preceding output shows that the usage of `stratis-filesystem1` has increased. The increase in the usage confirms that the thin-provisioned file system has dynamically expanded to accommodate the real-time data growth you caused by creating **`/stratisvol/file2`**.

9. Create a snapshot of `stratis-filesystem1` named `stratis-filesystem1-snap`. The snapshot will provide you with access to any file that is deleted from `stratis-filesystem1`.

9.1. Create a snapshot of `stratis-filesystem1` using the **`stratis filesystem snapshot`** command. It may take up to a minute for the command to complete.

```
[root@servera ~]# stratis filesystem snapshot stratispool1 \
stratis-filesystem1 stratis-filesystem1-snap
```

9.2. Verify the availability of the snapshot using the **`stratis filesystem list`** command.

```
[root@servera ~]# stratis filesystem list
...output omitted...
stratispool1  stratis-filesystem1-snap  2.53 GiB  Mar 29 2019 10:28  /
stratis/stratispool1/stratis-filesystem1-snap  291d...8a16
```

9.3. Remove the **`/stratisvol/file1`** file.

```
[root@servera ~]# rm /stratisvol/file1
rm: remove regular file '/stratisvol/file1'? y
```

9.4. Create the directory **`/stratisvol-snap`** using the **`mkdir`** command.

```
[root@servera ~]# mkdir /stratisvol-snap
```

9.5. Mount the `stratis-filesystem1-snap` snapshot on **`/stratisvol-snap`** using the **`mount`** command.

```
[root@servera ~]# mount /stratis/stratispool1/stratis-filesystem1-snap
 \
/stratisvol-snap
```

9.6. Confirm that you can still access the file you deleted from `stratis-filesystem1` using the `stratis-filesystem1-snap` snapshot.

```
[root@servera ~]# cat /stratisvol-snap/file1
Hello World!
```

10. Unmount **`/stratisvol`** and **`/stratisvol-snap`** using the **`umount`** command.

```
[root@servera ~]# umount /stratisvol-snap
[root@servera ~]# umount /stratisvol
```

11. Remove the thin-provisioned file system `stratis-filesystem1` and its snapshot `stratis-filesystem1-snap` from the system.

    11.1. Destroy `stratis-filesystem1-snap` using the **`stratis filesystem destroy`** command.

    ```
    [root@servera ~]# stratis filesystem destroy stratispool1 stratis-
    filesystem1-snap
    ```

    11.2. Destroy `stratis-filesystem1` using the **`stratis filesystem destroy`** command.

    ```
    [root@servera ~]# stratis filesystem destroy stratispool1 stratis-
    filesystem1
    ```

    11.3. Exit the `root` user's shell.

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

    11.4. Log off from `servera`.

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

Finish

On `workstation`, run **`lab advstorage-stratis finish`** to complete this exercise. This script deletes the partitions and files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab advstorage-stratis finish
```

This concludes the guided exercise.

COMPRESSING AND DEDUPLICATING STORAGE WITH VDO

OBJECTIVES

After completing this section, you should be able to optimize the use of storage space by using VDO to compress and deduplicate data on storage devices.

DESCRIBING VIRTUAL DATA OPTIMIZER

Red Hat Enterprise Linux 8 includes the Virtual Data Optimizer (VDO) driver, which optimizes the data footprint on block devices. VDO is a Linux device mapper driver that reduces disk space usage on block devices, and minimizes the replication of data, saving disk space and even increasing data throughput. VDO includes two kernel modules: the `kvdo` module to transparently control data compression, and the `uds` module for deduplication.

The VDO layer is placed on top of an existing block storage device, such as a RAID device or a local disk. Those block devices can also be encrypted devices. The storage layers, such as LVM logical volumes and file systems, are placed on top of a VDO device. The following diagram shows the placement of VDO in an infrastructure consisting of KVM virtual machines that are using optimized storage devices.

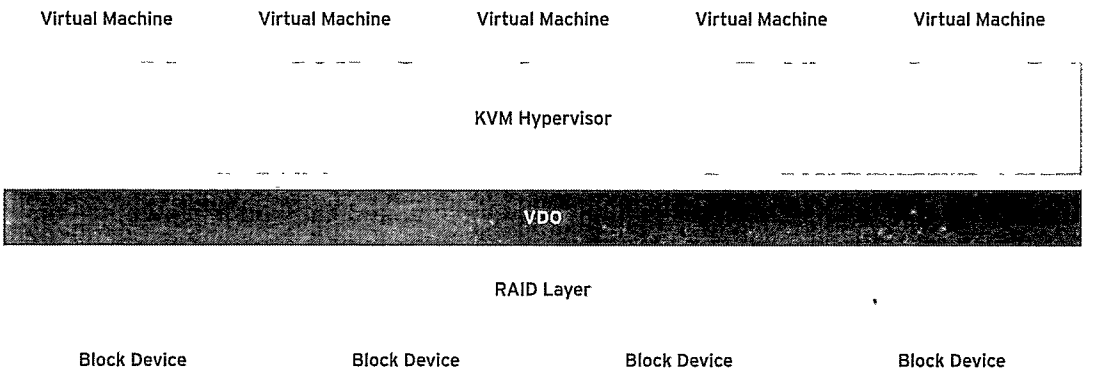


Figure 12.5: VDO-based virtual machines

VDO applies three phases to data in the following order to reduce the footprint on storage devices:

1. *Zero-Block Elimination* filters out data blocks that contain only zeroes (0) and records the information of those blocks only in the metadata. The nonzero data blocks are then passed to the next phase of processing. This phase enables the thin provisioning feature in the VDO devices.

2. *Deduplication* eliminates redundant data blocks. When you create multiple copies of the same data, VDO detects the duplicate data blocks and updates the metadata to use those duplicate blocks as references to the original data block without creating redundant data blocks. The universal deduplication service (UDS) kernel module checks redundancy of the data through the metadata it maintains. This kernel module ships as part of the VDO.

3. *Compression* is the last phase. The `kvdo` kernel module compresses the data blocks using LZ4 compression and groups them on 4 KB blocks.

IMPLEMENTING VIRTUAL DATA OPTIMIZER

The logical devices that you create using VDO are called *VDO volumes*. VDO volumes are similar to disk partitions; you can format the volumes with the desired file-system type and mount it like a regular file system. You can also use a VDO volume as an LVM physical volume.

To create a VDO volume, specify a block device and the name of the logical device that VDO presents to the user. You can optionally specify the logical size of the VDO volume. The logical size of the VDO volume can be more than the physical size of the actual block device.

Because the VDO volumes are thinly provisioned, users can only see the logical space in use and are unaware of the actual physical space available. If you do not specify the logical size while creating the volume, VDO assumes the actual physical size as the logical size of the volume. This 1:1 ratio of mapping logical size to physical size gives better performance but provides less efficient use of storage space. Based on your infrastructure requirements, you should prioritize either performance or space efficiency.

When the logical size of a VDO volume is more than the actual physical size, you should proactively monitor the volume statistics to view the actual usage using the **`vdostats --verbose`** command.

Enabling VDO

Install the *vdo* and *kmod-kvdo* packages to enable VDO in the system.

```
[root@host ~]# yum install vdo kmod-kvdo
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

Creating a VDO Volume

To create a VDO volume, run the **`vdo create`** command.

```
[root@host ~]# vdo create --name=vdo1 --device=/dev/vdd --vdoLogicalSize=50G
...output omitted...
```

If you omit the logical size, the resulting VDO volume gets the same size as its physical device.

When the VDO volume is in place, you can format it with the file-system type of your choice and mount it under the file-system hierarchy on your system.

Analyzing a VDO Volume

To analyze a VDO volume, run the **`vdo status`** command. This command displays a report on the VDO system, and the status of the VDO volume in YAML format. It also displays attributes of the VDO volume. Use the **`--name=`** option to specify the name of a particular volume. If you omit the name of the specific volume, the output of the **`vdo status`** command displays the status of all the VDO volumes.

```
[root@host ~]# vdo status --name=vdo1
...output omitted...
```

The **`vdo list`** command displays the list of VDO volumes that are currently started. You can start and stop a VDO volume using the **`vdo start`** and **`vdo stop`** commands, respectively.

REFERENCES

For more information, refer to the *Getting started with VDO* chapter in the *Red Hat Enterprise Linux 8 Deduplicating and Compressing Storage Guide* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/deduplicating_and_compressing_storage/

Introducing Virtual Data Optimizer
https://rhelblog.redhat.com/2018/04/11/introducing-virtual-data-optimizer-to-reduce-cloud-and-on-premise-storage-costs/

GUIDED EXERCISE

COMPRESSING AND DEDUPLICATING STORAGE WITH VDO

In this exercise, you will create a VDO volume, format it with a file system, mount it, store data on it, and investigate the impact of compression and deduplication on storage space actually used.

OUTCOMES

You should be able to:

- Create a volume using Virtual Data Optimizer, format it with a file-system type, and mount a file system on it.
- Investigate the impact of data deduplication and compression on a Virtual Data Optimizer volume.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab advstorage-vdo start`** to start the exercise. This script ensures that there are no partitions on the `/dev/vdd` disk and sets up the environment correctly.

```
[student@workstation ~]$ lab advstorage-vdo start
```

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

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

2. Create the VDO volume `vdo1`, using the `/dev/vdd` device. Set its logical size to 50 GB.

   2.1. Switch to the `root` user.

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

   2.2. Confirm that the *vdo* package is installed using the **`rpm`** command.

```
[root@servera ~]# yum list installed vdo
vdo-6.2.0.293-10.el8.x86_64
```

2.3. Create the vdo1 volume using the **vdo create** command.

```
[root@servera ~]# vdo create --name=vdo1 --device=/dev/vdd --
vdoLogicalSize=50G
...output omitted...
```

2.4. Verify the availability of the vdo1 volume using the **vdo list** command.

```
[root@servera ~]# vdo list
vdo1
```

3. Verify that the vdo1 volume has both the compression and deduplication features enabled.

3.1. Use **grep** to search for the lines containing the string **Deduplication** in the output of the **vdo status --name=vdo1** command.

```
[root@servera ~]# vdo status --name=vdo1 | grep Deduplication
    Deduplication: enabled
```

3.2. Use **grep** to search for the lines containing the string **Compression** in the output of the **vdo status --name=vdo1** command.

```
[root@servera ~]# vdo status --name=vdo1 | grep Compression
    Compression: enabled
```

4. Format the vdo1 volume with the XFS file-system type and mount it on **/mnt/vdo1**.

4.1. Format the vdo1 volume with the XFS file system using the **mkfs** command.

```
[root@servera ~]# mkfs.xfs -K /dev/mapper/vdo1
...output omitted...
```

The **-K** option in the preceding **mkfs.xfs** command prevents the unused blocks in the file system from being discarded immediately which lets the command return faster.

4.2. Use the **udevadm** command to register the new device node.

```
[root@servera ~]# udevadm settle
```

4.3. Create the **/mnt/vdo1** directory using the **mkdir** command.

```
[root@servera ~]# mkdir /mnt/vdo1
```

4.4. Mount the vdo1 volume on **/mnt/vdo1** using the **mount** command.

```
[root@servera ~]# mount /dev/mapper/vdo1 /mnt/vdo1
```

4.5. Verify that the vdo1 volume is successfully mounted using the **mount** command.

```
[root@servera ~]# mount
...output omitted...
/dev/mapper/vdo1 on /mnt/vdo1 type xfs
 (rw,relatime,seclabel,attr2,inode64,noquota)
```

5. Create three copies of the same file named **/root/install.img** on the vdo1 volume. Compare the statistics of the volume to verify the data deduplication and compression happening on the volume. The preceding output may vary on your system..

5.1. View the initial statistics and status of the volume using the **vdostats** command.

```
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1          5.0G      3.0G      2.0G  60%           99%
```

Notice that 3 GB of the volume is already used because when created, the VDO volume reserves 3-4 GB for itself. Also, note that the value **99%** in the **Space**

saving% field indicates that you have not created any content so far in the volume contributing to all of the saved volume space.

5.2. Copy **/root/install.img** to **/mnt/vdo1/install.img.1** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@servera ~]# cp /root/install.img /mnt/vdo1/install.img.1
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1          5.0G      3.4G      1.6G  68%           5%
```

Notice that the value of the **Used** field increased from **3.0G** to **3.4G** because you copied a file to the volume, and that occupies some space. Also, notice that the value of **Space saving%** field decreased from **99%** to **5%** because initially there was no content in the volume, contributing to the low volume space utilization and high volume space saving until you created a file. The volume space saving is considerably low because you created a unique copy of the file in the volume and there is nothing to deduplicate.

5.3. Copy **/root/install.img** to **/mnt/vdo1/install.img.2** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@servera ~]# cp /root/install.img /mnt/vdo1/install.img.2
[root@servera ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/vdo1          5.0G      3.4G      1.6G  68%          51%
```

Notice that the used volume space did not change rather the percentage of the saved volume space increased proving that the data deduplication occurred to reduce the space consumption for the redundant copies of the same file. The value of **Space saving%** in the preceding output may vary on your system.

5.4. Exit the `root` user's shell.

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

5.5. Log off from `servera`.

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

Finish

On `workstation`, run **lab advstorage-vdo finish** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab advstorage-vdo finish
```

This concludes the guided exercise.

▷ LAB

IMPLEMENTING ADVANCED STORAGE FEATURES

In this exercise, you will use the Stratis storage management solution to create file systems that grow to accommodate increased data demands, and Virtual Data Optimizer to create volumes for efficient utilization of storage space.

OUTCOMES

You should be able to:

- Create a thinly provisioned file system using Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thinly provisioned file system.
- Create a volume using Virtual Data Optimizer and mount it on a file system.
- Investigate the impact of data deduplication and compression on a Virtual Data Optimizer volume.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab advstorage-review start`** to start the lab. This script sets up the environment correctly and ensures that the additional disks on `serverb` are clean.

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

1. From `workstation`, open an SSH session to `serverb` as `student`.
2. Switch to the `root` user.
3. Install the *stratisd* and *stratis-cli* packages using **`yum`**.
4. Start and enable the `stratisd` service using the **`systemctl`** command.
5. Create the Stratis pool `labpool` containing the block device `/dev/vdb`.
6. Expand the capacity of `labpool` using the disk `/dev/vdc` available in the system.
7. Create a thinly provisioned file system named `labfs` in the `labpool` pool. Mount this file system on **`/labstratisvol`** so that it persists across reboots. Create a file named **`labfile1`** that contains the text **`Hello World!`** on the `labfs` file system. Don't forget to use the `x-systemd.requires=stratisd.service` mount option in **`/etc/fstab`**.
8. Verify that the thinly provisioned file system `labfs` dynamically grows as the data on the file system grows.
9. Create a snapshot named `labfs-snap` of the `labfs` file system. The snapshot allows you to access any file that is deleted from `labfs`.

10. Create the VDO volume `labvdo`, with the device `/dev/vdd`. Set its logical size to **`50 GB`**.
11. Mount the volume `labvdo` on **`/labvdovol`** with the XFS file system so that it persists across reboots. Don't forget to use the `x-systemd.requires=vdo.service` mount option in **`/etc/fstab`**.
12. Create three copies of the file named **`/root/install.img`** on the volume `labvdo`. Compare the statistics of the volume to verify the data deduplication and compression happening on the volume.

Evaluation

On `workstation`, run the **`lab advstorage-review grade`** command to confirm success of this exercise.

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

Finish

On `workstation`, run **`lab advstorage-review finish`** to complete this exercise. This script deletes the partitions and files created throughout the exercise and ensures that the environment is clean.

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

This concludes the lab.

▷ SOLUTION

IMPLEMENTING ADVANCED STORAGE FEATURES

In this exercise, you will use the Stratis storage management solution to create file systems that grow to accommodate increased data demands, and Virtual Data Optimizer to create volumes for efficient utilization of storage space.

OUTCOMES

You should be able to:

- Create a thinly provisioned file system using Stratis storage management solution.
- Verify that the Stratis volumes grow dynamically to support real-time data growth.
- Access data from the snapshot of a thinly provisioned file system.
- Create a volume using Virtual Data Optimizer and mount it on a file system.
- Investigate the impact of data deduplication and compression on a Virtual Data Optimizer volume.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab advstorage-review start`** to start the lab. This script sets up the environment correctly and ensures that the additional disks on `serverb` are clean.

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

1. From `workstation`, open an SSH session to `serverb` as `student`.

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

2. Switch to the `root` user.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3. Install the *stratisd* and *stratis-cli* packages using **`yum`**.

```
[root@serverb ~]# yum install stratisd stratis-cli
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

4. Start and enable the `stratisd` service using the **`systemctl`** command.

```
[root@serverb ~]# systemctl enable --now stratisd
```

5. Create the Stratis pool `labpool` containing the block device `/dev/vdb`.

   5.1. Create the Stratis pool `labpool` using the **`stratis pool create`** command.

```
[root@serverb ~]# stratis pool create labpool /dev/vdb
```

   5.2. Verify the availability of `labpool` using the **`stratis pool list`** command.

```
[root@serverb ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
labpool                  5 GiB               52 MiB
```

   Note the size of the pool in the preceding output.

6. Expand the capacity of `labpool` using the disk `/dev/vdc` available in the system.

   6.1. Add the block device `/dev/vdc` to `labpool` using the **`stratis pool add-data`** command.

```
[root@serverb ~]# stratis pool add-data labpool /dev/vdc
```

   6.2. Verify the size of `labpool` using the **`stratis pool list`** command.

```
[root@serverb ~]# stratis pool list
Name            Total Physical Size  Total Physical Used
labpool                 10 GiB               56 MiB
```

   The preceding output shows that the size of `labpool` has increased after a new disk was added to the pool.

   6.3. Use the **`stratis blockdev list`** command to list the block devices that are now members of `labpool`.

```
[root@serverb ~]# stratis blockdev list labpool
Pool Name     Device Node    Physical Size   State  Tier
labpool  /dev/vdb               5 GiB  In-use  Data
labpool  /dev/vdc               5 GiB  In-use  Data
```

7. Create a thinly provisioned file system named `labfs` in the `labpool` pool. Mount this file system on **`/labstratisvol`** so that it persists across reboots. Create a file named

labfile1 that contains the text **Hello World!** on the labfs file system. Don't forget to use the x-systemd.requires=stratisd.service mount option in **/etc/fstab**.

7.1. Create the thinly provisioned file system labfs in labpool using the **stratis filesystem create** command. It may take up to a minute for the command to complete.

```
[root@serverb ~]# stratis filesystem create labpool labfs
```

7.2. Verify the availability of labfs using the **stratis filesystem list** command.

```
[root@serverb ~]# stratis filesystem list
Pool Name     Name                 Used     Created            Device
                                   UUID
labpool  labfs  546 MiB  Mar 29 2019 07:48  /stratis/labpool/labfs
 9825...d6ca
```

Note the current usage of labfs. This usage of the file system increases on-demand in the following steps.

7.3. Determine the UUID of labfs using the **lsblk** command.

```
[root@serverb ~]# lsblk --output=UUID /stratis/labpool/labfs
UUID
9825e289-fb08-4852-8290-44d1b8f0d6ca
```

7.4. Edit **/etc/fstab** so that the thinly provisioned file system labfs is mounted at boot time. Use the UUID you determined in the preceding step. The following shows the line you should add to **/etc/fstab**. You can use the **vi /etc/fstab** command to edit the file.

```
UUID=9825...d6ca /labstratisvol  xfs defaults,x-
systemd.requires=stratisd.service 0 0
```

7.5. Create a directory named **/labstratisvol** using the **mkdir** command.

```
[root@serverb ~]# mkdir /labstratisvol
```

7.6. Mount the thinly provisioned file system labfs using the **mount** command to confirm that the **/etc/fstab** file contains the appropriate entries.

```
[root@serverb ~]# mount /labstratisvol
```

If the preceding command produces any errors, revisit the **/etc/fstab** file and ensure that it contains the appropriate entries.

7.7. Create a text file named **/labstratisvol/labfile1** using the **echo** command.

```
[root@serverb ~]# echo "Hello World!" > /labstratisvol/labfile1
```

8. Verify that the thinly provisioned file system `labfs` dynamically grows as the data on the file system grows.

8.1. View the current usage of `labfs` using the **`stratis filesystem list`** command.

```
[root@serverb ~]# stratis filesystem list
Pool Name      Name                  Used      Created          Device
                                     UUID
labpool  labfs  546 MiB  Mar 29 2019 07:48  /stratis/labpool/labfs
 9825...d6ca
```

8.2. Create a 2 GiB file in `labfs` using the **dd** command. It may take up to a minute for the command to complete.

```
[root@serverb ~]# dd if=/dev/urandom of=/labstratisvol/labfile2 bs=1M
 count=2048
```

8.3. Verify that the usage of `labfs` has increased, using the **`stratis filesystem list`** command.

```
[root@serverb ~]# stratis filesystem list
Pool Name      Name                  Used      Created            Device
                                      UUID
labpool  labfs  2.53 GiB  Mar 29 2019 07:48  /stratis/labpool/labfs
 9825...d6ca
```

9. Create a snapshot named `labfs-snap` of the `labfs` file system. The snapshot allows you to access any file that is deleted from `labfs`.

9.1. Create a snapshot of `labfs` using the **`stratis filesystem snapshot`** command. It may take up to a minute for the command to complete.

```
[root@serverb ~]# stratis filesystem snapshot labpool \
labfs labfs-snap
```

9.2. Verify the availability of the snapshot using the **`stratis filesystem list`** command.

```
[root@serverb ~]# stratis filesystem list
...output omitted...
labpool  labfs-snap  2.53 GiB  Mar 29 2019 10:28  /stratis/labpool/
labfs-snap  291d...8a16
```

9.3. Remove the **`/labstratisvol/labfile1`** file.

```
[root@serverb ~]# rm /labstratisvol/labfile1
rm: remove regular file '/labstratisvol/labfile1'? y
```

9.4. Create the **`/labstratisvol-snap`** directory using the **`mkdir`** command.

```
[root@serverb ~]# mkdir /labstratisvol-snap
```

9.5. Mount the snapshot `labfs-snap` on **`/labstratisvol-snap`** using the **`mount`** command.

```
[root@serverb ~]# mount /stratis/labpool/labfs-snap \
/labstratisvol-snap
```

9.6. Confirm that you can still access the file you deleted from `labfs` using the snapshot `labfs-snap`.

```
[root@serverb ~]# cat /labstratisvol-snap/labfile1
Hello World!
```

10. Create the VDO volume `labvdo`, with the device `/dev/vdd`. Set its logical size to **`50 GB`**.

10.1. Create the volume `labvdo` using the **`vdo create`** command.

```
[root@serverb ~]# vdo create --name=labvdo --device=/dev/vdd --
vdoLogicalSize=50G
...output omitted...
```

10.2. Verify the availability of the volume `labvdo` using the **`vdo list`** command.

```
[root@serverb ~]# vdo list
labvdo
```

11. Mount the volume `labvdo` on **`/labvdovol`** with the XFS file system so that it persists across reboots. Don't forget to use the `x-systemd.requires=vdo.service` mount option in **`/etc/fstab`**.

11.1. Format the `labvdo` volume with the XFS file system using the **`mkfs`** command.

```
[root@serverb ~]# mkfs.xfs -K /dev/mapper/labvdo
...output omitted...
```

11.2. Use the **`udevadm`** command to register the new device node.

```
[root@serverb ~]# udevadm settle
```

11.3. Create the **`/labvdovol`** directory using the **`mkdir`** command.

```
[root@serverb ~]# mkdir /labvdovol
```

11.4. Determine the UUID of `labvdo` using the **`lsblk`** command.

```
[root@serverb ~]# lsblk --output=UUID /dev/mapper/labvdo
UUID
ef8cce71-228a-478d-883d-5732176b39b1
```

11.5. Edit **`/etc/fstab`** so that `labvdo` is mounted at boot time. Use the UUID of the volume you determined in the preceding step. The following shows the line you should add to **`/etc/fstab`**. You can use the **`vi /etc/fstab`** command to edit the file.

```
UUID=ef8c...39b1 /labvdovol xfs defaults,x-systemd.requires=vdo.service
 0 0
```

11.6. Mount the `labvdo` volume using the **`mount`** command to confirm that the **`/etc/fstab`** file contains the appropriate entries.

```
[root@serverb ~]# mount /labvdovol
```

If the preceding command produces any errors, revisit the **`/etc/fstab`** file and ensure that it contains the appropriate entries.

12. Create three copies of the file named **`/root/install.img`** on the volume `labvdo`. Compare the statistics of the volume to verify the data deduplication and compression happening on the volume.

12.1. View the initial statistics and status of the volume using the **`vdostats`** command.

```
[root@serverb ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/labvdo          5.0G      3.0G      2.0G  60%           99%
```

Notice that 3 GB of the volume is already used because when created, the VDO volume reserves 3-4 GB for itself. Also note that the value **99%** in the **Space**

saving% field indicates that you have not created any content so far in the volume, contributing to all of the saved volume space.

12.2. Copy **/root/install.img** to **/labvdovol/install.img.1** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@serverb ~]# cp /root/install.img /labvdovol/install.img.1
[root@serverb ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/labvdo          5.0G      3.4G      1.6G  68%           5%
```

Notice that the value of the **Used** field increased from **3.0G** to **3.4G** because you copied a file in the volume, and that occupies some space. Also, notice that the value of **Space saving%** field decreased from **99%** to **5%** because initially there was no content in the volume, contributing to the low volume space utilization and high volume space saving until you created a file in there. The volume space saving is quite low because you created a unique copy of the file in the volume and there is nothing to deduplicate.

12.3. Copy **/root/install.img** to **/labvdovol/install.img.2** and verify the statistics of the volume. It may take up to a minute to copy the file.

```
[root@serverb ~]# cp /root/install.img /labvdovol/install.img.2
[root@serverb ~]# vdostats --human-readable
Device                    Size      Used Available Use% Space saving%
/dev/mapper/labvdo          5.0G      3.4G      1.6G  68%          51%
```

Notice that the used volume space did not change. Instead, the percentage of the saved volume space increased, proving that the data deduplication occurred to reduce the space consumption for the redundant copies of the same file. The value of **Space saving%** in the preceding output may vary on your system.

12.4. Reboot the serverb machine.

```
[root@serverb ~]# systemctl reboot
```

NOTE

Note: If on a reboot, serverb doesn't boot to a regular login prompt but instead has "Give root password for maintenance (or press **Control-D** to continue):" you likely made a mistake in **/etc/fstab**. After providing the root password of **redhat**, you will need to remount the root file system as read-write with:

```
[root@serverb ~]# mount -o remount,rw /
```

Verify that **/etc/fstab** is configured correctly as specified in the solutions. Pay special attention to the mount options for the lines related to **/labstratisvol** and **/labvdovol**.

Evaluation

On `workstation`, run the **lab advstorage-review grade** command to confirm success of this exercise.

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

Finish

On `workstation`, run **`lab advstorage-review finish`** to complete this exercise. This script deletes the partitions and files created throughout the exercise and ensures that the environment is clean.

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

This concludes the lab.

SUMMARY

In this chapter, you learned:

- Physical volumes, volume groups, and logical volumes are managed by a variety of tools such as pvcreate, vgreduce, and lvextend.
- Logical volumes can be formatted with a file system or swap space, and they can be mounted persistently.
- Additional storage can be added to volume groups and logical volumes can be extended dynamically.
- The Stratis volume-management solution implements flexible file systems that grow dynamically with data.
- Stratis volume-management solution supports thin provisioning, snapshotting, and monitoring.
- The Virtual Data Optimizer (VDO) aims to reduce the cost of data storage.
- The Virtual Data Optimizer applies zero-block eliminiation, data deduplication, and data compression to optimize disk space efficiency.

CHAPTER 13

SCHEDULING FUTURE TASKS

GOAL	Schedule tasks to automatically execute in the future.
OBJECTIVES	• Schedule commands to run on a repeating schedule using the system crontab file and directories.. • Enable and disable systemd timers, and configure a timer that manages temporary files
SECTIONS	• Scheduling Recurring System Jobs (and Guided Exercise) • Managing Temporary Files (and Guided Exercise)
LAB	Scheduling Future Tasks

SCHEDULING RECURRING SYSTEM JOBS

OBJECTIVES

After completing this section, you should be able to schedule commands to run on a repeating schedule using the system crontab file and directories.

DESCRIBING RECURRING SYSTEM JOBS

System administrators often need to run recurring jobs. Best practice is to run these jobs from system accounts rather than from user accounts. That is, do not schedule to run these jobs using the **crontab** command, but instead use system-wide crontab files. Job entries in the system-wide crontab files are similar to those of the users' crontab entries, excepting only that the system-wide crontab files have an extra field before the command field; the user under whose authority the command should run.

The **/etc/crontab** file has a useful syntax diagram in the included comments.

```
 # For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue ...
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed
```

Recurring system jobs are defined in two locations: the **/etc/crontab** file, and files within the **/etc/cron.d/** directory. You should always create your custom crontab files under the **/etc/cron.d** directory to schedule recurring system jobs. Place the custom crontab file in **/etc/cron.d** to protect it from being overwritten if any package update occurs to the provider of **/etc/crontab**, which may overwrite the existing contents in **/etc/crontab**. Packages that require recurring system jobs place their crontab files in **/etc/cron.d/** containing the job entries. Administrators also use this location to group related jobs into a single file.

The crontab system also includes repositories for scripts that need to run every hour, day, week, and month. These repositories are directories called **/etc/cron.hourly/**, **/etc/cron.daily/**, **/etc/cron.weekly/**, and **/etc/cron.monthly/**. Again, these directories contain executable shell scripts, not crontab files.

IMPORTANT

Remember to make any script you place in these directories executable. If a script is not executable, it will not run. To make a script executable, use the **chmod +x** ***script_name***.

A command called **run-parts** called from the **/etc/cron.d/0hourly** file runs the **/etc/cron.hourly/*** scripts. The **run-parts** command also runs the daily, weekly, and monthly jobs, but it is called from a different configuration file called **/etc/anacrontab**.

NOTE

In the past, a separate service called anacron used to handle the **/etc/anacrontab** file, but in Red Hat Enterprise Linux 7 and later, the regular crond service parses this file.

The purpose of **/etc/anacrontab** is to make sure that important jobs always run, and not skipped accidentally because the system was turned off or hibernating when the job should have been executed. For example, if a system job that runs daily was not executed last time it was due because the system was rebooting, the job is executed when the system becomes ready. However, there may be a delay of several minutes in starting the job depending on the value of the **Delay in minutes** parameter specified for the job in **/etc/anacrontab**.

There are different files in **/var/spool/anacron/** for each of the daily, weekly, and monthly jobs to determine if a particular job has run. When crond starts a job from **/etc/anacrontab**, it updates the time stamps of those files. The same time stamp is used to determine when a job was last run. The syntax of **/etc/anacrontab** is different from the regular **crontab** configuration files. It contains exactly four fields per line, as follows.

- **Period in days**

  The interval in days for the job that runs on a repeating schedule. This field accepts an integer or a macro as its value. For example, the macro **@daily** is equivalent to the integer **1**, which means that the job is executed on a daily basis. Similarly, the macro **@weekly** is equivalent to the integer **7**, which means that the job is executed on a weekly basis.

- **Delay in minutes**

  The amount of time the crond daemon should wait before starting this job.

- **Job identifier**

  The unique name the job is identified as in the log messages.

- **Command**

  The command to be executed.

The **/etc/anacrontab** file also contains environment variable declarations using the syntax **NAME=value**. Of special interest is the variable START_HOURS_RANGE, which specifies the time interval for the jobs to run. Jobs are not started outside of this range. If on a particular day, a job does not run within this time interval, the job has to wait until the next day for execution.

INTRODUCING SYSTEMD TIMER

With the advent of systemd in Red Hat Enterprise Linux 7, a new scheduling function is now available: systemd *timer units*. A systemd timer unit activates another unit of a different type (such as a service) whose unit name matches the timer unit name. The timer unit allows timer-based activation of other units. For easier debugging, systemd logs timer events in system journals.

Sample Timer Unit

The *sysstat* package provides a `systemd` timer unit called `sysstat-collect.timer` to collect system statistics every 10 minutes. The following output shows the configuration lines of **`/usr/lib/systemd/system/sysstat-collect.timer`**.

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

[Timer]
OnCalendar=*:00/10

[Install]
WantedBy=sysstat.service
```

The parameter **`OnCalendar=*:00/10`** signifies that this timer unit activates the corresponding unit (`sysstat-collect.service`) every 10 minutes. However, you can specify more complex time intervals. For example, a value of **`2019-03-* 12:35,37,39:16`** against the **`OnCalendar`** parameter causes the timer unit to activate the corresponding service unit at **`12:35:16`**, **`12:37:16`**, and **`12:39:16`** every day throughout the entire month of March, 2019. You can also specify relative timers using parameters such as **`OnUnitActiveSec`**. For example, the **`OnUnitActiveSec=15min`** option causes the timer unit to trigger the corresponding unit 15 minutes after the last time the timer unit activated its corresponding unit.

IMPORTANT

Do not modify any unit configuration file under the **`/usr/lib/systemd/system`** directory because any update to the provider package of the configuration file may override the configuration changes you made in that file. So, make a copy of the unit configuration file you intend to change under the **`/etc/systemd/system`** directory and then modify the copy so that the configuration changes you make with respect to a unit does not get overridden by any update to the provider package. If two files exist with the same name under the **`/usr/lib/systemd/system`** and **`/etc/systemd/system`** directories, `systemd` parses the file under the **`/etc/systemd/system`** directory.

After you change the timer unit configuration file, use the **`systemctl daemon-reload`** command to ensure that `systemd` is aware of the changes. This command reloads the `systemd` manager configuration.

```
[root@host ~]# systemctl daemon-reload
```

After you reload the `systemd` manager configuration, use the following **`systemctl`** command to activate the timer unit.

```
[root@host ~]# systemctl enable --now <unitname>.timer
```

REFERENCES

`crontab`(5), **`anacron`**(8), **`anacrontab`**(5), `systemd.time`(7), `systemd.timer`(5), and `crond`(8) man pages

GUIDED EXERCISE

SCHEDULING RECURRING SYSTEM JOBS

In this exercise, you will schedule commands to run on various schedules by adding configuration files to the system crontab directories.

OUTCOMES

You should be able to:

- Schedule a recurring system job to count the number of active users.
- Update the `systemd` timer unit that gathers system activity data.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab scheduling-system start`** to start the exercise. This script ensures that the environment is clean and set up correctly.

```
[student@workstation ~]$ lab scheduling-system start
```

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

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

2. Use the **`sudo -i`** command to switch to the `root` user's account.

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

3. Schedule a recurring system job that generates a log message indicating the number of currently active users in the system. The job must run daily. You can use the **`w -h | wc -l`** command to retrieve the number of currently active users in the system. Also, use the **`logger`** command to generate the log message.

   3.1. Create a script file called **`/etc/cron.daily/usercount`** with the following content. You can use the **`vi /etc/cron.daily/usercount`** command to create the script file.

```
#!/bin/bash
USERCOUNT=$(w -h | wc -l)
logger "There are currently ${USERCOUNT} active users"
```

3.2. Use the **chmod** command to enable the execute (**x**) permission on **/etc/cron.daily/usercount**.

```
[root@servera ~]# chmod +x /etc/cron.daily/usercount
```

4. The *sysstat* package provides the `systemd` units called `sysstat-collect.timer` and `sysstat-collect.service`. The timer unit triggers the service unit every 10 minutes to collect system activity data using the shell script called **/usr/lib64/sa/sa1**. Make sure that the *sysstat* package is installed and change the timer unit configuration file to collect the system activity data every two minutes.

4.1. Use the **yum** command to install the *sysstat* package.

```
[root@servera ~]# yum install sysstat
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  sysstat-11.7.3-2.el8.x86_64            lm_sensors-
libs-3.4.0-17.20180522git70f7e08.el8.x86_64

Complete!
```

4.2. Copy **/usr/lib/systemd/system/sysstat-collect.timer** to **/etc/systemd/system/sysstat-collect.timer**.

```
[root@servera ~]# cp /usr/lib/systemd/system/sysstat-collect.timer \
/etc/systemd/system/sysstat-collect.timer
```

IMPORTANT

You should not edit files under the **/usr/lib/systemd** directory. With `systemd`, you can copy the unit file to the **/etc/systemd/system** directory and edit that copy. The `systemd` process parses your customized copy instead of the file under the **/usr/lib/systemd** directory.

4.3. Edit **/etc/systemd/system/sysstat-collect.timer** so that the timer unit runs every two minutes. Also, replace any occurrence of the string **10 minutes** with **2 minutes** throughout the unit configuration file including the ones in the commented lines. You may use the **vi /etc/systemd/system/sysstat-collect.timer** command to edit the configuration file.

```
...
#        Activates activity collector every 2 minutes

[Unit]
```

```
Description=Run system activity accounting tool every 2 minutes

[Timer]
OnCalendar=*:00/02

[Install]
WantedBy=sysstat.service
```

The preceding changes cause the `sysstat-collect.timer` unit to trigger `sysstat-collect.service` unit every two minutes, which runs **`/usr/lib64/sa/sa1 1 1`**. Running **`/usr/lib64/sa/sa1 1 1`** collects the system activity data in a binary file under the **`/var/log/sa`** directory.

4.4. Use the **`systemctl daemon-reload`** command to make sure that `systemd` is aware of the changes.

```
[root@servera ~]# systemctl daemon-reload
```

4.5. Use the **`systemctl`** command to activate the `sysstat-collect.timer` timer unit.

```
[root@servera ~]# systemctl enable --now sysstat-collect.timer
```

4.6. Use the **`while`** command to wait until the binary file gets created under the **`/var/log/sa`** directory. Wait for your shell prompt to return.

```
[root@servera ~]# while [ $(ls /var/log/sa | wc -l) -eq 0 ]; do sleep
 1s; done
```

In the **`while`** command above, the **`ls /var/log/sa | wc -l`** returns a **`0`** if the file does not exist and a **`1`** if it does exist. The **`while`** determines if this equals **`0`** and

if so, enters the loop, which pauses for one second. When the file exists, the **while** loop exits.

4.7. Use the **ls -l** command to verify that the binary file under the **/var/log/sa** directory got modified within last two minutes.

```
[root@servera ~]# ls -l /var/log/sa
total 8
-rw-r--r--. 1 root root 5156 Mar 25 12:34 sa25
[root@servera ~]# date
Mon Mar 25 12:35:32 +07 2019
```

The output of the preceding commands may vary on your system.

4.8. Exit the `root` user's shell.

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

4.9. Log off from `servera`.

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

Finish

On `workstation`, run **lab scheduling-system finish** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab scheduling-system finish
```

This concludes the guided exercise.

MANAGING TEMPORARY FILES

OBJECTIVES

After completing this section, you should be able to enable and disable `systemd` timers, and configure a timer that manages temporary files.

MANAGING TEMPORARY FILES

A modern system requires a large number of temporary files and directories. Some applications (and users) use the **`/tmp`** directory to hold temporary data, while others use a more task-specific location such as daemon and user-specific *volatile* directories under **`/run`**. In this context, volatile means that the file system storing these files only exists in memory. When the system reboots or loses power, all the contents of volatile storage will be gone.

To keep a system running cleanly, it is necessary for these directories and files to be created when they do not exist, because daemons and scripts might rely on them being there, and for old files to be purged so that they do not fill up disk space or provide faulty information.

Red Hat Enterprise Linux 7 and later include a new tool called **`systemd-tmpfiles`**, which provides a structured and configurable method to manage temporary directories and files.

When `systemd` starts a system, one of the first service units launched is `systemd-tmpfiles-setup`. This service runs the command **`systemd-tmpfiles --create --remove`**. This command reads configuration files from **`/usr/lib/tmpfiles.d/*.conf`**, **`/run/tmpfiles.d/*.conf`**, and **`/etc/tmpfiles.d/*.conf`**. Any files and directories marked for deletion in those configuration files is removed, and any files and directories marked for creation (or permission fixes) will be created with the correct permissions if necessary.

Cleaning Temporary Files with a Systemd Timer

To ensure that long-running systems do not fill up their disks with stale data, a `systemd` timer unit called `systemd-tmpfiles-clean.timer` triggers `systemd-tmpfiles-clean.service` on a regular interval, which executes the **`systemd-tmpfiles --clean`** command.

The **`systemd`** timer unit configuration files have a **`[Timer]`** section that indicates how often the service with the same name should be started.

Use the following **`systemctl`** command to view the contents of the **`systemd-tmpfiles-clean.timer`** unit configuration file.

```
[user@host ~]$ systemctl cat systemd-tmpfiles-clean.timer
# /usr/lib/systemd/system/systemd-tmpfiles-clean.timer
#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published
#  by
#  the Free Software Foundation; either version 2.1 of the License, or
```

```
#  (at your option) any later version.

[Unit]
Description=Daily Cleanup of Temporary Directories
Documentation=man:tmpfiles.d(5) man:systemd-tmpfiles(8)

[Timer]
OnBootSec=15min
```

In the preceding configuration the parameter **OnBootSec=15min** indicates that the service unit called `systemd-tmpfiles-clean.service` gets triggered 15 minutes after the system has booted up. The parameter **OnUnitActiveSec=1d** indicates that any further trigger to the `systemd-tmpfiles-clean.service` service unit happens 24 hours after the service unit was activated last.

Based on your requirement, you can change the parameters in the **systemd-tmpfiles-clean.timer** timer unit configuration file. For example, the value **30min** for the parameter **OnUnitActiveSec** triggers the `systemd-tmpfiles-clean.service` service unit 30 minutes after the service unit was last activated. As a result, `systemd-tmpfiles-clean.service` gets triggered every 30 minutes after bringing the changes into effect.

After changing the timer unit configuration file, use the **systemctl daemon-reload** command to ensure that `systemd` is aware of the change. This command reloads the `systemd` manager configuration.

```
[root@host ~]# systemctl daemon-reload
```

After you reload the `systemd` manager configuration, use the following **systemctl** command to activate the `systemd-tmpfiles-clean.timer` unit.

```
[root@host ~]# systemctl enable --now systemd-tmpfiles-clean.timer
```

Cleaning Temporary Files Manually

The command **systemd-tmpfiles --clean** parses the same configuration files as the **systemd-tmpfiles --create** command, but instead of creating files and directories, it will purge all files which have not been accessed, changed, or modified more recently than the maximum age defined in the configuration file.

The format of the configuration files for **systemd-tmpfiles** is detailed in the **tmpfiles.d**(5) manual page. The basic syntax consists of seven columns: Type, Path, Mode, UID, GID, Age, and Argument. *Type* refers to the action that **systemd-tmpfiles** should take; for example, **d** to create a directory if it does not yet exist, or **Z** to recursively restore SELinux contexts and file permissions and ownership.

The following are some examples with explanations.

```
d /run/systemd/seats 0755 root root -
```

When creating files and directories, create the **/run/systemd/seats** directory if it does not yet exist, owned by the user `root` and the group `root`, with permissions set to **rwxr-xr-x**. This directory will not be automatically purged.

```
D /home/student 0700 student student 1d
```

Create the **/home/student** directory if it does not yet exist. If it does exist, empty it of all contents. When **systemd-tmpfiles --clean** is run, remove all files which have not been accessed, changed, or modified in more than one day.

```
L /run/fstablink - root root - /etc/fstab
```

Create the symbolic link **/run/fstablink** pointing to **/etc/fstab**. Never automatically purge this line.

Configuration File Precedence

Configuration files can exist in three places:

- **/etc/tmpfiles.d/*.conf**
- **/run/tmpfiles.d/*.conf**
- **/usr/lib/tmpfiles.d/*.conf**

The files in **/usr/lib/tmpfiles.d/** are provided by the relevant RPM packages, and you should not edit these files. The files under **/run/tmpfiles.d/** are themselves volatile files, normally used by daemons to manage their own runtime temporary files. The files under **/etc/tmpfiles.d/** are meant for administrators to configure custom temporary locations, and to override vendor-provided defaults.

If a file in **/run/tmpfiles.d/** has the same file name as a file in **/usr/lib/tmpfiles.d/**, then the file in **/run/tmpfiles.d/** is used. If a file in **/etc/tmpfiles.d/** has the same file name as a file in either **/run/tmpfiles.d/** or **/usr/lib/tmpfiles.d/**, then the file in **/etc/tmpfiles.d/** is used.

Given these precedence rules, you can easily override vendor-provided settings by *copying* the relevant file to **/etc/tmpfiles.d/**, and then editing it. Working in this fashion ensures that administrator-provided settings can be easily managed from a central configuration management system, and not be overwritten by an update to a package.

NOTE
When testing new or modified configurations, it can be useful to only apply the commands from one configuration file. This can be achieved by specifying the name of the configuration file on the command line.

REFERENCES
systemd-tmpfiles(8), **tmpfiles.d**(5), **stat**(1), **stat**(2), and **systemd.timer**(5) man pages

GUIDED EXERCISE

MANAGING TEMPORARY FILES

In this exercise, you will configure **systemd-tmpfiles** in order to change how quickly it removes temporary files from **/tmp**, and also to periodically purge files from another directory.

OUTCOMES

You should be able to:

- Configure **systemd-tmpfiles** to remove unused temporary files from **/tmp**.
- Configure **systemd-tmpfiles** to periodically purge files from another directory.

BEFORE YOU BEGIN

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

On workstation, run **lab scheduling-tempfiles start** to start the exercise. This script creates the necessary files and ensures that the environment is set up correctly.

```
[student@workstation ~]$ lab scheduling-tempfiles start
```

1. From workstation, open an SSH session to servera as student.

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

2. Configure **systemd-tmpfiles** to clean the **/tmp** directory so that it does not contain files that that have not been used in the last five days. Ensure that the configuration does not get overwritten by any package update.

   2.1. Use the **sudo -i** command to switch to the root user.

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

   2.2. Copy **/usr/lib/tmpfiles.d/tmp.conf** to **/etc/tmpfiles.d/tmp.conf**.

```
[root@servera ~]# cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d/
tmp.conf
```

   2.3. Search for the configuration line in **/etc/tmpfiles.d/tmp.conf** that applies to the **/tmp** directory. Replace the existing age of the temporary files in that configuration line with the new age of **5** days. Remove all the other lines from the

file including the commented ones. You can use the **vim /etc/tmpfiles.d/tmp.conf** command to edit the configuration file. The **/etc/tmpfiles.d/tmp.conf** file should appear as follows:

```
q /tmp 1777 root root 5d
```

The preceding configuration causes **systemd-tmpfiles** to ensure that the directory **/tmp** exists with the octal permissions set to **1777**. Both the owning user and group of **/tmp** must be `root`. The **/tmp** directory must be free from the temporary files which are unused in the last five days.

2.4. Use the **systemd-tmpfiles --clean** command to verify that the **/etc/tmpfiles.d/tmp.conf** file contains the correct configuration.

```
[root@servera ~]# systemd-tmpfiles --clean /etc/tmpfiles.d/tmp.conf
```

Because the preceding command did not return any errors, it confirms that the configuration settings are correct.

3. Add a new configuration that ensures that the **/run/momentary** directory exists with user and group ownership set to `root`. The octal permissions for the directory must be **0700**. The configuration should purge any file in this directory that remains unused in the last 30 seconds.

3.1. Create the file called **/etc/tmpfiles.d/momentary.conf** with the following content. You can use the **vim /etc/tmpfiles.d/momentary.conf** command to create the configuration file.

```
d /run/momentary 0700 root root 30s
```

The preceding configuration causes **systemd-tmpfiles** to ensure that the **/run/momentary** directory exists with its octal permissions set to **0700**. The user and group ownership of **/run/momentary** must be `root`. Any file in this directory that remains unused in the last 30 seconds must be purged.

3.2. Use the **systemd-tmpfiles --create** command to verify that the **/etc/tmpfiles.d/momentary.conf** file contains the appropriate configuration. The command creates the **/run/momentary** directory if it does not exist.

```
[root@servera ~]# systemd-tmpfiles --create /etc/tmpfiles.d/
momentary.conf
```

Because the preceding command did not return any errors, it confirms that the configuration settings are correct.

3.3. Use the **ls** command to verify that the **/run/momentary** directory is created with the appropriate permissions, owner, and group owner.

```
[root@servera ~]# ls -ld /run/momentary
drwx------. 2 root root 40 Mar 21 16:39 /run/momentary
```

Notice that the octal permission set of **/run/momentary** is **0700** and that the user and group ownership are set to `root`.

4. Verify that any file under the **/run/momentary** directory, unused in the last 30 seconds, is removed, based on the **systemd-tmpfiles** configuration for the directory.

   4.1. Use the **touch** command to create a file called **/run/momentary/testfile**.

   ```
   [root@servera ~]# touch /run/momentary/testfile
   ```

   4.2. Use the **sleep** command to configure your shell prompt not to return for 30 seconds.

   ```
   [root@servera ~]# sleep 30
   ```

   4.3. After your shell prompt returns, use the **systemd-tmpfiles --clean** command to clean stale files from **/run/momentary**, based on the rule mentioned in **/etc/tmpfiles.d/momentary.conf**.

   ```
   [root@servera ~]# systemd-tmpfiles --clean /etc/tmpfiles.d/
   momentary.conf
   ```

   The preceding command removes **/run/momentary/testfile** because the file remained unused for 30 seconds and should have been removed based on the rule mentioned in **/etc/tmpfiles.d/momentary.conf**.

   4.4. Use the **ls -l** command to verify that the **/run/momentary/testfile** file does not exist.

   ```
   [root@servera ~]# ls -l /run/momentary/testfile
   ls: cannot access '/run/momentary/testfile': No such file or directory
   ```

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

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

   4.6. Log off from `servera`.

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

Finish

On `workstation`, run **lab scheduling-tempfiles finish** to complete this exercise. This script deletes the files created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab scheduling-tempfiles finish
```

This concludes the guided exercise.

QUIZ

SCHEDULING FUTURE TASKS

Choose the correct answers to the following questions.

1. Which command displays all the user jobs that are currently scheduled to run as deferred jobs?
   a. `atq`
   b. `atrm`
   c. `at -c`
   d. `at --display`

2. Which command removes the deferred user job that has the job number 5?
   a. `at -c 5`
   b. `atrm 5`
   c. `at 5`
   d. `at --delete 5`

3. Which command displays all the recurring user jobs scheduled for the currently logged-in user?
   a. `crontab -r`
   b. `crontab -l`
   c. `crontab -u`
   d. `crontab -V`

4. Which job format executes `/usr/local/bin/daily_backup` hourly from 9 a.m. to 6 p.m. on all days from Monday through Friday?
   a. `00 * * * Mon-Fri /usr/local/bin/daily_backup`
   b. `* */9 * * Mon-Fri /usr/local/bin/daily_backup`
   c. `00 */18 * * * /usr/local/bin/daily_backup`
   d. `00 09-18 * * Mon-Fri /usr/local/bin/daily_backup`

5. Which directory contains the shell scripts intended to run on a daily basis?
   a. `/etc/cron.d`
   b. `/etc/cron.hourly`
   c. `/etc/cron.daily`
   d. `/etc/cron.weekly`

6. **Which configuration file defines the settings for the system jobs that run on a daily, weekly, and monthly basis?**
   a. **`/etc/crontab`**
   b. **`/etc/anacrontab`**
   c. **`/etc/inittab`**
   d. **`/etc/sysconfig/crond`**

7. **Which `systemd` unit regularly triggers the cleanup of the temporary files?**
   a. `systemd-tmpfiles-clean.timer`
   b. `systemd-tmpfiles-clean.service`
   c. `dnf-makecache.timer`
   d. `unbound-anchor.timer`

SOLUTION

SCHEDULING FUTURE TASKS

Choose the correct answers to the following questions.

1. **Which command displays all the user jobs that are currently scheduled to run as deferred jobs?**
   a. **`atq`**
   b. `atrm`
   c. `at -c`
   d. `at --display`

2. **Which command removes the deferred user job that has the job number 5?**
   a. `at -c 5`
   b. **`atrm 5`**
   c. `at 5`
   d. `at --delete 5`

3. **Which command displays all the recurring user jobs scheduled for the currently logged-in user?**
   a. `crontab -r`
   b. **`crontab -l`**
   c. `crontab -u`
   d. `crontab -V`

4. **Which job format executes `/usr/local/bin/daily_backup` hourly from 9 a.m. to 6 p.m. on all days from Monday through Friday?**
   a. `00 * * * Mon-Fri /usr/local/bin/daily_backup`
   b. `* */9 * * Mon-Fri /usr/local/bin/daily_backup`
   c. `00 */18 * * * /usr/local/bin/daily_backup`
   d. **`00 09-18 * * Mon-Fri /usr/local/bin/daily_backup`**

5. **Which directory contains the shell scripts intended to run on a daily basis?**
   a. `/etc/cron.d`
   b. `/etc/cron.hourly`
   c. **`/etc/cron.daily`**
   d. `/etc/cron.weekly`

6. **Which configuration file defines the settings for the system jobs that run on a daily, weekly, and monthly basis?**
   a. `/etc/crontab`
   b. **`/etc/anacrontab`**
   c. `/etc/inittab`
   d. `/etc/sysconfig/crond`

7. **Which `systemd` unit regularly triggers the cleanup of the temporary files?**
   a. `systemd-tmpfiles-clean.timer`
   b. `systemd-tmpfiles-clean.service`
   c. `dnf-makecache.timer`
   d. `unbound-anchor.timer`

SUMMARY

In this chapter, you learned:

- Recurring system jobs execute tasks on a repeating schedule.
- Recurring system jobs accomplish administrative tasks on a repeating schedule that have system-wide impact.
- The `systemd` timer units can execute the recurring jobs.

CHAPTER 14

ACCESSING NETWORK-ATTACHED STORAGE

GOAL	Access network-attached storage, using the NFS protocol.
OBJECTIVES	• Mount, use, and unmount an NFS export from the command line and at boot. • Configure the automater to automatically mount an NFS file system on demand, and unmount it when it is no longer in use.
SECTIONS	• Mounting Network-Attached Storage with NFS (and Guided Exercise) • Automounting Network-Attached Storage (and Guided Exercise)
LAB	Accessing Network-Attached Storage

MOUNTING NETWORK-ATTACHED STORAGE WITH NFS

OBJECTIVES

After completing this section, you should be able to:

- Identify NFS share information.
- Create a directory to use as a mount point.
- Mount an NFS share using the **mount** command or by configuring the **/etc/fstab** file.
- Unmount an NFS share using the **umount** command.
- Configure an NFS client to use NFSv4 using the new **nfsconf** tool.

MOUNTING AND UNMOUNTING NFS SHARES

NFS, the *Network File System*, is an internet standard protocol used by Linux, UNIX, and similar operating systems as their native network file system. It is an open standard, still being actively enhanced, which supports native Linux permissions and file-system features.

The default NFS version in Red Hat Enterprise Linux 8 is 4.2. NFSv4 and NFSv3 major versions are supported. NFSv2 is no longer supported. NFSv4 uses only the TCP protocol to communicate with the server; earlier NFS versions could use either TCP or UDP.

NFS servers *export* shares (directories). NFS clients mount an exported share to a local mount point (directory), which must exist. NFS shares can be mounted a number of ways:

- Manually, using the **mount** command.
- Automatically at boot time using **/etc/fstab** entries.
- On demand, using either the `autofs` service or the `systemd.automount` facility.

Mounting NFS Shares

To mount an NFS share, follow these three steps:

1. **Identify:** The administrator of the NFS client system can identify available NFS shares in various ways:

   The administrator for the NFS server may provide export details, including security requirements.

   Alternatively, the client administrator can identify NFSv4 shares by mounting the root directory of the NFS server and exploring the exported directories. Do this as the `root` user. Access to shares that use Kerberos security will be denied, but the share (directory) name will be visible. Other shared directories will be browsable.

   ```
   [user@host ~]$ sudo mkdir mountpoint
   [user@host ~]$ sudo mount serverb:/ mountpoint
   [user@host ~]$ sudo ls mountpoint
   ```

2. **Mount point:** Use **mkdir** to create a mount point in a suitable location.

```
[user@host ~]$ mkdir -p mountpoint
```

3. **Mount:** As with file systems on partitions, NFS shares must be mounted to be available. To mount an NFS share, select from the following. In each case, you must run these commands as a superuser either by logging in as `root` or by using the **sudo** command.

   - Mount temporarily: Mount the NFS share using the **mount** command:

```
[user@host ~]$ sudo mount -t nfs -o rw,sync serverb:/share mountpoint
```

   The **-t nfs** option is the file-system type for NFS shares (not strictly required but shown for completeness). The **-o sync** option tells **mount** to immediately synchronize write operations with the NFS server (the default is asynchronous).

   This command mounts the share immediately but not persistently; the next time the system boots, this NFS share will not be available. This is useful for one-time access to data. It is also useful for test mounting an NFS share before making the share available persistently.

   - Mount persistently: To ensure that the NFS share is mounted at boot time, edit the **/etc/fstab** file to add the mount entry.

```
[user@host ~]$ sudo vim /etc/fstab
...
serverb:/share  /mountpoint  nfs  rw,soft  0 0
```

   Then, mount the NFS share:

```
[user@host ~]$ sudo mount /mountpoint
```

   Because the NFS server and mount options are found in the **/etc/fstab** file by the NFS client service, you do not need to specify these on the command line.

Unmounting NFS Shares

As `root` (or using **sudo**), unmount an NFS share using the **umount** command.

```
[user@host ~]$ sudo umount mountpoint
```

NOTE

Unmounting a share does not remove its **/etc/fstab** entry. Unless you remove or comment out the entry, the NFS share will be remounted either at the next system boot or when the NFS client service is restarted.

THE nfsconf TOOL

Red Hat Enterprise Linux 8 introduces the **nfsconf** tool to manage the NFS client and server configuration files under NFSv4 and NFSv3. Configure the **nfsconf** tool using **/etc/nfs.conf**

(the **/etc/sysconfig/nfs** file from earlier versions of the operating system is deprecated now). Use the **nfsconf** tool to get, set, or unset NFS configuration parameters.

The **/etc/nfs.conf** configuration file is composed of multiple sections starting with a key word in square brackets (**[*keyword*]**) with value assignments within the section. For an NFS server, configure the **[nfsd]** section. A value assignment or key is composed of a name for the value, an equals sign, and a setting for the value, such as **vers4.2=y**. The lines starting with "#" or ";" are ignored, as are any blank lines.

```
[user@host ~]$ sudo cat /etc/nfs.conf
...output omitted...
[nfsd]
# debug=0
# threads=8
# host=
# port=0
# grace-time=90
# lease-time=90
# tcp=y
# vers2=n
# vers3=y
# vers4=y
# vers4.0=y
# vers4.1=y
# vers4.2=y
# rdma=n
#
```

By default the **[nfsd]** section's key-value pairs are commented out. However, the comments show the default options that will take effect if they are unchanged. This provides you with a good starting point for NFS configuration.

Use the **nfsconf --set *section key value*** to set a value for the key in the specified section.

```
[user@host ~]$ sudo nfsconf --set nfsd vers4.2 y
```

This command updates the **/etc/nfs.conf** configuration file:

```
[user@host ~]$ sudo cat /etc/nfs.conf
...output omitted...
[nfsd]
vers4.2 = y
# debug=0
# threads=8
# host=
# port=0
# grace-time=90
# lease-time=90
# tcp=y
# vers2=n
# vers3=y
# vers4=y
```

```
# vers4.0=y
# vers4.1=y
# vers4.2=y
# rdma=n
#
```

Use the **`nfsconf --get section key`** to retrieve the value for the key in the specified section:

```
[user@host ~]$ sudo nfsconf --get nfsd vers4.2
y
```

Use the **`nfsconf --unset section key`** to unset the value for the key in the specified section:

```
[user@host ~]$ sudo nfsconf --unset nfsd vers4.2
```

Configure an NFSv4-only Client

You can configure an NFSv4-only client by setting the following values on the **`/etc/nfs.conf`** configuration file.

Start by disabling UDP and other NFSv2 and NFSv3 related keys:

```
[user@host ~]$ sudo nfsconf --set nfsd udp n
[user@host ~]$ sudo nfsconf --set nfsd vers2 n
[user@host ~]$ sudo nfsconf --set nfsd vers3 n
```

Enable TCP, and NFSv4, related keys.

```
[user@host ~]$ sudo nfsconf --set nfsd tcp y
[user@host ~]$ sudo nfsconf --set nfsd vers4 y
[user@host ~]$ sudo nfsconf --set nfsd vers4.0 y
[user@host ~]$ sudo nfsconf --set nfsd vers4.1 y
[user@host ~]$ sudo nfsconf --set nfsd vers4.2 y
```

As before, the changes appear in the **`/etc/nfs.conf`** configuration file:

```
[[user@host ~]$ cat /etc/nfs.conf
[nfsd]
udp = n
vers2 = n
vers3 = n
tcp = y
vers4 = y
vers4.0 = y
vers4.1 = y
vers4.2 = y
```

REFERENCES

mount(8), **umount**(8), **fstab**(5), **mount.nfs**(8), **nfs.conf**(8) and **nfsconf**(8) man pages

GUIDED EXERCISE

MANAGING NETWORK-ATTACHED STORAGE WITH NFS

PERFORMANCE CHECKLIST

In this exercise, you will modify the **/etc/fstab** file to persistently mount an NFS export at boot time.

OUTCOMES

You should be able to:

- Test an NFS Server using the **mount** command.
- Configure NFS Shares in the **/etc/fstab** configuration file to save changes even after a system reboots.
- Configure NFS clients to use NFSv4 using the new **nfsconf** tool.

BEFORE YOU BEGIN

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

On `workstation`, run the **lab netstorage-nfs start** command. This command runs a start script that determines if the `servera` and `serverb` machines are reachable on the network. The script will alert you if they are not available. The start script configures `serverb` as an NFSv4 Server, sets up permissions, and exports directories. It creates users and groups needed on both `servera` and `serverb`.

```
[student@workstation ~]$ lab netstorage-nfs start
```

A shipping company uses a central server, `serverb`, to host a number of shared documents and directories. Users on `servera`, who are all members of the `admin` group, need access to the persistently mounted NFS share.

Important information:

- `serverb` shares the **/shares/public** directory, which contains some text files.
- Members of the `admin` group (`admin1`, `sysmanager1`) have read and write access to the **/shares/public** shared directory.
- The principle mount point for `servera` is **/public**.
- All user passwords are set to `redhat`.

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

   1.1. Log in to `servera` as the `student` user.

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

1.2. Use the **sudo -i** command to switch to the `root` user. The password for the `student` user is `student`.

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

2. Use the **nfsconf** tool to configure **/etc/nfs.conf** to enable NFS Clients to work in version 4.X only and to ensure that TCP mode is enabled and UDP mode is disabled.

2.1. Use the **nfsconf** tool to disable the keys udp, `vers2`, `vers3`.

```
[root@servera ~]# nfsconf --set nfsd udp n
[root@servera ~]# nfsconf --set nfsd vers2 n
[root@servera ~]# nfsconf --set nfsd vers3 n
```

2.2. Use the **nfsconf** tool to enable the keys `tcp`, `vers4`, `vers4.0`, `vers4.1`, `vers4.2`.

```
[root@servera ~]# nfsconf --set nfsd tcp y
[root@servera ~]# nfsconf --set nfsd vers4 y
[root@servera ~]# nfsconf --set nfsd vers4.0 y
[root@servera ~]# nfsconf --set nfsd vers4.1 y
[root@servera ~]# nfsconf --set nfsd vers4.2 y
```

3. Test the NFS server on `serverb` using `servera` as the NFS client.

3.1. Create the mount point **/public** on `servera`.

```
[root@servera ~]# mkdir /public
```

3.2. On `servera`, use the **mount** command to verify that the **/share/public** NFS share exported by `serverb` mounts correctly on the **/public** mount point.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares/
public /public
```

3.3. List the contents of the mounted NFS share.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r--. 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root admin 27 Apr  8 22:36 Trackings.txt
```

3.4. Explore the **mount** options for the NFS mounted share.

```
[root@servera ~]# mount | grep public
serverb.lab.example.com:/shares/public on /public type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
```

3.5. Unmount the NFS share.

```
[root@servera ~]# umount /public
```

4. Configure `servera` to ensure that the share used above is persistently mounted.

4.1. Open the **`/etc/fstab`** file for editing.

```
[root@servera ~]# vim /etc/fstab
```

Add the following line to the end of the file:

```
serverb.lab.example.com:/shares/public  /public  nfs  rw,sync  0 0
```

4.2. Use the **`mount`** command to mount the shared directory.

```
[root@servera ~]# mount /public
```

4.3. List the contents of the shared directory.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r--. 1 root   admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root   admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root   admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root   admin 27 Apr  8 22:36 Trackings.txt
```

4.4. Reboot the `servera` machine.

```
[root@servera ~]# systemctl reboot
```

5. After `servera` has finished rebooting, log in to `servera` as the `admin1` user and test the persistently mounted NFS share.

5.1. Log in to `servera` as the `admin1` user.

```
[student@workstation ~]$ ssh admin1@servera
[admin1@servera ~]$
```

5.2. Test the NFS share mounted on **`/public`**

```
[admin1@servera ~]$ ls -l /public
total 16
-rw-r--r--. 1 root   admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root   admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root   admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root   admin 27 Apr  8 22:36 Trackings.txt
[admin1@servera ~]$ cat /public/NOTES.txt
```

```
###In this file you can log all your notes###
[admin1@servera ~]$ echo "This is a test" > /public/Test.txt
[admin1@servera ~]$ cat /public/Test.txt
This is a test
```

5.3. Log off from `servera`.

```
[admin1@servera ~]$ exit
logout
Connection to servera closed.
```

Finish

On `workstation`, run the **`lab netstorage-nfs finish`** script to complete this exercise.

```
[student@workstation ~]$ lab netstorage-nfs finish
```

This concludes the guided exercise.

AUTOMOUNTING NETWORK-ATTACHED STORAGE

OBJECTIVES

After completing this section, you should be able to:

- Describe the benefits of using the automounter.
- Automount NFS shares using direct and indirect maps, including wildcards.

MOUNTING NFS SHARES WITH THE AUTOMOUNTER

The *automounter* is a service (`autofs`) that automatically mounts NFS shares "on-demand," and will automatically unmount NFS shares when they are no longer being used.

Automounter Benefits

- Users do not need to have root privileges to run the **`mount`** and **`umount`** commands.
- NFS shares configured in the automounter are available to all users on the machine, subject to access permissions.
- NFS shares are not permanently connected like entries in **`/etc/fstab`**, freeing network and system resources.
- The automounter is configured on the client side; no server-side configuration is required.
- The automounter uses the same options as the **`mount`** command, including security options.
- The automounter supports both direct and indirect mount-point mapping, for flexibility in mount-point locations.
- `autofs` creates and removes indirect mount points, eliminating manual management.
- NFS is the default automounter network file system, but other network file systems can be automatically mounted.
- `autofs` is a service that is managed like other system services.

Create an automount

Configuring an automount is a multiple step process:

1. Install the *autofs* package.

```
[user@host ~]$ sudo yum install autofs
```

   This package contains everything needed to use the automounter for NFS shares.

2. Add a *master map* file to **`/etc/auto.master.d`**. This file identifies the base directory used for mount points and identifies the mapping file used for creating the automounts.

```
[user@host ~]$ sudo vim /etc/auto.master.d/demo.autofs
```

The name of the master map file is arbitrary (although typically meaningful), but it must have an extension of **.autofs** for the subsystem to recognize it. You can place multiple entries in a single master map file; alternatively, you can create multiple master map files each with its own entries grouped logically.

Add the master map entry, in this case, for indirectly mapped mounts:

```
/shares  /etc/auto.demo
```

This entry uses the **/shares** directory as the base for indirect automounts. The **/etc/auto.demo** file contains the mount details. Use an absolute file name. The **auto.demo** file needs to be created before starting the `autofs` service.

3. Create the mapping files. Each mapping file identifies the mount point, mount options, and source location to mount for a set of automounts.

```
[user@host ~]$ sudo vim /etc/auto.demo
```

The mapping file-naming convention is **/etc/auto.*name***, where *name* reflects the content of the map.

```
work  -rw,sync  serverb:/shares/work
```

The format of an entry is *mount point*, *mount options*, and *source location*. This example shows a basic indirect mapping entry. Direct maps and indirect maps using wildcards are covered later in this section.

- Known as the *key* in the man pages, the *mount point* is created and removed automatically by the `autofs` service. In this case, the fully qualified mount point is **/shares/work** (see the master map file). The **/shares** directory and the **/shares/work** directories are created and removed as needed by the `autofs` service.

  In this example, the local mount point mirrors the server's directory structure, however this is not required; the local mount point can be named anything. The `autofs` service does not enforce a specific naming structure on the client.

- Mount options start with a dash character (-) and are comma-separated with no white space. Mount options available to a manual mounting of a file system are available when automounting. In this example, the automounter mounts the share with read/write access (**rw** option), and the server is synchronized immediately during write operations (**sync** option).

  Useful automounter-specific options include **-fstype=** and **-strict**. Use **fstype** to specify the file system type, for example, **nfs4** or **xfs**, and use **strict** to treat errors when mounting file systems as fatal.

- The source location for NFS shares follows the `host:/pathname` pattern; in this example, `serverb:/shares/work`. For this automount to succeed, the NFS server, `serverb`, must *export* the directory with read/write access and the user requesting access must have standard Linux file permissions on the directory. If `serverb` exports the directory with read/only access, then the client will get read/only access even though it requested read/write access.

4. Start and enable the automounter service.

Use **systemctl** to start and enable the autofs service.

```
[user@host ~]$ sudo systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /
usr/lib/systemd/system/autofs.service.
```

Direct Maps

Direct maps are used to map an NFS share to an existing absolute path mount point.

To use directly mapped mount points, the master map file might appear as follows:

```
/-  /etc/auto.direct
```

All direct map entries use **/-** as the base directory. In this case, the mapping file that contains the mount details is **/etc/auto.direct**.

The content for the **/etc/auto.direct** file might appear as follows:

```
/mnt/docs  -rw,sync  serverb:/shares/docs
```

The mount point (or key) is always an absolute path. The rest of the mapping file uses the same structure.

In this example the **/mnt** directory exists, and it is not managed by autofs. The full directory **/mnt/docs** will be created and removed automatically by the autofs service.

Indirect Wildcard Maps

When an NFS server exports multiple subdirectories within a directory, then the automounter can be configured to access any one of those subdirectories using a single mapping entry.

Continuing the previous example, if serverb:/shares exports two or more subdirectories and they are accessible using the same mount options, then the content for the **/etc/auto.demo** file might appear as follows:

```
*  -rw,sync  serverb:/shares/&
```

The mount point (or key) is an asterisk character (*), and the subdirectory on the source location is an ampersand character (&). Everything else in the entry is the same.

When a user attempts to access **/shares/work**, the key ***** (which is **work** in this example) replaces the ampersand in the source location and serverb:/shares/work is mounted. As with the indirect example, the **work** directory is created and removed automatically by autofs.

REFERENCES

autofs(5), **automount**(8), **auto.master**(5), and **mount.nfs**(8) man pages

GUIDED EXERCISE

AUTOMOUNTING NETWORK-ATTACHED STORAGE

PERFORMANCE CHECKLIST

In this exercise, you will create direct mapped and indirect mapped automount-managed mount points that mount NFS file systems.

OUTCOMES

You should be able to:

- Install required packages needed for the automounter.
- Configure direct and indirect automounter maps, getting resources from a preconfigured NFSv4 server.
- Understand the difference between direct and indirect automounter maps.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab netstorage-autofs start`** command. This start script determines if `servera` and `serverb` are reachable on the network. The script will alert you if they are not available. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. It also creates users and groups needed on both `servera` and `serverb`.

```
[student@workstation ~]$ lab netstorage-autofs start
```

An internet service provider uses a central server, `serverb`, to host shared directories containing important documents that need to be available on demand. When users log in to `servera` they need access to the automounted shared directories.

Important information:

- `serverb` is exporting as an NFS share the **`/shares/indirect`** directory, which in turn contains the **`west`**, **`central`**, and **`east`** subdirectories.
- `serverb` is also exporting as an NFS share the **`/shares/direct/external`** directory.
- The `operators` group consists of the `operator1` and `operator2` users. They have read and write access to the shared directories **`/shares/indirect/west`**, **`/shares/indirect/central`**, and **`/shares/indirect/east`**.
- The `contractors` group consists of the `contractor1` and `contractor2` users. They have read and write access to the **`/shares/direct/external`** shared directory.
- The expected mount points for `servera` are **`/external`** and **`/internal`**.

- The **/shares/direct/external** shared directory should be automounted on servera using a *direct* map on **/external**.
- The **/shares/indirect/west** shared directory should be automounted on servera using an *indirect* map on **/internal/west**.
- The **/shares/indirect/central** shared directory should be automounted on servera using an *indirect* map on **/internal/central**.
- The **/shares/indirect/east** shared directory should be automounted on servera using an *indirect* map on **/internal/east**.
- All user passwords are set to redhat.

1. Log in to servera and install the required packages.

   1.1. Log in to servera as the student user.

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

   1.2. Use the **sudo -i** command to switch to the root user. The password for the student user is student.

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

   1.3. Install the *autofs* package.

```
[root@servera ~]# yum install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2. Configure an automounter direct map on servera using shares from serverb. Create the direct map using files named **/etc/auto.master.d/direct.autofs** for the master map and **/etc/auto.direct** for the mapping file. Use the **/external** directory as the main mount point on servera.

   2.1. Test the NFS server and share before proceeding to configure the automounter.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/direct/external /mnt
[root@servera ~]# ls -l /mnt
total 4
-rw-r--r--. 1 root contractors 22 Apr  7 23:15 README.txt
[root@servera ~]# umount /mnt
```

   2.2. Create a master map file named **/etc/auto.master.d/direct.autofs**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.master.d/direct.autofs
/- /etc/auto.direct
```

2.3. Create a direct map file named **/etc/auto.direct**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.direct
/external -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/direct/
external
```

3. Configure an automounter indirect map on servera using shares from serverb. Create the indirect map using files named **/etc/auto.master.d/indirect.autofs** for the master map and **/etc/auto.indirect** for the mapping file. Use the **/internal** directory as the main mount point on servera.

3.1. Test the NFS server and share before proceeding to configure the automounter.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares/
indirect /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 east
drwxrws---. 2 root operators 24 Apr  7 23:34 west
[root@servera ~]# umount /mnt
```

3.2. Create a master map file named **/etc/auto.master.d/indirect.autofs**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.master.d/indirect.autofs
/internal /etc/auto.indirect
```

3.3. Create an indirect map file named **/etc/auto.indirect**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.indirect
* -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/indirect/&
```

4. Start the `autofs` service on `servera` and enable it to start automatically at boot time. Reboot `servera` to determine if the `autofs` service starts automatically.

4.1. Start and enable the `autofs` service on `servera`.

```
[root@servera ~]# systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/
autofs.service → /usr/lib/systemd/system/autofs.service.
```

4.2. Reboot the `servera` machine.

```
[root@servera ~]# systemctl reboot
```

5. Test the direct automounter map as the `contractor1` user. When done, exit from the `contractor1` user session on `servera`.

5.1. After the `servera` machine has finished booting, log in to `servera` as the `student` user.

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

5.2. Switch to the `contractor1` user.

```
[student@servera ~]$ su - contractor1
Password: redhat
```

5.3. List the **`/external`** mount point.

```
[contractor1@servera ~]$ ls -l /external
total 4
-rw-r--r--. 1 root contractors 22 Apr  7 23:34 README.txt
```

5.4. Review the content and test the access to the **`/external`** mount point.

```
[contractor1@servera ~]$ cat /external/README.txt
###External Folder###
[contractor1@servera ~]$ echo testing-direct > /external/testing.txt
[contractor1@servera ~]$ cat /external/testing.txt
testing-direct
```

5.5. Exit from the `contractor1` user session.

```
[contractor1@servera ~]$ exit
logout
```

6. Test the indirect automounter map as the `operator1` user. When done, log off from `servera`.

6.1. Switch to `operator1` user.

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

6.2. List the **`/internal`** mount point.

```
[operator1@servera ~]$ ls -l /internal
total 0
```

NOTE

You will notice that in an automounter indirect map, even if you are in the mapped mount point, you need to call each of the shared subdirectories or files on demand to get access to them. In an automounter direct map, after you open the mapped mount point, you get access to the directories and content configured in the shared directory.

6.3. Test the **`/internal/west`** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /internal/west/
total 4
-rw-r--r--. 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/west/README.txt
###West Folder###
[operator1@servera ~]$ echo testing-1 > /internal/west/testing-1.txt
[operator1@servera ~]$ cat /internal/west/testing-1.txt
testing-1
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

6.4. Test the **`/internal/central`** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /internal/central
total 4
-rw-r--r--. 1 root operators 21 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/central/README.txt
###Central Folder###
[operator1@servera ~]$ echo testing-2 > /internal/central/
testing-2.txt
[operator1@servera ~]$ cat /internal/central/testing-2.txt
testing-2
```

```
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

6.5. Test the **/internal/east** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /internal/east
total 4
-rw-r--r--. 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/east/README.txt
###East Folder###
[operator1@servera ~]$ echo testing-3 > /internal/east/testing-3.txt
[operator1@servera ~]$ cat /internal/east/testing-3.txt
testing-3
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 east
drwxrws---. 2 root operators 24 Apr  7 23:34 west
```

6.6. Test the **/external** automounter shared directory access.

```
[operator1@servera ~]$ ls -l /external
ls: cannot open directory '/external': Permission denied
```

6.7. Log off from servera.

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

Finish

On workstation, run the **lab netstorage-autofs finish** script to complete this exercise.

```
[student@workstation ~]$ lab netstorage-autofs finish
```

This concludes the guided exercise.

▷ LAB

ACCESSING NETWORK-ATTACHED STORAGE

PERFORMANCE CHECKLIST

In this lab, you will set up the automounter with an indirect map, using shares from an NFSv4 server.

OUTCOMES

You should be able to:

- Install required packages needed to set up the automounter.
- Configure an automounter indirect map, getting resources from a preconfigured NFSv4 server.
- Configure an NFS client to use NFSv4 using the **`nfsconf`** tool.

BEFORE YOU BEGIN

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

On `workstation`, run the **`lab netstorage-review start`** command. This start script determines if the `servera` and `serverb` systems are reachable on the network. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. It also creates users and groups needed on both `servera` and `serverb` systems.

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

An IT support company uses a central server, `serverb`, to host some shared directories on **`/remote/shares`** for their groups and users. Users need to be able to log in and have their shared directories mounted on demand and ready to use, under the **`/shares`** directory on `servera`.

Important information:

- `serverb` is sharing the **`/shares`** directory, which in turn contains the **`management`**, **`production`** and **`operation`** subdirectories.
- The `managers` group consists of the `manager1` and `manager2` users. They have read and write access to the **`/shares/management`** shared directory.
- The `production` group consists of the `dbuser1` and `sysadmin1` users. They have read and write access to the **`/shares/production`** shared directory.
- The `operators` group consists of the `contractor1` and `consultant1` users. They have read and write access to the **`/shares/operation`** shared directory.
- The main mount point for `servera` is the **`/remote`** directory.

- The **/shares/management** shared directory should be automounted on **/remote/management** on servera.
- The **/shares/production** shared directory should be automounted on **/remote/production** on servera.
- The **/shares/operation** shared directory should be automounted on **/remote/operation** on servera.
- All user passwords are set to redhat.

1. Log in to servera and install the required packages.
2. Use the **nfsconf** command to configure **/etc/nfs.conf**. Enable the NFS client to work only in version 4.X and ensure that TCP mode is enabled and UDP mode is disabled.
3. Configure an automounter indirect map on servera using shares from serverb. Create an indirect map using files named **/etc/auto.master.d/shares.autofs** for the master map and **/etc/auto.shares** for the mapping file. Use the **/remote** directory as the main mount point on servera. Reboot servera to determine if the autofs service starts automatically.
4. Test the autofs configuration with the various users. When done, log off from servera.

Evaluation

On workstation, run the **lab netstorage-review grade** command to confirm success of this exercise.

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

Finish

On workstation, run the **lab netstorage-review finish** command to complete this exercise.

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

This concludes the lab.

▷ SOLUTION

ACCESSING NETWORK-ATTACHED STORAGE

PERFORMANCE CHECKLIST

In this lab, you will set up the automounter with an indirect map, using shares from an NFSv4 server.

OUTCOMES

You should be able to:

- Install required packages needed to set up the automounter.
- Configure an automounter indirect map, getting resources from a preconfigured NFSv4 server.
- Configure an NFS client to use NFSv4 using the **nfsconf** tool.

BEFORE YOU BEGIN

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

On `workstation`, run the **lab netstorage-review start** command. This start script determines if the `servera` and `serverb` systems are reachable on the network. The start script configures `serverb` as an NFSv4 server, sets up permissions, and exports directories. It also creates users and groups needed on both `servera` and `serverb` systems.

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

An IT support company uses a central server, `serverb`, to host some shared directories on **/remote/shares** for their groups and users. Users need to be able to log in and have their shared directories mounted on demand and ready to use, under the **/shares** directory on `servera`.

Important information:

- `serverb` is sharing the **/shares** directory, which in turn contains the **management**, **production** and **operation** subdirectories.
- The `managers` group consists of the `manager1` and `manager2` users. They have read and write access to the **/shares/management** shared directory.
- The `production` group consists of the `dbuser1` and `sysadmin1` users. They have read and write access to the **/shares/production** shared directory.
- The `operators` group consists of the `contractor1` and `consultant1` users. They have read and write access to the **/shares/operation** shared directory.
- The main mount point for `servera` is the **/remote** directory.

- The **/shares/management** shared directory should be automounted on **/remote/management** on servera.
- The **/shares/production** shared directory should be automounted on **/remote/production** on servera.
- The **/shares/operation** shared directory should be automounted on **/remote/operation** on servera.
- All user passwords are set to redhat.

1. Log in to servera and install the required packages.

   1.1. Log in to servera as the student user.

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

   1.2. Use the **sudo -i** command to switch to the root user. The password for the student user is student.

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

   1.3. Install the *autofs* package.

```
[root@servera ~]# yum install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

2. Use the **nfsconf** command to configure **/etc/nfs.conf**. Enable the NFS client to work only in version 4.X and ensure that TCP mode is enabled and UDP mode is disabled.

   2.1. Use the **nfsconf** tool to disable the keys udp, vers2, vers3.

```
[root@servera ~]# nfsconf --set nfsd udp n
[root@servera ~]# nfsconf --set nfsd vers2 n
[root@servera ~]# nfsconf --set nfsd vers3 n
```

   2.2. Use the **nfsconf** tool to enable the keys tcp, vers4, vers4.0, vers4.1, vers4.2.

```
[root@servera ~]# nfsconf --set nfsd tcp y
[root@servera ~]# nfsconf --set nfsd vers4 y
[root@servera ~]# nfsconf --set nfsd vers4.0 y
[root@servera ~]# nfsconf --set nfsd vers4.1 y
[root@servera ~]# nfsconf --set nfsd vers4.2 y
```

3. Configure an automounter indirect map on servera using shares from serverb. Create an indirect map using files named **/etc/auto.master.d/shares.autofs** for the master map and **/etc/auto.shares** for the mapping file. Use the **/remote** directory as the

main mount point on servera. Reboot servera to determine if the autofs service starts automatically.

3.1. Test the NFS server before proceeding to configure the automounter.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrwx---. 2 root managers   25 Apr  4 01:13 management
drwxrwx---. 2 root operators  25 Apr  4 01:13 operation
drwxrwx---. 2 root production 25 Apr  4 01:13 production
[root@servera ~]# umount /mnt
```

3.2. Create a master map file named **/etc/auto.master.d/shares.autofs**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.master.d/shares.autofs
/remote /etc/auto.shares
```

3.3. Create an indirect map file named **/etc/auto.shares**, insert the following content, and save the changes.

```
[root@servera ~]# vim /etc/auto.shares
* -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/&
```

3.4. Start and enable the autofs service on servera.

```
[root@servera ~]# systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/
autofs.service → /usr/lib/systemd/system/autofs.service.
```

3.5. Reboot theservera machine.

```
[root@servera ~]# systemctl reboot
```

4. Test the autofs configuration with the various users. When done, log off from servera.

4.1. After the servera machine has finished booting, log in to servera as the student user.

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

4.2. Use the **su - manager1** command to switch to the manager1 user and test access.

```
[student@servera ~]$ su - manager1
Password: redhat
[manager1@servera ~]$ ls -l /remote/management/
total 4
-rw-r--r--. 1 root managers 46 Apr  4 01:13 Welcome.txt
```

```
[manager1@servera ~]$ cat /remote/management/Welcome.txt
###Welcome to Management Folder on SERVERB###
[manager1@servera ~]$ echo TEST1 > /remote/management/Test.txt
[manager1@servera ~]$ cat /remote/management/Test.txt
TEST1
[manager1@servera ~]$ ls -l /remote/operation/
ls: cannot open directory '/remote/operation/': Permission denied
[manager1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[manager1@servera ~]$ exit
logout
```

4.3. Switch to the `dbuser1` user and test access.

```
[student@servera ~]$ su - dbuser1
Password: redhat
[dbuser1@servera ~]$ ls -l /remote/production/
total 4
-rw-r--r--. 1 root production 46 Apr  4 01:13 Welcome.txt
[dbuser1@servera ~]$ cat /remote/production/Welcome.txt
###Welcome to Production Folder on SERVERB###
[dbuser1@servera ~]$ echo TEST2 > /remote/production/Test.txt
[dbuser1@servera ~]$ cat /remote/production/Test.txt
TEST2
[dbuser1@servera ~]$ ls -l /remote/operation/
ls: cannot open directory '/remote/operation/': Permission denied
[dbuser1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
[dbuser1@servera ~]$ exit
logout
```

4.4. Switch to the `contractor1` user and test access.

```
[student@servera ~]$ su - contractor1
Password: redhat
[contractor1@servera ~]$ ls -l /remote/operation/
total 4
-rw-r--r--. 1 root operators 45 Apr  4 01:13 Welcome.txt
[contractor1@servera ~]$ cat /remote/operation/Welcome.txt
###Welcome to Operation Folder on SERVERB###
[contractor1@servera ~]$ echo TEST3 > /remote/operation/Test.txt
[contractor1@servera ~]$ cat /remote/operation/Test.txt
TEST3
[contractor1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
```

```
[contractor1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[contractor1@servera ~]$ exit
logout
```

4.5. Explore the **mount** options for the NFS automounted share.

```
[student@servera ~]$ mount | grep nfs
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
serverb.lab.example.com:/shares/management on /remote/management
 type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/operation on /remote/operation
 type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/production on /remote/production
 type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,sync,proto=tcp,timeo=600,
retrans=2,sec=sys,clientaddr=172.25.250.10,local_lock=none,addr=172.25.250.11)
```

4.6. Log off from servera.

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

Evaluation

On workstation, run the **lab netstorage-review grade** command to confirm success of this exercise.

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

Finish

On workstation, run the **lab netstorage-review finish** command to complete this exercise.

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

This concludes the lab.

SUMMARY

In this chapter, you learned how to:

- Mount and unmount an NFS export from the command line.
- Configure an NFS export to automatically mount at startup.
- Configure the automounter with direct and indirect maps, and describe their differences.
- Configure NFS clients to use NFSv4 using the new **nfsconf** tool.

CHAPTER 15

MANAGING NETWORK SECURITY

GOAL	Control network connections to services using the system firewall.
OBJECTIVES	• Accept or reject network connections to system services using firewalld rules.
SECTIONS	• Managing Server Firewalls (and Guided Exercise)
LAB	Managing Network Security

MANAGING SERVER FIREWALLS

OBJECTIVES

After completing this section, you should be able to accept or reject network connections to system services using firewalld rules.

FIREWALL ARCHITECTURE CONCEPTS

The Linux kernel includes `netfilter`, a framework for network traffic operations such as packet filtering, network address translation and port translation. By implementing handlers in the kernel that intercept function calls and messages, `netfilter` allows other kernel modules to interface directly with the kernel's networking stack. Firewall software uses these hooks to register filter rules and packet-modifying functions, allowing every packet going through the network stack to be processed. Any incoming, outgoing, or forwarded network packet can be inspected, modified, dropped, or routed programmatically before reaching user space components or applications. `Netfilter` is the primary component in Red Hat Enterprise Linux 8 firewalls.

Nftables enhances netfilter

The Linux kernel also includes `nftables`, a new filter and packet classification subsystem that has enhanced portions of `netfilter`'s code, but retaining the `netfilter` architecture such as networking stack hooks, connection tracking system, and the logging facility. The advantages of the `nftables` update is faster packet processing, faster ruleset updates, and simultaneous IPv4 and IPv6 processing from the same rules. Another major difference between `nftables` and the original `netfilter` are their interfaces. `Netfilter` is configured through multiple utility frameworks, including `iptables`, `ip6tables`, `arptables`, and `ebtables`, which are now deprecated. Nftables uses the single **`nft`** user-space utility, allowing all protocol management to occur through a single interface, eliminating historical contention caused by diverse front ends and multiple `netfilter` interfaces.

Introducing firewalld

`Firewalld` is a dynamic firewall manager, a front end to the `nftables` framework using the **`nft`** command. Until the introduction of `nftables`, `firewalld` used the **`iptables`** command to configure `netfilter` directly, as an improved alternative to the `iptables` service. In RHEL 8, `firewalld` remains the recommended front end, managing firewall rulesets using **`nft`**. `Firewalld` remains capable of reading and managing `iptables` configuration files and rulesets, using `xtables-nft-multi` to translate `iptables` objects directly into `nftables` rules and objects. Although strongly discouraged, `firewalld` can be configured to revert to the `iptables` back-end for complex use cases where existing `iptables` rulesets cannot be properly processed by **`nft`** translations.

Applications query the subsystem using the `D-Bus` interface. The `firewalld` subsystem, available from the *firewalld* RPM package, is not included in a minimal install, but is included in a base installation. With `firewalld`, firewall management is simplified by classifying all network traffic into *zones*. Based on criteria such as the source IP address of a packet or the incoming network interface, traffic is diverted into the firewall rules for the appropriate zone. Each zone has its own list of ports and services that are either open or closed.

NOTE

For laptops or other machines that regularly change networks, ***NetworkManager*** can be used to automatically set the firewall zone for a connection. The zones are customized with rules appropriate for particular connections.

This is especially useful when traveling between home, work, and public wireless networks. A user might want their system's `sshd` service to be reachable when connected to their home and corporate networks, but not when connected to the public wireless network in the local coffee shop.

Firewalld checks the source address for every packet coming into the system. If that source address is assigned to a specific zone, the rules for that zone apply. If the source address is not assigned to a zone, **`firewalld`** associates the packet with the zone for the incoming network interface and the rules for that zone apply. If the network interface is not associated with a zone for some reason, then **`firewalld`** associates the packet with the default zone.

The default zone is not a separate zone, but is a designation for an existing zone. Initially, **`firewalld`** designates the `public` zone as default, and maps the `lo` loopback interface to the `trusted` zone.

Most zones allow traffic through the firewall, which matches a list of particular ports and protocols, such as **`631/udp`**, or pre-defined services, such as `ssh`. If the traffic does not match a permitted port and protocol or service, it is generally rejected. (The `trusted` zone, which permits all traffic by default, is one exception to this.)

Pre-defined Zones

Firewalld has pre-defined zones, each of which you can customize. By default, all zones permit any incoming traffic which is part of a communication initiated by the system, and all outgoing traffic. The following table details these initial zone configuration.

Default Configuration of Firewalld Zones

ZONE NAME	DEFAULT CONFIGURATION
`trusted`	Allow all incoming traffic.
`home`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh`, `mdns`, `ipp-client`, `samba-client`, or `dhcpv6-client` pre-defined services.
`internal`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh`, `mdns`, `ipp-client`, `samba-client`, or `dhcpv6-client` pre-defined services (same as the `home` zone to start with).
`work`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh`, `ipp-client`, or `dhcpv6-client` pre-defined services.
`public`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` or `dhcpv6-client` pre-defined services. *The default zone for newly added network interfaces.*

ZONE NAME	DEFAULT CONFIGURATION
`external`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` pre-defined service. Outgoing IPv4 traffic forwarded through this zone is *masqueraded* to look like it originated from the IPv4 address of the outgoing network interface.
`dmz`	Reject incoming traffic unless related to outgoing traffic or matching the `ssh` pre-defined service.
`block`	Reject all incoming traffic unless related to outgoing traffic.
`drop`	Drop all incoming traffic unless related to outgoing traffic (do not even respond with ICMP errors).

For a list of available pre-defined zones and intended use, see **`firewalld.zones`**(5).

Pre-defined Services

Firewalld has a number of pre-defined services. These service definitions help you identify particular network services to configure. Instead of having to research relevant ports for the `samba-client` service, for example, specify the pre-built `samba-client` service to configure the correct ports and protocols. The following table lists the pre-defined services used in the initial firewall zones configuration.

Selected Pre-defined Firewalld Services

SERVICE NAME	CONFIGURATION
`ssh`	Local SSH server. Traffic to 22/tcp
`dhcpv6-client`	Local DHCPv6 client. Traffic to 546/udp on the fe80::/64 IPv6 network
`ipp-client`	Local IPP printing. Traffic to 631/udp.
`samba-client`	Local Windows file and print sharing client. Traffic to 137/udp and 138/udp.
`mdns`	Multicast DNS (mDNS) local-link name resolution. Traffic to 5353/udp to the 224.0.0.251 (IPv4) or ff02::fb (IPv6) multicast addresses.

NOTE

Many pre-defined services are included in the *firewalld* package. Use **`firewall-cmd --get-services`** to list them. Configuration files for pre-defined services are found in **`/usr/lib/firewalld/services`**, in a format defined by **`firewalld.zone`**(5).

Either use the pre-defined services or directly specify the port and protocol required. The Web Console graphical interface is used to review pre-defined services and to define additional services.

CONFIGURING THE FIREWALL

System administrators interact with `firewalld` in three ways:

- Directly edit configuration files in **`/etc/firewalld/`** (not discussed in this chapter)
- The Web Console graphical interface
- The **`firewall-cmd`** command-line tool

Configuring Firewall Services Using the Web Console

To configure firewall services with Web Console, log in with privileged access by clicking the ***Reuse my password for privileged tasks*** option. This permits the user to execute commands with sudo privileges to modify firewall services.

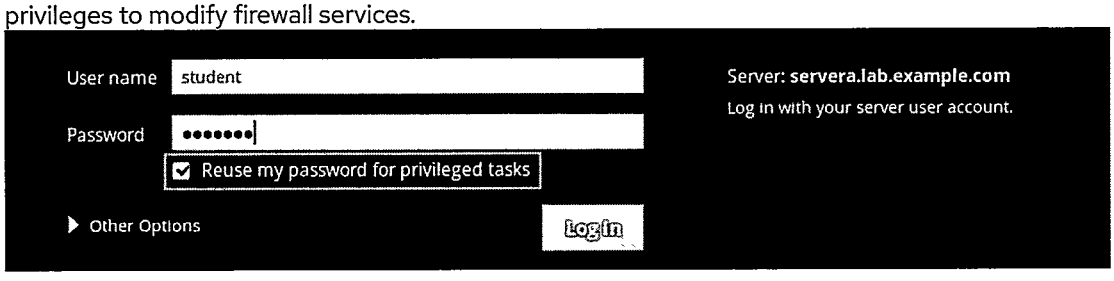


Figure 15.1: The Web Console privileged login

Click the ***Networking*** option in the left navigation menu to display the ***Firewall*** section in the main networking page. Click the ***Firewall*** link to access the allowed services list.

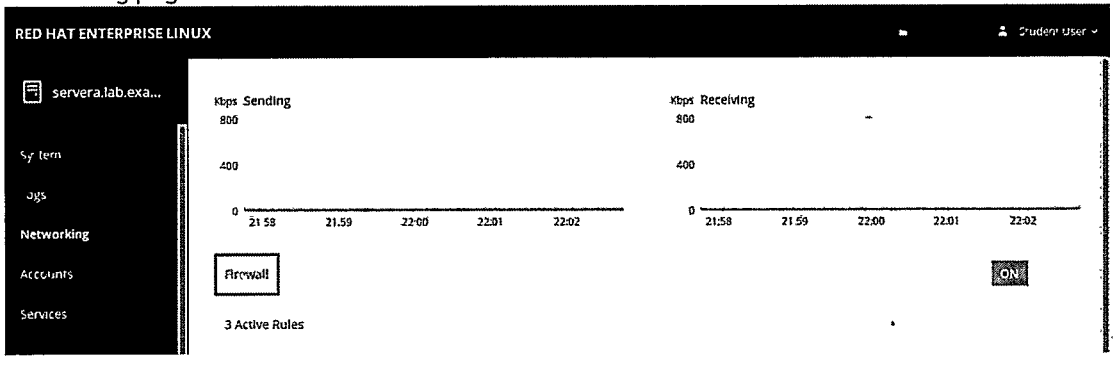


Figure 15.2: The Web Console networking

The allowed services listed are those that are currently permitted by the firewall. Click the arrow (>) to the left of the service name to view service details. To add a service, click the ***Add Services...*** button in the upper right corner of the ***Firewall Allowed Services*** page.

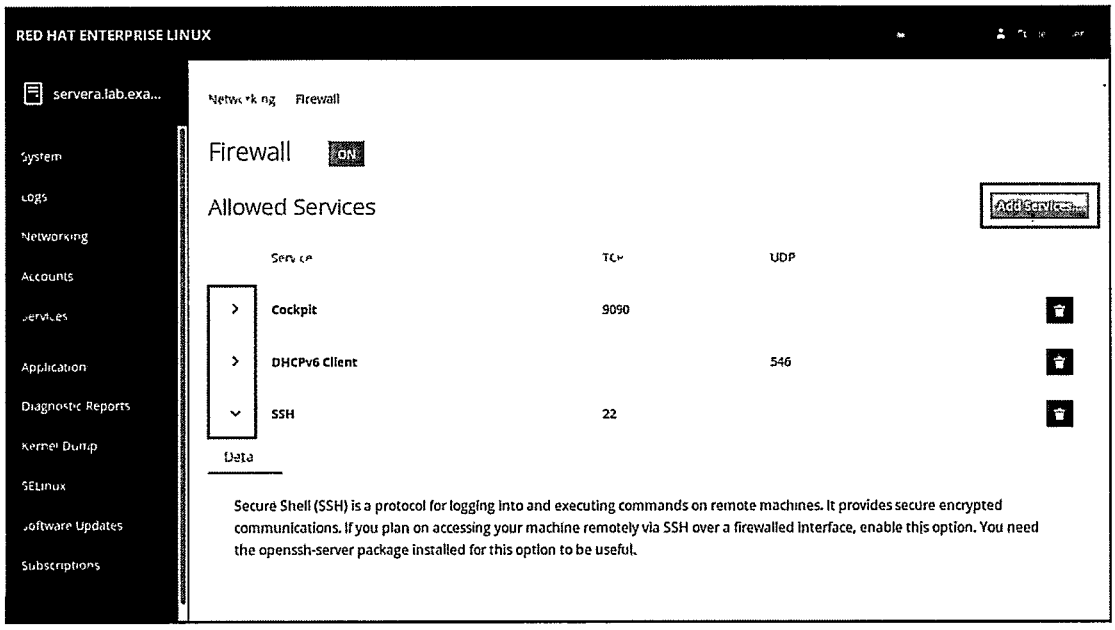


Figure 15.3: The Web Console firewall allowed services list

The ***Add Services*** page displays the available pre-defined services.

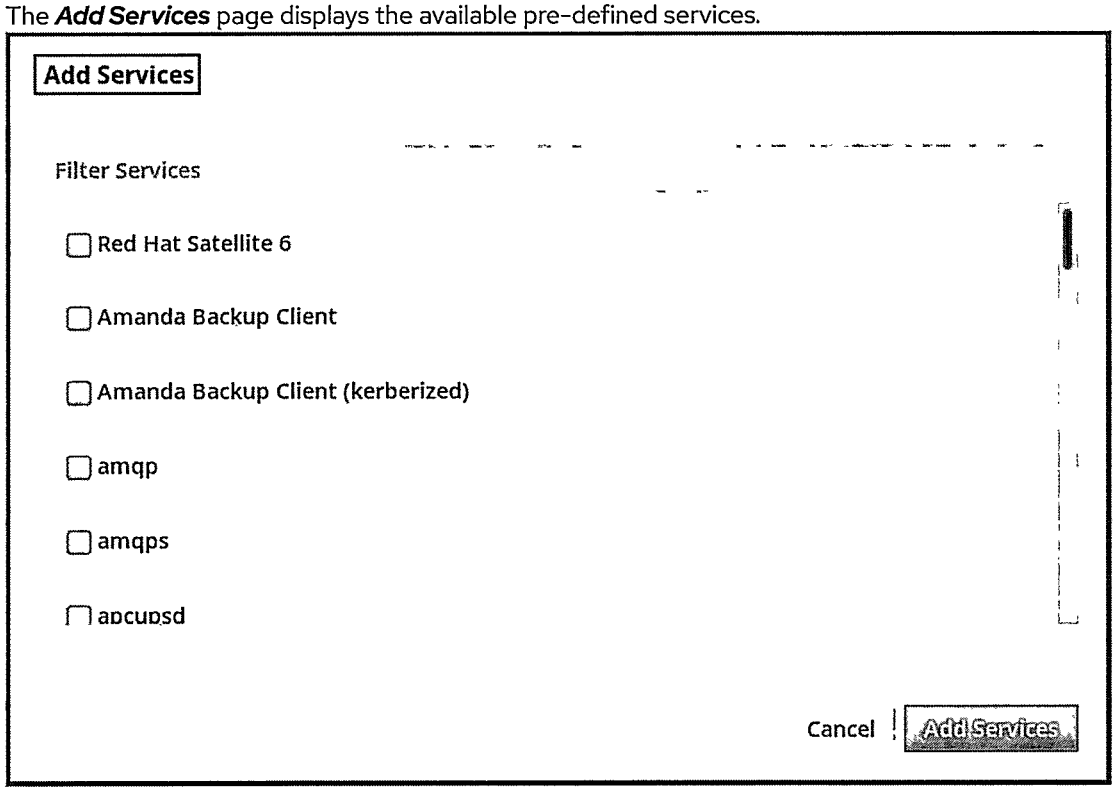


Figure 15.4: The Web Console add services interface

To select a service, scroll through the list or enter a selection in the ***Filter Services*** text box. In the following example, the string **`http`** is entered into the search text box to find services containing that string; that is, web related services. Select the check box to the left of the services to allow through the firewall. Click the ***Add Services*** button to complete the process.

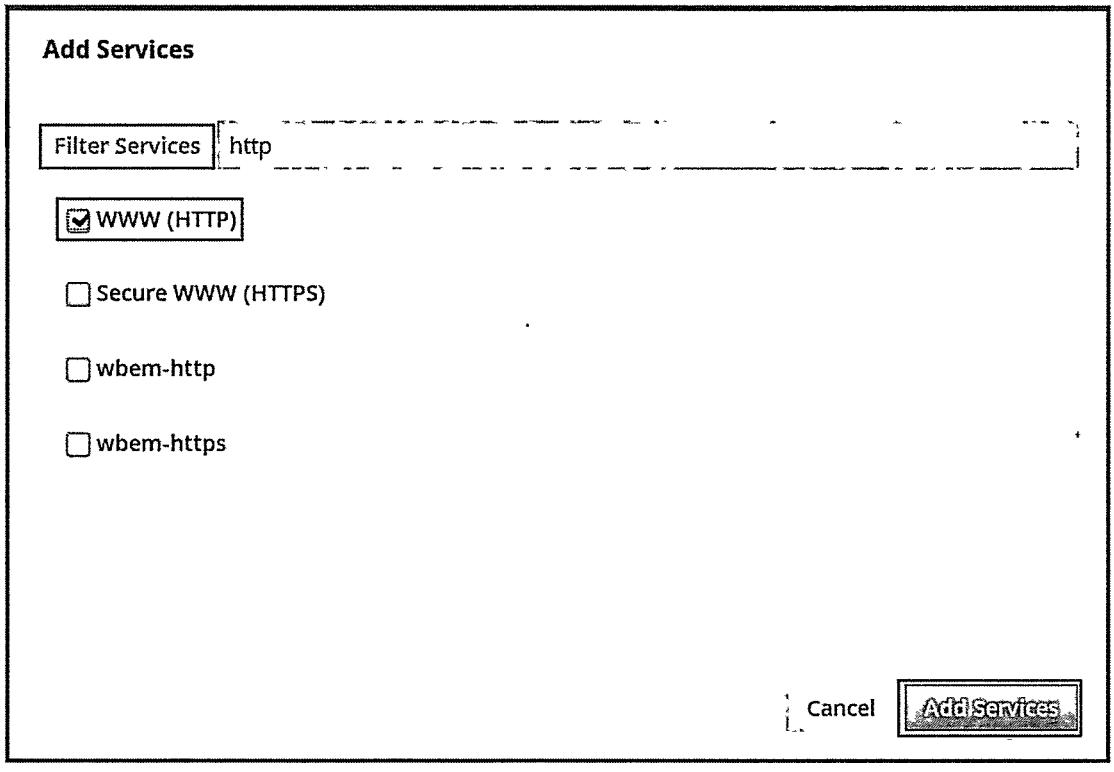


Figure 15.5: The Web Console services filter search

The interface returns to the ***Firewall Allowed Services*** page, where you can review the updated allowed services list.

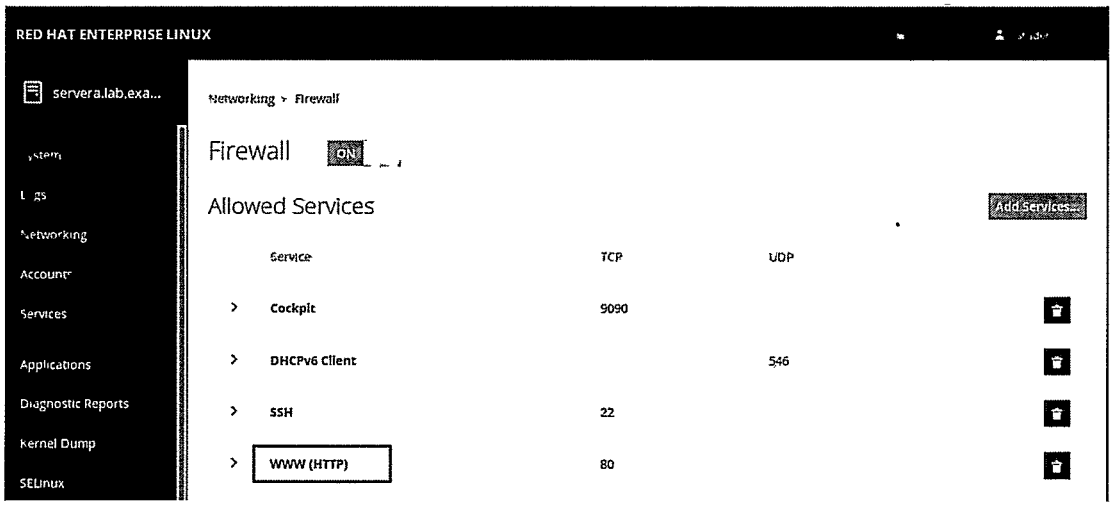


Figure 15.6: The Web Console services list

Configuring the Firewall from the Command Line

The **`firewall-cmd`** command interacts with the **`firewalld`** dynamic firewall manager. It is installed as part of the main *firewalld* package and is available for administrators who prefer to work on the command line, for working on systems without a graphical environment, or for scripting a firewall setup.

The following table lists a number of frequently used **`firewall-cmd`** commands, along with an explanation. Note that unless otherwise specified, almost all commands will work on the *runtime* configuration, unless the **`--permanent`** option is specified. If the **`--permanent`** option

is specified, you must activate the setting by also running the **`firewall-cmd --reload`** command, which reads the current permanent configuration and applies it as the new runtime configuration. Many of the commands listed take the **`--zone=`***`ZONE`* option to determine which zone they affect. Where a netmask is required, use CIDR notation, such as 192.168.1/24.

FIREWALL-CMD COMMANDS	EXPLANATION
`--get-default-zone`	Query the current default zone.
`--set-default-zone=`*`ZONE`*	Set the default zone. This changes both the runtime and the permanent configuration.
`--get-zones`	List all available zones.
`--get-active-zones`	List all zones currently in use (have an interface or source tied to them), along with their interface and source information.
`--add-source=`*`CIDR`* **`[--zone=`***`ZONE`***`]`**	Route all traffic coming from the IP address or network/netmask to the specified zone. If no **`--zone=`** option is provided, the default zone is used.
`--remove-source=`*`CIDR`* **`[--zone=`***`ZONE`***`]`**	Remove the rule routing all traffic from the zone coming from the IP address or network/netmask network. If no **`--zone=`** option is provided, the default zone is used.
`--add-interface=`*`INTERFACE`* **`[--zone=`***`ZONE`***`]`**	Route all traffic coming from *`INTERFACE`* to the specified zone. If no **`--zone=`** option is provided, the default zone is used.
`--change-interface=`*`INTERFACE`* **`[--zone=`***`ZONE`***`]`**	Associate the interface with *ZONE* instead of its current zone. If no **`--zone=`** option is provided, the default zone is used.
`--list-all [--zone=`*`ZONE`***`]`**	List all configured interfaces, sources, services, and ports for *ZONE*. If no **`--zone=`** option is provided, the default zone is used.
`--list-all-zones`	Retrieve all information for all zones (interfaces, sources, ports, services).
`--add-service=`*`SERVICE`* **`[--zone=`***`ZONE`***`]`**	Allow traffic to *SERVICE*. If no **`--zone=`** option is provided, the default zone is used.
`--add-port=`*`PORT/PROTOCOL`* **`[--zone=`***`ZONE`***`]`**	Allow traffic to the *PORT/PROTOCOL* port(s). If no **`--zone=`** option is provided, the default zone is used.
`--remove-service=`*`SERVICE`* **`[--zone=`***`ZONE`***`]`**	Remove *SERVICE* from the allowed list for the zone. If no **`--zone=`** option is provided, the default zone is used.

FIREWALL-CMD COMMANDS	EXPLANATION
--remove-port=*PORT/PROTOCOL* [--zone=*ZONE*]	Remove the *PORT/PROTOCOL* port(s) from the allowed list for the zone. If no **--zone=** option is provided, the default zone is used.
--reload	Drop the runtime configuration and apply the persistent configuration.

The example commands below set the default zone to `dmz`, assign all traffic coming from the `192.168.0.0/24` network to the `internal` zone, and open the network ports for the `mysql` service on the `internal` zone.

```
[root@host ~]# firewall-cmd --set-default-zone=dmz
[root@host ~]# firewall-cmd --permanent --zone=internal \
--add-source=192.168.0.0/24
[root@host ~]# firewall-cmd --permanent --zone=internal --add-service=mysql
[root@host ~]# firewall-cmd --reload
```

NOTE

For situations where the basic syntax of **firewalld** is not enough, you can also add *rich-rules*, a more expressive syntax, to write complex rules. If even the rich-rules syntax is not enough, you can also use *Direct Configuration* rules, raw **nft** syntax mixed in with **firewalld** rules.

These advanced modes are beyond the scope of this chapter.

REFERENCES

firewall-cmd(1), **firewalld**(1), **firewalld.zone**(5), **firewalld.zones**(5), and **nft**(8) man pages

GUIDED EXERCISE

MANAGING SERVER FIREWALLS

In this exercise, you will control access to system services by adjusting system firewall rules with `firewalld`.

OUTCOMES

You should be able to configure firewall rules to control access to services.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab netsecurity-firewalls start`** command. The command runs a start script to determine whether the `servera` host is reachable on the network.

```
[student@workstation ~]$ lab netsecurity-firewalls start
```

1. From `workstation`, use SSH to log in to `servera` as `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

2. On the `servera` system, ensure that both *httpd* and *mod_ssl* packages are installed. These packages provide the *Apache* web server you will protect with a firewall, and the necessary extensions for the web server to serve content over SSL.

```
[student@servera ~]$ sudo yum install httpd mod_ssl
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

3. As the `student` user on `servera`, create the **`/var/www/html/index.html`** file. Add one line of text that reads: **`I am servera.`**

```
[student@servera ~]$ sudo bash -c \
"echo 'I am servera.' > /var/www/html/index.html"
```

4. Start and enable the `httpd` service on your `servera` system.

```
[student@servera ~]$ sudo systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service
 → /usr/lib/systemd/system/httpd.service.
```

5. Exit from servera.

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

6. From workstation, attempt to access your web server on servera using both the cleartext port 80/TCP and the SSL encapsulated port 443/TCP. Both attempts should fail.

   6.1. This command should fail:

```
[student@workstation ~]$ curl -k http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80: No
 route to host
```

   6.2. This command should also fail:

```
[student@workstation ~]$ curl -k https://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 443: No
 route to host
```

7. Log in to servera as the student user.

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

8. On `servera`, make sure that the `nftables` service is masked and the `firewalld` service is enabled and running.

8.1. Determine whether the status of the `nftables` service is `masked`.

```
[student@servera ~]$ sudo systemctl status nftables
[sudo] password for student: student
● nftables.service - Netfilter Tables
   Loaded: loaded (/usr/lib/systemd/system/nftables.service; disabled;
 vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:nft(8)
```

The results show that `nftables` is disabled and inactive but not masked. Run the following command to mask the service.

```
[student@servera ~]$ sudo systemctl mask nftables
Created symlink /etc/systemd/system/nftables.service → /dev/null.
```

8.2. Verify that the status of the `nftables` service is `masked`.

```
[student@servera ~]$ sudo systemctl status nftables
● nftables.service
   Loaded: masked (Reason: Unit nftables.service is masked.)
   Active: inactive (dead)
```

8.3. Verify that the status of the `firewalld` service is enabled and running.

```
[student@servera ~]$ sudo systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled;
 vendor preset: enabled)
   Active: active (running) since Wed 2019-05-22 15:36:02 CDT; 5min
 ago
     Docs: man:firewalld(1)
 Main PID: 703 (firewalld)
    Tasks: 2 (limit: 11405)
   Memory: 29.8M
   CGroup: /system.slice/firewalld.service
```

```
                 └─703 /usr/libexec/platform-python -s /usr/sbin/firewalld
 --nofork --nopid

May 22 15:36:01 jegui.ilt.example.com systemd[1]: Starting firewalld -
 dynamic firewall daemon...
May 22 15:36:02 jegui.ilt.example.com systemd[1]: Started firewalld -
 dynamic firewall daemon.
```

8.4. Exit from `servera`.

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

9. From `workstation`, open Firefox and log in to the Web Console running on `servera` to add the `httpd` service to the `public` network zone.

9.1. Open Firefox and browse to `https://servera.lab.example.com:9090` to access the Web Console. Accept the self-signed certificate used by `servera` by adding an exception.

9.2. Select the check box next to ***Reuse my password for privileged tasks*** to ensure administrative privileges.

Log in as `student` user with `student` as the password.

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

9.4. Click the ***Firewall*** link in main ***Networking*** page.

9.5. Click the ***Add Services...*** button located in the upper right side of the ***Firewall*** page.

9.6. In the ***Add Services*** user interface, scroll down or use ***Filter Services*** to locate and select the check box next to the ***Secure WWW (HTTPS)*** service.

9.7. Click the ***Add Services*** button located at the lower right side of the ***Add Services*** user interface.

10. Return to a terminal on `workstation` and verify your work by attempting to view the web server contents of `servera`.

    10.1. This command should fail:

```
[student@workstation ~]$ curl -k http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80: No
 route to host
```

    10.2. This command should succeed:

```
[student@workstation ~]$ curl -k https://servera.lab.example.com
I am servera.
```

> **NOTE**
> If you use Firefox to connect to the web server, it will prompt for verification of the host certificate if it successfully gets past the firewall.

Finish

On `workstation`, run the **`lab netsecurity-firewalls finish`** script to complete this exercise.

```
[student@workstation ~]$ lab netsecurity-firewalls finish
```

This concludes the guided exercise.

▷ LAB

MANAGING NETWORK SECURITY

PERFORMANCE CHECKLIST

In this lab, you will configure firewall and SELinux settings to allow access to multiple web servers running on `serverb`.

OUTCOMES

You should be able to configure firewall and SELinux settings on a web server host.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab netsecurity-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

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

Your company has decided to run a new web app. This application listens on ports **`80/TCP`** and **`1001/TCP`**. Port **`22/TCP`** for **`ssh`** access must also be available. All changes you make should persist across a reboot.

If prompted by **`sudo`**, use `student` as the password.

1. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.
2. Log in to `serverb` to determine what is preventing access to the web servers.
3. Configure SELinux to allow the `httpd` service to listen on port **`1001/TCP`**.
4. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.
5. Log in to `serverb` to determine whether the correct ports are assigned to the firewall.
6. Add port **`1001/TCP`** to the permanent configuration for the `public` network zone. Confirm your configuration.
7. From `workstation`, confirm that the default web server at `serverb.lab.example.com` returns **`SERVER B`** and the virtual host at `serverb.lab.example.com:1001` returns **`VHOST 1`**.

Evaluation

On `workstation`, run the **`lab netsecurity-review grade`** command to confirm success of this lab exercise.

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

Finish

On `workstation`, run the **`lab netsecurity-review finish`** script to complete this exercise.

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

This concludes the lab.

▷ SOLUTION

MANAGING NETWORK SECURITY

PERFORMANCE CHECKLIST

In this lab, you will configure firewall and SELinux settings to allow access to multiple web servers running on `serverb`.

OUTCOMES

You should be able to configure firewall and SELinux settings on a web server host.

BEFORE YOU BEGIN

Log in as the `student` user on `workstation` using `student` as the password.

From `workstation`, run the **`lab netsecurity-review start`** command. The command runs a start script to determine whether the `serverb` host is reachable on the network.

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

Your company has decided to run a new web app. This application listens on ports **`80/TCP`** and **`1001/TCP`**. Port **`22/TCP`** for **`ssh`** access must also be available. All changes you make should persist across a reboot.

If prompted by **sudo**, use `student` as the password.

1. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.

   1.1. Test access to the `http://serverb.lab.example.com` web server. The test currently fails. Ultimately, the web server should return **SERVER B**.

   ```
   [student@workstation ~]$ curl http://serverb.lab.example.com
   curl: (7) Failed to connect to serverb.lab.example.com port 80:
    Connection refused
   ```

   1.2. Test access to the `http://serverb.lab.example.com:1001` virtual host. The test currently fails. Ultimately, the virtual host should return **VHOST 1**.

   ```
   [student@workstation ~]$ curl http://serverb.lab.example.com:1001
   curl: (7) Failed to connect to serverb.lab.example.com port 1001: No
    route to host
   ```

2. Log in to `serverb` to determine what is preventing access to the web servers.

2.1. From `workstation`, open an SSH session to `serverb` as `student` user. The systems are configured to use SSH keys for authentication, so a password is not required.

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

2.2. Determine whether the `httpd` service is active.

```
[student@serverb ~]$ systemctl is-active httpd
inactive
```

2.3. Enable and start the `httpd` service. The `httpd` service fails to start.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
[sudo] password for student: student
Created symlink /etc/systemd/system/multi-user.target.wants/
httpd.service → /usr/lib/systemd/system/httpd.service.
Job for httpd.service failed because the control process exited with
 error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

2.4. Investigate the reasons why the `httpd.service` service failed to start.

```
[student@serverb ~]$ systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled;
 vendor preset: disabled)
   Active: failed (Result: exit-code) since Thu 2019-04-11 19:25:36
 CDT; 19s ago
     Docs: man:httpd.service(8)
  Process: 9615 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
 (code=exited, status=1/FAILURE)
 Main PID: 9615 (code=exited, status=1/FAILURE)
   Status: "Reading configuration..."

Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Starting The Apache
 HTTP Server...
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission
 denied: AH00072: make_sock: could not bind to address [::]:1001
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission
 denied: AH00072: make_sock: could not bind to address 0.0.0.0:1001
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: no listening
 sockets available, shutting down
```

```
Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: AH00015: Unable to
 open logs
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service: Main
 process exited, code=exited, status=1/FAILURE
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service:
 Failed with result 'exit-code'.
Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Failed to start The
 Apache HTTP Server.
```

2.5. Use the **sealert** command to check whether SELinux is blocking the `httpd` service from binding to port **1001/TCP**.

```
[student@serverb ~]$ sudo sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------

SELinux is preventing /usr/sbin/httpd from name_bind access on the
 tcp_socket port 1001.

*****  Plugin bind_ports (99.5 confidence) suggests
 ************************

If you want to allow /usr/sbin/httpd to bind to network port 1001
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 1001
    where PORT_TYPE is one of the following: http_cache_port_t,
 http_port_t, jboss_management_port_t, jboss_messaging_port_t,
 ntop_port_t, puppet_port_t.

*****  Plugin catchall (1.49 confidence) suggests
 *************************

...output omitted...
```

3. Configure SELinux to allow the `httpd` service to listen on port **1001/TCP**.

   3.1. Use the **semanage** command to find the correct port type.

```
[student@serverb ~]$ sudo semanage port -l | grep 'http'
http_cache_port_t      tcp  8080, 8118, 8123, 10001-10010
http_cache_port_t      udp  3130
http_port_t            tcp  80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t    tcp  5988
pegasus_https_port_t   tcp  5989
```

   3.2. Use the **semanage** command to bind port **1001/TCP** to the `http_port_t` type.

```
[student@serverb ~]$ sudo semanage port -a -t http_port_t -p tcp 1001
[student@serverb ~]$
```

   3.3. Confirm that port **1001/TCP** is bound to the `http_port_t` port type.

```
[student@serverb ~]$ sudo semanage port -l | grep '^http_port_t'
http_port_t            tcp  1001, 80, 81, 443, 488, 8008, 8009, 8443,
 9000
```

   3.4. Enable and start the `httpd` service.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
```

   3.5. Verify the running state of the `httpd` service.

```
[student@serverb ~]$ systemctl is-active httpd; systemctl is-enabled
 httpd
active
enabled
```

   3.6. Exit from `serverb`.

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

4. From `workstation`, test access to the default web server at `http://serverb.lab.example.com` and to the virtual host at `http://serverb.lab.example.com:1001`.

   4.1. Test access to the `http://serverb.lab.example.com` web server. The web server should return **SERVER B**.

```
[student@workstation ~]$ curl http://serverb.lab.example.com
SERVER B
```

   4.2. Test access to the `http://serverb.lab.example.com:1001` virtual host. The test continues to fail.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No
 route to host
```

5. Log in to `serverb` to determine whether the correct ports are assigned to the firewall.

   5.1. From `workstation`, log in to `serverb` as the `student` user.

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

   5.2. Verify that the default firewall zone is set to **`public`**.

```
[student@serverb ~]$ firewall-cmd --get-default-zone
public
```

   5.3. If the previous step did not return `public` as the default zone, correct it with the following command:

```
[student@serverb ~]$ sudo firewall-cmd --set-default-zone public
```

   5.4. Determine the open ports listed in the `public` network zone.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --
list-all
[sudo] password for student: student
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client http ssh
  ports:
  protocols:
  masquerade: no
```

```
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

6. Add port **1001/TCP** to the permanent configuration for the public network zone. Confirm your configuration.

   6.1. Add port **1001/TCP** to the public network zone.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public \
--add-port=1001/tcp
success
```

   6.2. Reload the firewall configuration.

```
[student@serverb ~]$ sudo firewall-cmd --reload
success
```

   6.3. Confirm your configuration.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --
list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client http ssh
  ports: 1001/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

   6.4. Exit from serverb.

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

7. From `workstation`, confirm that the default web server at `serverb.lab.example.com` returns **SERVER B** and the virtual host at `serverb.lab.example.com:1001` returns **VHOST 1**.

   7.1. Test access to the `http://serverb.lab.example.com` web server.

```
[student@workstation ~]$ curl http://serverb.lab.example.com
SERVER B
```

   7.2. Test access to the `http://serverb.lab.example.com:1001` virtual host.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
VHOST 1
```

Evaluation

On `workstation`, run the **`lab netsecurity-review grade`** command to confirm success of this lab exercise.

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

Finish

On `workstation`, run the **`lab netsecurity-review finish`** script to complete this exercise.

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

This concludes the lab.

SUMMARY

In this chapter, you learned:

- The `netfilter` subsystem allows kernel modules to inspect every packet traversing the system. All incoming, outgoing or forwarded network packets are inspected.
- The use of **`firewalld`** has simplified management by classifying all network traffic into zones. Each zone has its own list of ports and services. The **`public`** zone is set as the default zone.
- The `firewalld` service ships with a number of pre-defined services. They can be listed using the **`firewall-cmd --get-services`** command.

CHAPTER 16

COMPREHENSIVE REVIEW

GOAL

Review tasks from *RHCSA Rapid Track*

OBJECTIVES

- Review tasks from *RHCSA Rapid Track*

LAB

- Lab: Fixing Boot Issues and Maintaining Servers
- Lab: Configuring and Managing File Systems and Storage
- Lab: Configuring and Managing Server Security

COMPREHENSIVE REVIEW

OBJECTIVES

After completing this section, students should have reviewed and refreshed the knowledge and skills learned in *RHCSA Rapid Track*.

REVIEWING *RHCSA RAPID TRACK*

Before beginning the comprehensive review for this course, students 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, *Accessing Systems and Obtaining Support*

Log in to local and remote Linux system, and investigate problem resolution methods provided through Red Hat Support and Red Hat Insights.

- Log in to a Linux system on a local text console and run simple commands using the shell.
- Configure key-based authentication for a user account to log in to remote systems securely without a password.
- Describe key resources available through the Red Hat Customer Portal, and find information from Red Hat documentation and the Knowledgebase.
- Analyze servers for issues, remediate or resolve them, and confirm the solution with Red Hat Insights.

Chapter 2, *Navigating File Systems*

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

- Describe how Linux organizes files, and the purposes of various directories in the file-system hierarchy.
- Create, copy, move, and remove files and directories.
- Make multiple file names reference the same file using hard links and symbolic (or "soft") links.

Chapter 3, *Managing 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 using the **sudo** command.
- Create, modify, and delete locally defined user accounts.
- Create, modify, and delete locally defined group accounts.

- Set a password management policy for users, and manually lock and unlock user accounts.

Chapter 4, *Controlling Access to Files*

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

- Change the permissions and ownership of files using command-line tools.
- Control the default permissions of new files created by users, explain the effect of special permissions, and use special permissions and default permissions to set the group owner of files created in a particular directory.

Chapter 5, *Managing SELinux Security*

Protect and manage the security of a server by using SELinux.

- Describe how SELinux works and how to switch a server between its various enforcment modes.
- Adjust the SELinux type of a file in order to control which processes can access it.
- Change the accesses allowed by the SELinux policy by setting tunable parameters called SELinux booleans.
- Perform basic investigation and troubleshooting of accesses blocked by SELinux.

Chapter 6, *Tuning System Performance*

Evaluate and control processes, set tuning parameters and adjust process scheduling priorities on a Red Hat Enterprise Linux system.

- Control and terminate processes that are not associated with your shell, and forcibly end user sessions and processes.
- Describe what load average is and determine processes responsible for high resource use on a server.
- Optimize system performance by selecting a tuning profile managed by the tuned daemon.
- Prioritize or de-prioritize specific processes, with the nice and renice commands.

Chapter 7, *Installing and Updating Software Packages*

Download, install, update, and manage software packages from Red Hat and Yum package repositories.

- Register a system to your Red Hat account and assign it entitlements for software updates and support services using Red Hat Subscription Management.
- Find, install, and update software packages using the **yum** command.
- Enable and disable use of Red Hat or third-party Yum repositories by a server.
- Explain how modules allow installation of specific versions of software, list, enable, and switch module streams, and install and update packages from a module.

Chapter 8, *Managing Basic Storage*

Create and manage storage devices, partitions, file systems, and swap spaces from the command line.

- Explain what a block device is, interpret the file names of storage devices, and identify the storage device used by the file system for a particular directory or file.
- Access file systems by attaching them to a directory in the file system hierarchy.
- Create storage partitions, format them with the file systems, and mount them for use.
- Create and manage swap spaces to supplement physical memory.

Chapter 9, *Controlling Services and the Boot Process*

Control and monitor network services, system daemons and the boot process using `systemd`.

- List system daemons and network services started by the `systemd` service and socket units.
- Control system daemons and network services, using **`systemctl`**.
- Describe the Red Hat Enterprise Linux boot process, set the default target used when booting, and boot a system to a non-default target.
- Log into a system and change the root password when the current root password has been lost.
- Manually repair file system configuration or corruption issues that stop the boot process.

Chapter 10, *Managing Networking*

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

- Test and inspect current network configuration with command-line utilities.
- Manage network settings and devices using **`nmcli`**.
- Modify network settings by editing the configuration files.
- Configure a server's static host name and its name resolution, and test the results.

Chapter 11, *Analyzing and Storing Logs*

Locate and accurately interpret logs of system events for troubleshooting purposes.

- Describe the basic logging architecture used by Red Hat Enterprise Linux to record events.
- Interpret events in relevant syslog files to troubleshoot problems or review system status.
- Find and interpret entries in the system journal to troubleshoot problems or review system status.
- Configure the system journal to preserve the record of events when a server is rebooted.
- Maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

Chapter 12, *Implementing Advanced Storage Features*

Create and manage logical volumes containing file systems and swap spaces from the command line, and configure advanced storage features with Stratis and VDO.

- Create and manage logical volumes from storage devices, and format them with file systems or prepare them with swap spaces.

- Add and remove storage assigned to volume groups, and non-destructively extend the size of a logical volume formatted with an XFS or ext4 file system.
- Manage multiple storage layers at once using Stratis local storage management.
- Optimize use of storage space by using VDO to compress and deduplicate data on storage devices.

Chapter 13, *Scheduling Future Tasks*

Schedule tasks to automatically execute in the future.

- Schedule commands to run on a repeating schedule using the system crontab file and directories..
- Enable and disable systemd timers, and configure a timer that manages temporary files

Chapter 14, *Accessing Network-Attached Storage*

Access network-attached storage, using the NFS protocol.

- Mount, use, and unmount an NFS export from the command line and at boot.
- Configure the automater to automatically mount an NFS file system on demand, and unmount it when it is no longer in use.

Chapter 15, *Managing Network Security*

Control network connections to services using the system firewall.

- Accept or reject network connections to system services using firewalld rules.

▶ LAB

FIXING BOOT ISSUES AND MAINTAINING SERVERS

In this review, you will troubleshoot and repair boot problems and update the system default target. You will also schedule tasks to run on a repeating schedule as a normal user.

OUTCOMES

You should be able to:

- Diagnose issues and recover the system from emergency mode.
- Change the default target from `graphical.target` to `multi-user.target`.
- Schedule recurring jobs to run as a normal user.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now.

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

On `workstation`, run **`lab rhcsa-compreview1 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview1 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review:

- On `workstation`, run the **`lab rhcsa-compreview1 break1`** command. This break script causes the boot process to fail on `serverb`. It also sets a longer timeout on the GRUB2 menu to help interrupt the boot process, and reboots `serverb`.

  Troubleshoot the possible cause and repair the boot failure. The fix must ensure that `serverb` reboots without intervention. Use `redhat` as the password of the superuser, when required.

- On `workstation`, run the **`lab rhcsa-compreview1 break2`** command. This break script causes the default target to switch from the `multi-user` target to the `graphical` target on `serverb`. It also sets a longer timeout for the GRUB2 menu to help interrupt the boot process, and reboots `serverb`.

  On `serverb`, fix the default target to use the `multi-user` target. The default target settings must persist after reboot without manual intervention.

  Use the **sudo** command, as the `student` user with `student` as the password, for performing privileged commands.

- Schedule a recurring job as the `student` user that executes the **`/home/student/backup-home.sh`** script on an hourly basis between 7 p.m. and 9 p.m. on all days except Saturday and Sunday.

  Download the backup script from http://materials.example.com/labs/backup-home.sh. The **`backup-home.sh`** backup script backs up the **`/home/student`** directory from `serverb` to `servera` in the **`/home/student/serverb-backup`** directory. Use the **`backup-home.sh`** script to schedule the recurring job as the `student` user on `serverb`.

- Reboot the system and wait for the boot to complete before grading.

Evaluation

On `workstation`, run the **`lab rhcsa-compreview1 grade`** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview1 grade
```

Finish

On `workstation`, run **`lab rhcsa-compreview1 finish`** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview1 finish
```

Save any files or work you want to keep to other systems, and then reset `workstation`, `servera`, and `serverb` before the next exercise.

This concludes the comprehensive review.

▸ SOLUTION

FIXING BOOT ISSUES AND MAINTAINING SERVERS

In this review, you will troubleshoot and repair boot problems and update the system default target. You will also schedule tasks to run on a repeating schedule as a normal user.

OUTCOMES

You should be able to:

- Diagnose issues and recover the system from emergency mode.
- Change the default target from `graphical.target` to `multi-user.target`.
- Schedule recurring jobs to run as a normal user.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now.

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

On `workstation`, run **`lab rhcsa-compreview1 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview1 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review:

- On `workstation`, run the **`lab rhcsa-compreview1 break1`** command. This break script causes the boot process to fail on `serverb`. It also sets a longer timeout on the GRUB2 menu to help interrupt the boot process, and reboots `serverb`.

  Troubleshoot the possible cause and repair the boot failure. The fix must ensure that `serverb` reboots without intervention. Use `redhat` as the password of the superuser, when required.

- On `workstation`, run the **`lab rhcsa-compreview1 break2`** command. This break script causes the default target to switch from the `multi-user` target to the `graphical` target on `serverb`. It also sets a longer timeout for the GRUB2 menu to help interrupt the boot process, and reboots `serverb`.

  On `serverb`, fix the default target to use the `multi-user` target. The default target settings must persist after reboot without manual intervention.

  Use the **sudo** command, as the `student` user with `student` as the password, for performing privileged commands.

- Schedule a recurring job as the `student` user that executes the **`/home/student/backup-home.sh`** script on an hourly basis between 7 p.m. and 9 p.m. on all days except Saturday and Sunday.

  Download the backup script from http://materials.example.com/labs/backup-home.sh. The **`backup-home.sh`** backup script backs up the **`/home/student`** directory from `serverb` to `servera` in the **`/home/student/serverb-backup`** directory. Use the **`backup-home.sh`** script to schedule the recurring job as the `student` user on `serverb`.

- Reboot the system and wait for the boot to complete before grading.

1. On `workstation`, run the **`lab rhcsa-compreview1 break1`** command.

```
[student@workstation ~]$ lab rhcsa-compreview1 break1
```

2. After `serverb` boots up, access the console and notice that the boot process stopped early. Take a minute to speculate about a possible cause for this behavior.

   2.1. Locate the icon for the `serverb` console, as appropriate for your classroom environment. Open the console.

   2.2. Looking at the error, it appears that at least parts of the system are still functioning.

   2.3. Press **`Ctrl`**+**`Alt`**+**`Del`** to reboot `serverb`.

   When the boot-loader menu appears, press any key except **`Enter`** to interrupt the countdown.

   2.4. Edit the default boot-loader entry, in memory, to log in to emergency mode.

   Press **e** to edit the current entry.

   2.5. Use the cursor keys to navigate to the line that starts with **`linux`**. Append **`systemd.unit=emergency.target`** to the end of the line.

   2.6. Press **`Ctrl`**+**`x`** to boot using the modified configuration.

   2.7. Log in to emergency mode. The `root` password is `redhat`.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

3. Remount the **/** file system read/write. Use the **`mount -a`** command to attempt to mount all the other file systems.

   3.1. Remount the **/** file system read/write to edit the file system.

```
[root@serverb ~]# mount -o remount,rw /
```

   3.2. Use the **`mount -a`** command to attempt to mount all the other file systems. Notice that one of the file systems cannot be mounted.

```
[root@serverb ~]# mount -a
mount: /FakeMount: can't find UUID=fake.
```

3.3. Edit **/etc/fstab** to fix the issue. Remove or comment out the incorrect line.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
#UUID=fake     /FakeMount  xfs   defaults    0 0
```

3.4. Update systemd for the system to register the new **/etc/fstab** configuration.

```
[root@serverb ~]# systemctl daemon-reload
[ 206.828912] systemd[1]: Reloading.
```

3.5. Verify that **/etc/fstab** is now correct by attempting to mount all entries.

```
[root@serverb ~]# mount -a
```

3.6. Reboot serverb and wait for the boot to complete. The system should now boot normally.

```
[root@serverb ~]# systemctl reboot
```

4. On workstation, run the **lab rhcsa-compreview1 break2** command.

```
[student@workstation ~]$ lab rhcsa-compreview1 break2
```

Wait for the reboot to complete before proceeding.

5. On `serverb`, switch to the `multi-user` target. Set the default target to `multi-user`. Use the **sudo** command to run any required administrative command and if prompted, use `student` as the password.

   5.1. From `workstation`, open an SSH session to `serverb` as the `student` user.

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

   5.2. As the `student` user on `serverb`, determine the default target.

```
[student@serverb ~]$ systemctl get-default
graphical.target
```

   5.3. Switch to the `multi-user` target. Use the **sudo** command and if prompted, use `student` as the password.

```
[student@serverb ~]$ sudo systemctl isolate multi-user.target
[sudo] password for student: student
```

   5.4. Set `serverb` to use the `multi-user` target as the default target.

```
[student@serverb ~]$ sudo systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/
system/multi-user.target.
```

   5.5. Reboot `serverb` to verify that the `multi-user` target is set as the default target.

```
[student@serverb ~]$ sudo systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

   5.6. After reboot, open an SSH session to `serverb` as the `student` user. Verify that the `multi-user` target is set as the default target.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ systemctl get-default
multi-user.target
```

6. Schedule a recurring job as the `student` user that executes the **/home/student/backup-home.sh** script on an hourly basis between 7 p.m. and 9 p.m. on all days except Saturday and Sunday.

   Use the **backup-home.sh** script to schedule the recurring job. Download the backup script from http://materials.example.com/labs/backup-home.sh.

   6.1. On `serverb`, download the backup script from http://materials.example.com/labs/backup-home.sh. Use **chmod** to make the backup script executable.

```
[student@serverb ~]$ wget http://materials.example.com/labs/backup-
home.sh
...output omitted...
[student@serverb ~]$ chmod +x backup-home.sh
```

   6.2. Use the **crontab -e** command to open the crontab file using the default text editor.

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

   6.3. Edit the file to add the following line:

```
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

   Save the changes and exit the editor.

   6.4. Use the **crontab -l** command to list the scheduled recurring jobs.

```
[student@serverb ~]$ crontab -l
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

7. Reboot `serverb` and wait for the boot to complete before grading.

```
[student@serverb ~]$ sudo systemctl reboot
[sudo] password for student: student
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

Evaluation

On `workstation`, run the **lab rhcsa-compreview1 grade** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview1 grade
```

Finish

On `workstation`, run **lab rhcsa-compreview1 finish** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview1 finish
```

Save any files or work you want to keep to other systems, and then reset `workstation`, `servera`, and `serverb` before the next exercise.

This concludes the comprehensive review.

▷ LAB

CONFIGURING AND MANAGING FILE SYSTEMS AND STORAGE

In this review, you will create an LVM logical volume, mount a network file system, create a swap partition that is automatically activated at boot, and configure temporary unused files to be cleaned from the system.

OUTCOMES

You should be able to:

- Create an LVM logical volume.
- Mount a network file system.
- Create a swap partition that is automatically activated at boot.
- Configure temporary unused files to be cleaned from the system.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now, unless you just finished resetting them at the end of the last exercise.

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

On `workstation`, run **`lab rhcsa-compreview2 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview2 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review.

- Configure a new 1 GiB logical volume called `vol_home` in a new 2 GiB volume group called `extra_storage`. Use the unpartitioned `/dev/vdb` disk to create partitions.
- The logical volume `vol_home` should be formatted with the XFS file-system type, and mounted persistently on **`/home-directories`**.
- Ensure that the network file system called **`/share`** is persistently mounted on **`/local-share`** across reboot. The NFS server `servera.lab.example.com` exports the **`/share`** network file system. The NFS export path is `servera.lab.example.com:/share`.
- Create a new 512 MiB partition on the `/dev/vdc` disk to be used as swap space. This swap space must be automatically activated at boot.
- Create a new group called `production`. Create the `production1`, `production2`, `production3`, and `production4` users. Ensure that they use the new group called `production` as their supplementary group.

- Configure your system so that it uses a new directory called **/run/volatile** to store temporary files. Files in this directory should be subject to time based cleanup if they are not accessed for more than 30 seconds. The octal permissions for the directory must be **0700**. Make sure that you use the **/etc/tmpfiles.d/volatile.conf** file to configure the time based cleanup for the files in **/run/volatile**.

Evaluation

On workstation, run the **lab rhcsa-compreview2 grade** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview2 grade
```

Finish

On workstation, run **lab rhcsa-compreview2 finish** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview2 finish
```

This concludes the comprehensive review.

▷ SOLUTION

CONFIGURING AND MANAGING FILE SYSTEMS AND STORAGE

In this review, you will create an LVM logical volume, mount a network file system, create a swap partition that is automatically activated at boot, and configure temporary unused files to be cleaned from the system.

OUTCOMES

You should be able to:

- Create an LVM logical volume.
- Mount a network file system.
- Create a swap partition that is automatically activated at boot.
- Configure temporary unused files to be cleaned from the system.

BEFORE YOU BEGIN

Copy any files or work you wish to keep to other systems before resetting. Reset the `workstation`, `servera`, and `serverb` systems now, unless you just finished resetting them at the end of the last exercise.

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

On `workstation`, run **`lab rhcsa-compreview2 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview2 start
```

INSTRUCTIONS

Perform the following tasks on `serverb` to complete the comprehensive review.

- Configure a new 1 GiB logical volume called `vol_home` in a new 2 GiB volume group called `extra_storage`. Use the unpartitioned `/dev/vdb` disk to create partitions.
- The logical volume `vol_home` should be formatted with the XFS file-system type, and mounted persistently on **`/home-directories`**.
- Ensure that the network file system called **`/share`** is persistently mounted on **`/local-share`** across reboot. The NFS server `servera.lab.example.com` exports the **`/share`** network file system. The NFS export path is `servera.lab.example.com:/share`.
- Create a new 512 MiB partition on the `/dev/vdc` disk to be used as swap space. This swap space must be automatically activated at boot.
- Create a new group called `production`. Create the `production1`, `production2`, `production3`, and `production4` users. Ensure that they use the new group called `production` as their supplementary group.

- Configure your system so that it uses a new directory called **/run/volatile** to store temporary files. Files in this directory should be subject to time based cleanup if they are not accessed for more than 30 seconds. The octal permissions for the directory must be **0700**. Make sure that you use the **/etc/tmpfiles.d/volatile.conf** file to configure the time based cleanup for the files in **/run/volatile**.

1. From workstation, open an SSH session to serverb as student.

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

2. Switch to the root user.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3. Create a 2 GiB partition on /dev/vdb.

```
[root@serverb ~]# parted /dev/vdb mklabel msdos
[root@serverb ~]# parted /dev/vdb mkpart primary 1GiB 3GiB
```

4. Create a logical volume called vol_home using the 2 GiB partition you created on /dev/vdb. Name the volume group extra_storage.

   4.1. Declare the /dev/vdb1 block device as a physical volume.

```
[root@serverb ~]# pvcreate /dev/vdb1
...output omitted...
```

   4.2. Create the extra_storage volume group using /dev/vdb1.

```
[root@serverb ~]# vgcreate extra_storage /dev/vdb1
...output omitted...
```

   4.3. Create a 1 GiB logical volume named vol_home.

```
[root@serverb ~]# lvcreate -L 1GiB -n vol_home extra_storage
...output omitted...
```

5. Format vol_home with the XFS file-system type, and mount it on **/home-directories**.

   5.1. Create a directory called **/home-directories**.

```
[root@serverb ~]# mkdir /home-directories
```

   5.2. Format /dev/extra_storage/vol_home with the XFS file-system type.

```
[root@serverb ~]# mkfs -t xfs /dev/extra_storage/vol_home
...output omitted...
```

5.3. Persistently mount `/dev/extra_storage/vol_home` on **`/home-directories`**. Use the structure's UUID when creating the entry in **`/etc/fstab`**.

```
[root@serverb ~]# lsblk -o UUID /dev/extra_storage/vol_home
UUID
988cf149-0667-4733-abca-f80c6ec50ab6
[root@serverb ~]# echo "UUID=988c...0ab6 /home-directories \
xfs defaults 0 0" >> /etc/fstab
[root@serverb ~]# mount -a
```

6. Ensure that the network file system called **`/share`** is persistently mounted on **`/local-share`** across reboot. The NFS server `servera.lab.example.com` exports the **`/share`** network file system. The NFS export path is `servera.lab.example.com:/share`.

6.1. Create the **`/local-share`** directory.

```
[root@serverb ~]# mkdir /local-share
```

6.2. Append the appropriate entry to **`/etc/fstab`** so that the network file system available at `servera.lab.example.com:/share` is persistently mounted on **`/local-share`** across reboot.

```
[root@serverb ~]# echo "servera.lab.example.com:/share /local-share \
nfs rw,sync 0 0" >> /etc/fstab
```

6.3. Mount the network file system on **`/local-share`** based on the entry in **`/etc/fstab`**.

```
[root@serverb ~]# mount /local-share
```

7. Create a new 512 MiB partition on the `/dev/vdc` disk to be used as swap space. This swap space must be automatically activated at boot time.

   7.1. Create a 512 MiB partition on `/dev/vdc`.

```
[root@serverb ~]# parted /dev/vdc mklabel msdos
[root@serverb ~]# parted /dev/vdc mkpart primary 1MiB 513MiB
```

   7.2. Make the swap space on `/dev/vdc1`.

```
[root@serverb ~]# mkswap /dev/vdc1
...output omitted...
```

   7.3. Activate the swap space so that it persists across reboot. Use the structure's UUID when creating the entry in **`/etc/fstab`**.

```
[root@serverb ~]# lsblk -o UUID /dev/vdc1
UUID
cc18ccb6-bd29-48a5-8554-546bf3471b69
[root@serverb ~]# echo "UUID=cc18...1b69 swap \
swap defaults 0 0" >> /etc/fstab
[root@serverb ~]# swapon -a
```

8. Create the `production1`, `production2`, `production3`, and `production4` users. Ensure that they use the new group called `production` as their supplementary group.

```
[root@serverb ~]# groupadd production
[root@serverb ~]# for i in 1 2 3 4; do useradd -G production production$i;
 done
```

9. Configure your system so that it uses a new directory called **`/run/volatile`** to store temporary files. Files in this directory should be subject to time based cleanup if they are not accessed for more than 30 seconds. The octal permissions for the directory must be **`0700`**.

Make sure that you use the **/etc/tmpfiles.d/volatile.conf** file to configure the time based cleanup for the files in **/run/volatile**.

9.1. Create a file called **/etc/tmpfiles.d/volatile.conf** with the following content.

```
d /run/volatile 0700 root root 30s
```

9.2. Use the **systemd-tmpfiles --create** command to create the **/run/volatile** directory if it does not exist.

```
[root@servera ~]# systemd-tmpfiles --create /etc/tmpfiles.d/
volatile.conf
```

9.3. Exit the root user's shell.

```
[root@serverb ~]# exit
logout
```

9.4. Log off from serverb.

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

Evaluation

On workstation, run the **lab rhcsa-compreview2 grade** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview2 grade
```

Finish

On workstation, run **lab rhcsa-compreview2 finish** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview2 finish
```

This concludes the comprehensive review.

▶ LAB

CONFIGURING AND MANAGING SERVER SECURITY

In this review, you will configure SSH key-based authentication, change firewall settings, adjust the SELinux mode and an SELinux Boolean, and troubleshoot SELinux issues.

OUTCOMES

You should be able to:

- Configure SSH keys for key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab rhcsa-compreview3 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview3 start
```

INSTRUCTIONS

Perform the following tasks to complete the comprehensive review:

- Generate SSH keys for the `student` user on `serverb`. Do not protect the private key with a passphrase.
- On `servera`, configure the `student` user to accept login authentication using the SSH key pair created for `student` on `serverb`. The `student` user on `serverb` should be able to log in to `servera` using SSH without entering a password. Use `student` as the password of the `student` user, if required.
- On `servera`, change the default SELinux mode to **`permissive`**.
- Configure `serverb` to automatically mount the home directory of the `production5` user when the user logs in, using the network file system **`/home-directories/production5`**. This network file system is exported from `servera.lab.example.com`. Adjust the appropriate SELinux Boolean so that `production5` can use the NFS-mounted home directory on `serverb` after authenticating via SSH key-based authentication. The `production5` user's password is `redhat`.
- On `serverb`, adjust the firewall settings so that the SSH connections originating from `servera` are rejected.

- On `serverb`, investigate and fix the issue with the Apache HTTPD daemon, which is configured to listen on port `30080/TCP`, but which fails to start. Adjust the firewall settings appropriately so that the port `30080/TCP` is open for incoming connections.

Evaluation

On `workstation`, run the **`lab rhcsa-compreview3 grade`** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview3 grade
```

Finish

On `workstation`, run **`lab rhcsa-compreview3 finish`** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview3 finish
```

Save any files or work you want to keep to other systems, and then reset `workstation`, `servera`, and `serverb`.

This concludes the comprehensive review.

SOLUTION

CONFIGURING AND MANAGING SERVER SECURITY

In this review, you will configure SSH key-based authentication, change firewall settings, adjust the SELinux mode and an SELinux Boolean, and troubleshoot SELinux issues.

OUTCOMES

You should be able to:

- Configure SSH keys for key-based authentication.
- Configure firewall settings.
- Adjust the SELinux mode and SELinux Booleans.
- Troubleshoot SELinux issues.

BEFORE YOU BEGIN

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

On `workstation`, run **`lab rhcsa-compreview3 start`** to start the comprehensive review. This script creates the necessary files to set up the environment correctly.

```
[student@workstation ~]$ lab rhcsa-compreview3 start
```

INSTRUCTIONS

Perform the following tasks to complete the comprehensive review:

- Generate SSH keys for the `student` user on `serverb`. Do not protect the private key with a passphrase.
- On `servera`, configure the `student` user to accept login authentication using the SSH key pair created for `student` on `serverb`. The `student` user on `serverb` should be able to log in to `servera` using SSH without entering a password. Use `student` as the password of the `student` user, if required.
- On `servera`, change the default SELinux mode to **`permissive`**.
- Configure `serverb` to automatically mount the home directory of the `production5` user when the user logs in, using the network file system **`/home-directories/production5`**. This network file system is exported from `servera.lab.example.com`. Adjust the appropriate SELinux Boolean so that `production5` can use the NFS-mounted home directory on `serverb` after authenticating via SSH key-based authentication. The `production5` user's password is `redhat`.
- On `serverb`, adjust the firewall settings so that the SSH connections originating from `servera` are rejected.

- On `serverb`, investigate and fix the issue with the Apache HTTPD daemon, which is configured to listen on port `30080/TCP`, but which fails to start. Adjust the firewall settings appropriately so that the port `30080/TCP` is open for incoming connections.

1. From `workstation`, open an SSH session to `serverb` as `student`.

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

2. Generate SSH keys for the `student` user on `serverb` using the **ssh-keygen** command. Do not protect the private key with a passphrase.

```
[student@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1TPZ4TXYwiGWfExUGtRTHgfKQbF9hVuLa+VmH4vgkFY
 student@serverb.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|          .+@BO**|
|          .=.#+B*|
|          . X.*o=|
|         . E +.+ |
|        S o   +  |
|         + . o = |
|        . o o + +|
|           . . ..|
|                 |
+----[SHA256]-----+
```

3. On `servera`, configure the `student` user to accept login authentication using the SSH key pair you created for `student` on `serverb`. The `student` user on `serverb` should be able to log in to `servera` using SSH without entering a password. Use `student` as the password of the `student` user, when required.

   3.1. Use the **ssh-copy-id** command to transfer the public key of the SSH key pair of `student` on `serverb` to `student` on `servera`. Use `student` as the password of the `student` user, if prompted.

```
[student@serverb ~]$ ssh-copy-id student@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
student/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be
 established.
```

```
ECDSA key fingerprint is SHA256:g/
fIMtVzDWTbTi1l0OwC30sL6cHmro9Tf563NxmeyyE.
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
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. Use the **ssh** command to verify that the `student` user can log in to `servera` from `serverb` without entering a password.

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

4. On `servera`, change the default SELinux mode to **`permissive`**.

4.1. Edit **`/etc/sysconfig/selinux`** to set the value of the parameter **`SELINUX`** to **`permissive`**. You can use the **`sudo vi /etc/sysconfig/selinux`** command to edit the configuration file as the superuser. Use the password `student`, if prompted.

```
...output omitted...
#SELINUX=enforcing
SELINUX=permissive
...output omitted...
```

4.2. Use the **`sudo systemctl reboot`** command to reboot the system as the superuser.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@serverb ~]$
```

5. Configure `serverb` to automatically mount the home directory of the `production5` user when the user logs in, using the network file system **`/home-directories/production5`**. This network file system is exported from `servera.lab.example.com`. Adjust the appropriate SELinux Boolean so that `production5` can use the NFS-mounted home

directory on `serverb` after authenticating via SSH key-based authentication. The `production5` user's password is `redhat`.

5.1. On `serverb`, use the **`sudo -i`** command to switch to the `root` user account.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

5.2. Install the *autofs* package.

```
[root@serverb ~]# yum install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  autofs-1:5.1.4-29.el8.x86_64

Complete!
```

5.3. Create the `autofs` master map file called **`/etc/auto.master.d/production5.autofs`** with the following content.

```
/- /etc/auto.production5
```

5.4. Create the **`/etc/auto.production5`** file with the following content.

```
/localhome/production5 -rw servera.lab.example.com:/home-directories/
production5
```

5.5. Restart the `autofs` service.

```
[root@serverb ~]# systemctl restart autofs
```

6. On servera, verify that the production5 user is not able to log in to serverb using SSH public-key authentication. An SELinux Boolean causes this issue which you will fix in the following steps.

   6.1. From workstation, open an SSH session to servera as student.

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

   6.2. Switch to the production5 user using the password redhat.

```
[student@servera ~]$ su - production5
Password: redhat
[production5@servera ~]$
```

   6.3. Use the **ssh-keygen** command to generate the SSH keys as production5.

```
[production5@servera ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/production5/.ssh/
id_rsa): Enter
Created directory '/home/production5/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/production5/.ssh/id_rsa.
Your public key has been saved in /home/production5/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:zmin1nmCt4H8LA+4FPimtdg81nl7ATbInUFW3HSPxk4
 production5@servera.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|       .oo.o. .  |
|       .. . .o o |
|     . o o    E .|
|   .  o *    +   |
|  . .  .So    .  |
|   . + =  .      |
```

```
|    *.*+=. .      |
|   Oo+***.o       |
|  o.=o.=**        |
+----[SHA256]-----+
```

6.4. Use the **ssh-copy-id** command to transfer the public key of the SSH key pair of production5 on servera to production5 on serverb. Use redhat as the password of the production5 user, if prompted.

```
[production5@servera ~]$ ssh-copy-id production5@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
production5/.ssh/id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be
 established.
ECDSA key fingerprint is
 SHA256:ciCkaRWF4g6eR9nSdPxQ7KL8czpViXal6BousK544TY.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
 to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
 are prompted now it is to install the new keys
production5@serverb's password: redhat

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'production5@serverb'"
and check to make sure that only the key(s) you wanted were added.
```

6.5. Use the SSH public key-based authentication instead of password-based authentication to log in to serverb as production5. This command should fail.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
production5@serverb: Permission denied (publickey,gssapi-keyex,gssapi-
with-mic,password).
```

7. Set the appropriate SELinux Boolean setting on serverb, so that production5 can log in to serverb using the SSH public key-based authentication and use the home directory.

7.1. On serverb as root, set the use_nfs_home_dirs SELinux Boolean to **true**.

```
[root@serverb ~]# setsebool -P use_nfs_home_dirs true
```

7.2. Use the SSH public key-based authentication instead of password-based authentication to log in to serverb as production5. This command should succeed.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
...output omitted...
[production5@serverb ~]$
```

8. On `serverb`, adjust the firewall settings so that SSH connections originating from `servera` are rejected. The `servera` system uses the IPv4 address `172.25.250.10`.

   8.1. Use the **`firewall-cmd`** command to add the IPv4 address of `servera` to the `firewalld` zone called `block`.

```
[root@serverb ~]# firewall-cmd --add-source=172.25.250.10/32 \
--zone=block --permanent
success
```

   8.2. Use the **`firewall-cmd --reload`** command to reload the changes in the firewall settings.

```
[root@serverb ~]# firewall-cmd --reload
success
```

9. On `serverb`, investigate and fix the issue with the Apache HTTPD daemon, which is configured to listen on port `30080/TCP`, but which fails to start. Adjust the firewall settings appropriately so that port `30080/TCP` is open for incoming connections.

   9.1. Use the **`systemctl`** command to restart the `httpd` service. This command fails to restart the service.

```
[root@serverb ~]# systemctl restart httpd.service
Job for httpd.service failed because the control process exited with
 error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

   9.2. Use the **`systemctl status`** command to investigate the reason for the failure of the `httpd` service.

```
[root@serverb ~]# systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled;
 vendor preset: disabled)
   Active: failed (Result: exit-code) since Mon 2019-04-15 06:42:41
 EDT; 5min ago
     Docs: man:httpd.service(8)
  Process: 27313 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
 (code=exited, status=1/FAILURE)
 Main PID: 27313 (code=exited, status=1/FAILURE)
   Status: "Reading configuration..."

Apr 15 06:42:41 serverb.lab.example.com systemd[1]: Starting The Apache
 HTTP Server...
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: (13)Permission
 denied: AH00072: make_sock: could not bind to address [::]:30080
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: (13)Permission
 denied: AH00072: make_sock: could not bind to address 0.0.0.0:30080
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: no listening
 sockets available, shutting down
```

```
Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: AH00015: Unable
 to open logs
Apr 15 06:42:41 serverb.lab.example.com systemd[1]: httpd.service: Main
 process exited, code=exited, status=1/FAILURE
Apr 15 06:42:41 serverb.lab.example.com systemd[1]: httpd.service:
 Failed with result 'exit-code'.
Apr 15 06:42:41 serverb.lab.example.com systemd[1]: Failed to start The
 Apache HTTP Server.
```

Notice the permission error in the preceding output, which signifies that the `httpd` daemon failed to bind to port `30080/TCP`. The SELinux policy can be a potential restriction for an application to bind to a port. Press **q** to quit the preceding **`systemctl`** command.

9.3. Use the **`sealert`** command to determine if an SELinux policy is preventing `httpd` from binding to port `30080/TCP`.

```
[root@serverb ~]# sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
--------------------------------------------------------------------------------

SELinux is preventing /usr/sbin/httpd from name_bind access on the
 tcp_socket port 30080.

*****  Plugin bind_ports (92.2 confidence) suggests
 ************************

If you want to allow /usr/sbin/httpd to bind to network port 30080
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 30080
    where PORT_TYPE is one of the following: http_cache_port_t,
 http_port_t, jboss_management_port_t, jboss_messaging_port_t,
 ntop_port_t, puppet_port_t.
```

The preceding log message reveals that the port `30080/TCP` does not have the appropriate SELinux context **`http_port_t`**, causing SELinux to prevent `httpd` to

bind to this port. The log message also produces the syntax of the **semanage port** command so that you can easily fix the issue.

9.4. Use the **semanage port** command to set the appropriate SELinux context on the port 30080/TCP for `httpd` to bind to it.

```
[root@serverb ~]# semanage port -a -t http_port_t -p tcp 30080
```

9.5. Use the **systemctl** command to restart `httpd`. This command should successfully restart the service.

```
[root@serverb ~]# systemctl restart httpd
```

9.6. Add the port 30080/TCP to the default `firewalld` zone called `public`.

```
[root@serverb ~]# firewall-cmd --add-port=30080/tcp --permanent
success
[root@serverb ~]# firewall-cmd --reload
success
```

9.7. Exit the `root` user's shell.

```
[root@serverb ~]# exit
logout
```

9.8. Log off from `serverb`.

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

Evaluation

On `workstation`, run the **lab rhcsa-compreview3 grade** script to confirm success on this exercise. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab rhcsa-compreview3 grade
```

Finish

On `workstation`, run **lab rhcsa-compreview3 finish** to complete this exercise. This script deletes the files and resources created throughout the exercise and ensures that the environment is clean.

```
[student@workstation ~]$ lab rhcsa-compreview3 finish
```

Save any files or work you want to keep to other systems, and then reset `workstation`, `servera`, and `serverb`.

This concludes the comprehensive review.