

**RED HAT®  
TRAINING**



*Comprehensive, hands-on training that solves real world problems*

# **RHCSA Rapid Track**

---

## **Student Workbook**





# **RHCSA**

## **RAPID TRACK**

# Red Hat Enterprise Linux 7 RH199

## RHCSA Rapid Track

### Edition 1

Authors: Wander Boessenkool, Bruce Wolfe, Scott McBrien, George Hacker,  
Chen Chang, Philip Sweany, Susan Lauber, Rudolf Kastl  
Editor: Scott McBrien

Copyright © 2014 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2014 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](mailto:training@redhat.com) or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the 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.

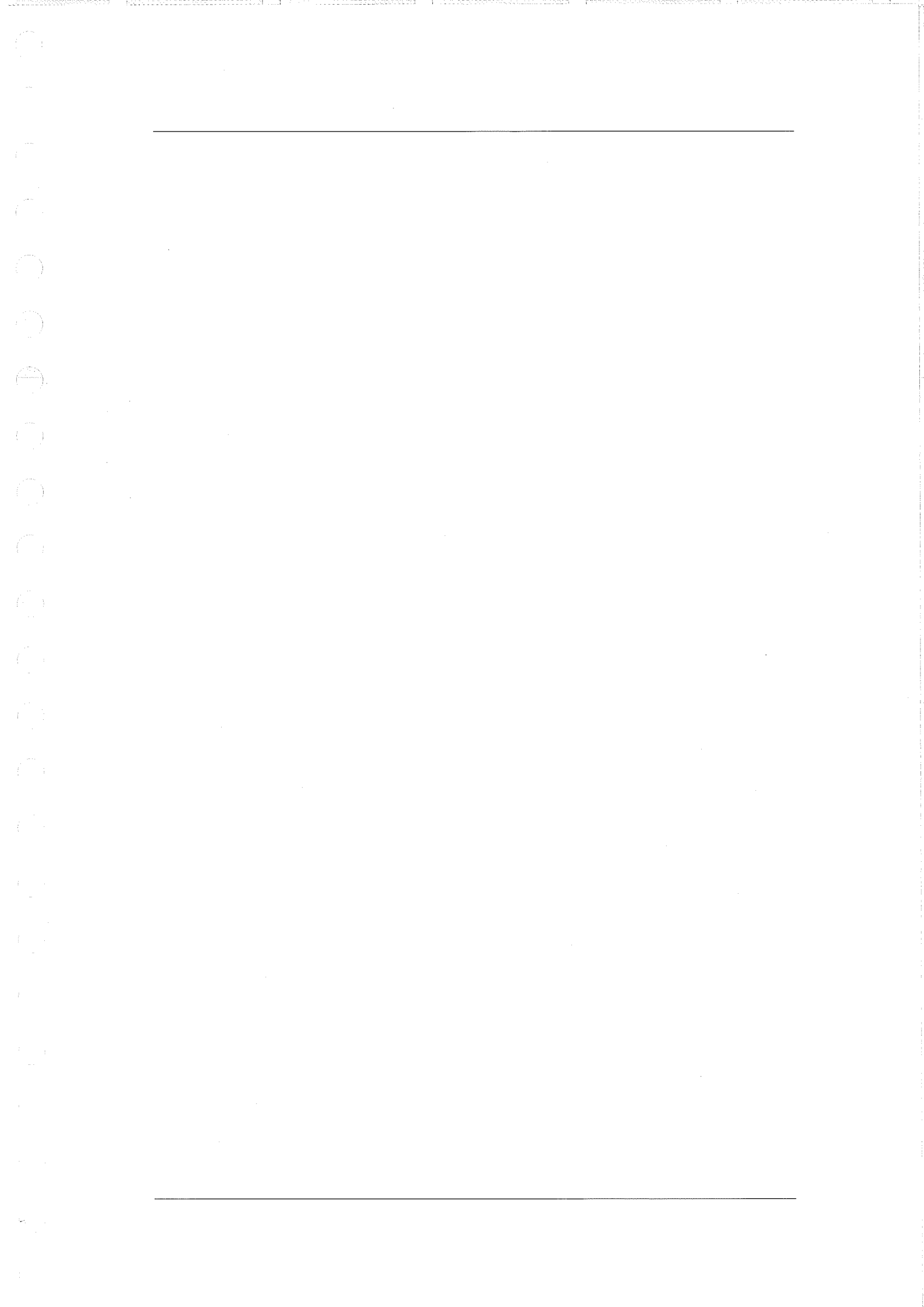
The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the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

All other trademarks are the property of their respective owners.

Contributors: Rob Locke, Bowe Strickland, Forrest Taylor, Steve Bonneville

Reviewers: Christian Linden, Mathias Homann

---



---

---

<b>Document Conventions</b>	<b>xi</b>
Notes and Warnings .....	xi
<b>Introduction</b>	<b>xiii</b>
RHCSA Rapid Track .....	xiii
Orientation to the Classroom Environment .....	xiv
Internationalization .....	xv
<b>1. Local and Remote Logins</b>	<b>1</b>
Accessing the Command Line Using the Local Console .....	2
Practice: Local Console Access Terms .....	5
Configuring SSH Key-based Authentication .....	8
Practice: Using SSH Key-based Authentication .....	10
Getting Help From Red Hat .....	11
Practice: Creating and Viewing an SoS Report .....	17
<b>2. File System Navigation</b>	<b>21</b>
The Linux File System Hierarchy .....	22
Practice: File System Hierarchy .....	25
Managing Files Using Command-Line Tools .....	28
Practice: Command-Line File Management .....	33
Making Links Between Files .....	36
Practice: Making Links Between Files .....	38
<b>3. Users and Groups</b>	<b>41</b>
Users and Groups .....	42
Practice: User and Group Concepts .....	44
Gaining Superuser Access .....	46
Practice: Running Commands as root .....	50
Managing Local User Accounts .....	53
Practice: Creating Users Using Command-line Tools .....	56
Managing Local Group Accounts .....	58
Practice: Managing Groups Using Command-line Tools .....	60
Managing User Passwords .....	62
Practice: Managing User Password Aging .....	65
Using Identity Management Services .....	67
Practice: Connecting to a Central LDAP and Kerberos Server .....	73
Lab: Managing Linux Users and Groups .....	75
<b>4. File Permissions</b>	<b>83</b>
Managing File System Permissions from the Command Line .....	84
Practice: Managing File Security from the Command Line .....	87
Managing Default Permissions and File Access .....	89
Practice: Controlling New File Permissions and Ownership .....	93
POSIX Access Control Lists (ACLs) .....	95
Practice: Interpret ACLs .....	100
Securing Files with ACLs .....	103
Practice: Using ACLs to Grant and Limit Access .....	107
<b>5. SELinux Permissions</b>	<b>113</b>
Enabling and Monitoring Security Enhanced Linux (SELinux) .....	114
Practice: SELinux Concepts .....	118
Changing SELinux Modes .....	120
Practice: Changing SELinux Modes .....	122

---

Changing SELinux Contexts .....	123
Practice: Changing SELinux Contexts .....	125
Changing SELinux Booleans .....	127
Practice: Changing SELinux Booleans .....	129
Troubleshooting SELinux .....	131
Practice: Troubleshooting SELinux .....	134
Lab: Managing SELinux Security .....	137
<b>6. Process Management .....</b>	<b>143</b>
Killing Processes .....	144
Practice: Killing Processes .....	149
Monitoring Process Activity .....	151
Practice: Monitoring Process Activity .....	155
Using nice and renice to Influence Process Priority .....	157
Practice: Discovering Process Priorities .....	159
Lab: Managing Priority of Linux Processes .....	162
<b>7. Updating Software Packages .....</b>	<b>165</b>
Attaching Systems to Subscriptions for Software Updates .....	166
Practice: Red Hat Subscription Management .....	172
Managing Software Updates with <b>yum</b> .....	174
Practice: Installing and Updating Software with <b>yum</b> .....	181
Enabling <b>yum</b> Software Repositories .....	185
Practice: Enabling Software Repositories .....	188
Lab: Installing and Updating Software Packages .....	190
<b>8. Creating and Mounting File Systems .....</b>	<b>193</b>
Mounting and Unmounting File Systems .....	194
Practice: Mounting and Unmounting File Systems .....	197
Adding Partitions, File Systems, and Persistent Mounts .....	199
Practice: Adding Partitions, File Systems, and Persistent Mounts .....	210
Managing Swap Space .....	213
Practice: Adding and Enabling Swap Space .....	217
Lab: Adding Disks, Partitions, and File Systems to a Linux System .....	221
<b>9. Service Management and Boot Troubleshooting .....</b>	<b>229</b>
Identifying Automatically Started System Processes .....	230
Practice: Identify the Status of <b>systemd</b> Units .....	234
Controlling System Services .....	236
Practice: Using <b>systemctl</b> to Manage Services .....	239
The Red Hat Enterprise Linux Boot Process .....	241
Practice: Selecting a Boot Target .....	245
Repairing Common Boot Issues .....	247
Practice: Resetting a Lost root Password .....	250
Repairing File System Issues at Boot .....	252
Practice: Repairing Boot Problems .....	253
Repairing Boot Loader Issues .....	255
Practice: Repairing a Boot Loader Problem .....	257
Lab: Controlling Services and Daemons .....	259
<b>10. Network Configuration .....</b>	<b>263</b>
Validating Network Configuration .....	264
Practice: Examining Network Configuration .....	267
Configuring Networking with <b>nmcli</b> .....	269

Practice: Configuring Networking with <b>nmcli</b> .....	274
Editing Network Configuration Files .....	277
Practice: Editing Network Configuration Files .....	278
Configuring Host Names and Name Resolution .....	280
Practice: Configuring Host Names and Name Resolution .....	283
Lab: Managing Red Hat Enterprise Linux Networking .....	286
<b>11. System Logging and NTP .....</b>	<b>289</b>
System Log Architecture .....	290
Practice: System Logging Components .....	292
Reviewing Syslog Files .....	294
Practice: Finding Log Entries .....	298
Reviewing systemd Journal Entries .....	299
Practice: Finding Events With journalctl .....	302
Preserving the systemd Journal .....	303
Practice: Configure a Persistent systemd Journal .....	305
Maintaining Accurate Time .....	306
Practice: Adjusting System Time .....	310
Lab: Analyzing and Storing Logs .....	312
<b>12. Logical Volume Management .....</b>	<b>317</b>
Managing Logical Volumes .....	318
Practice: Adding a Logical Volume .....	324
Extending Logical Volumes .....	328
Practice: Extending a Logical Volume .....	333
Lab: Managing Logical Volume Management (LVM) Storage .....	337
<b>13. Scheduled Processes .....</b>	<b>343</b>
Scheduling System cron Jobs .....	344
Practice: Scheduling System cron Jobs .....	346
Managing Temporary Files .....	348
Practice: Managing Temporary Files .....	351
<b>14. Mounting Network File Systems .....</b>	<b>355</b>
Mounting Network Storage with NFS .....	356
Practice: Mounting and Unmounting NFS .....	359
Automounting Network Storage with NFS .....	363
Practice: Automounting NFS .....	367
Accessing Network Storage with SMB .....	371
Practice: Mounting a SMB File System .....	375
Lab: Accessing Network Storage with Network File System (NFS) .....	377
Lab: Accessing Network Storage with SMB .....	382
<b>15. Firewall Configuration .....</b>	<b>391</b>
Limiting Network Communication .....	392
Practice: Limiting Network Communication .....	399
Lab: Limiting Network Communication .....	401
<b>16. Virtualization and Kickstart .....</b>	<b>405</b>
Defining the Anaconda Kickstart System .....	406
Practice: Kickstart File Syntax and Modification .....	412
Deploying a New Virtual System with Kickstart .....	416
Practice: Installing a System Using Kickstart .....	420
Managing a Local Virtualization Host .....	424

Practice: Managing a Local Virtualization Host .....	430
--	-----



---

# Document Conventions

## Notes and Warnings



### 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.



### Comparison

"Comparisons" look at similarities and differences between the technology or topic being discussed and similar technologies or topics in other operating systems or environments.



### References

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



### 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

In this course, students will do most hands-on practice exercises and lab work with two computer systems, which will be referred to as **desktop** and **server**. These machines have the hostnames `desktopX.example.com` and `serverX.example.com`, where the *X* in the computers' hostnames will be a number that will vary from student to student. Both machines have a standard user account, *student*, with the password *student*. The *root* password on both systems is *redhat*.

In a live instructor-led classroom, students will be assigned a physical computer ("foundationX") which will be used to access these two machines. The **desktop** and **server** systems are virtual machines running on that host. Students should log in to this machine as user *kiosk* with the password *redhat*.

On foundationX, a special command called **rht-vmctl** is used to work with the **desktop** and **server** machines. The commands in the table below should be run as the *kiosk* user on foundationX, and can be used with **server** (as in the examples) or **desktop**.

### **rht-vmctl** commands

Action	Command
Start <b>server</b> machine	<b>rht-vmctl start server</b>
View "physical console" to log in and work with <b>server</b> machine	<b>rht-vmctl view server</b>
Reset <b>server</b> machine to its previous state and restarts virtual machine	<b>rht-vmctl reset server</b>

At the start of a lab exercise, if the instruction "reset your server" appears, that means the command **rht-vmctl reset server** should be run in a prompt on the foundationX system as user *kiosk*. Likewise, if the instruction "reset your desktop" appears, that means the command **rht-vmctl reset desktop** should be run on foundationX as user *kiosk*.

Each student is on the IPv4 network 172.25.X.0/24, where the *X* matches the number of their `desktopX` and `serverX` systems. The instructor runs a central utility server which acts as a router for the classroom networks and which provides DNS, DHCP, HTTP, and other content services, `classroom.example.com`.

### **Classroom Machines**

Machine name	IP addresses	Role
<code>desktopX.example.com</code>	172.25.X.10	Student "client" computer
<code>serverX.example.com</code>	172.25.X.11	Student "server" computer
<code>classroom.example.com</code>	172.25.254.254	Classroom utility server

# Internationalization

## Language support

Red Hat Enterprise Linux 7 officially supports 22 languages: English, Assamese, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Kannada, Korean, Malayalam, Marathi, Odia, Portuguese (Brazilian), Punjabi, Russian, Spanish, Tamil, and Telugu.

## Per-user language selection

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

### Language settings

In the GNOME desktop environment, the user may 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. Run the command **gnome-control-center region**, or from the top bar, select **(User) > Settings**. In the window that opens, select **Region & Language**. The user can click the **Language** box and select their preferred language from the list that appears. This will also update the **Formats** setting to the default for that language. The next time the user logs in, these changes will take full effect.

These settings affect the GNOME desktop environment and any applications, including **gnome-terminal**, started inside it. However, they do not apply to that account if accessed through an **ssh** login from a remote system or a local text console (such as **tty2**).



### Note

A user can make their shell environment use the same **LANG** setting as their graphical environment, even when they log in through a text console or over **ssh**. One way to do this is to place code similar to the following in the user's **~/ .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=$(grep 'Language=' /var/lib/AccountService/users/${USER} \
| sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Japanese, Korean, Chinese, or other languages with a non-Latin character set may not display properly on local text 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 24 17:55:01 CDT 2014

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

### Input method settings

GNOME 3 in Red Hat Enterprise Linux 7 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's 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.

Once 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 find also 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 the user knows the character's Unicode code point, by typing **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, *root* can change the system-wide locale settings with the **localectl** command. If **localectl** is run with no arguments, it will display the current system-wide locale settings.

To set the system-wide language, run the command **localectl set-locale LANG=locale**, where *locale* is the appropriate **\$LANG** 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** and clicking the **Login Screen** button at the upper-right corner of the window. Changing the **Language** of the login screen will also adjust the system-wide default language setting stored in the **/etc/locale.conf** configuration file.



## Important

Local text consoles such as **ttty2** are more limited in the fonts that they can display than **gnome-terminal** and **ssh** sessions. For example, Japanese, Korean, and Chinese characters may not display as expected on a local text console. For this reason, it may make sense to use English or another language with a Latin character set for the system's text console.

Likewise, local text 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 local text virtual consoles and the X11 graphical environment. See the **localectl(1)**, **kbd(4)**, and **vconsole.conf(5)** man pages for more information.

## Language packs

When using non-English languages, you may want to install additional "language packs" to provide additional translations, dictionaries, and so forth. To view the list of available langpacks, run **yum langavailable**. To view the list of langpacks currently installed on the system, run **yum langlist**. To add an additional langpack to the system, run **yum langinstall code**, where *code* is the code in square brackets after the language name in the output of **yum langavailable**.



## References

**locale(7)**, **localectl(1)**, **kbd(4)**, **locale.conf(5)**, **vconsole.conf(5)**, **unicode(7)**, **utf-8(7)**, and **yum-langpacks(8)** 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

### Language Codes

Language	\$LANG value
English (US)	en_US.utf8
Assamese	as_IN.utf8
Bengali	bn_IN.utf8
Chinese (Simplified)	zh_CN.utf8
Chinese (Traditional)	zh_TW.utf8
French	fr_FR.utf8
German	de_DE.utf8
Gujarati	gu_IN.utf8
Hindi	hi_IN.utf8
Italian	it_IT.utf8
Japanese	ja_JP.utf8
Kannada	kn_IN.utf8
Korean	ko_KR.utf8
Malayalam	ml_IN.utf8
Marathi	mr_IN.utf8
Odia	or_IN.utf8
Portuguese (Brazilian)	pt_BR.utf8
Punjabi	pa_IN.utf8
Russian	ru_RU.utf8
Spanish	es_ES.utf8
Tamil	ta_IN.utf8
Telugu	te_IN.utf8





## CHAPTER 1

# LOCAL AND REMOTE LOGINS

Overview	
<b>Goal</b>	To review methods for accessing the system and engaging Red Hat Support.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Use Bash shell syntax to enter commands at a Linux console.</li><li>• Set up ssh to allow secure password-free logins by using a private authentication key file.</li><li>• Use the redhat-support-tool command.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Accessing the Command Line Using the Local Console (and Practice)</li><li>• Configuring SSH Key-based Authentication (and Practice)</li><li>• Getting Help From Red Hat (and Practice)</li></ul>

# Accessing the Command Line Using the Local Console

## Objectives

After completing this section, students should be able to log into a Linux system on a local text console and run simple commands using the shell.

## 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*. Over the long history of UNIX-like systems, many shells have been developed. 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.

```
[student@desktopX ~]$
```

The **\$** is replaced by a **#** 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 in the privileged account.

```
[root@desktopX ~]#
```

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. Mac OS X administrators who use the Macintosh's **Terminal** utility may be pleased to note that **bash** is the default shell in Mac OS X.

## Virtual consoles

Users access the **bash** shell through a *terminal*. A terminal provides a keyboard for user input and a display for output. On text-based installations, this can be the Linux machine's *physical console*, the hardware keyboard and display. Terminal access can also be configured through serial ports.

Another way to access a shell is from a *virtual console*. A Linux machine's physical console supports multiple virtual consoles which act like separate terminals. Each virtual console supports an independent login session.

If the graphical environment is available, it will run on the *first* virtual console in Red Hat Enterprise Linux 7. Five additional text login prompts are available on consoles two through six (or one through five if the graphical environment is turned off). With a graphical environment running, access a text login prompt on a virtual console by holding **Ctrl+Alt** and pressing a function key (**F2** through **F6**). Press **Ctrl+Alt+F1** to return to the first virtual console and the graphical desktop.



### Important

In the pre-configured virtual images delivered by Red Hat, login prompts have been disabled in the virtual consoles.



### Note

In Red Hat Enterprise Linux 5 and earlier, the first *six* virtual consoles always provided text login prompts. When the graphical environment was launched, it ran on virtual console seven (accessed through **Ctrl+Alt+F7**).

## 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 on.

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

To use a command effectively, a user needs to know what options and arguments it takes and in what order it expects them (the *syntax* of the command). Most commands have a **--help** option. This causes the command to print a description of what it does, a "usage statement" that describes the command's syntax, and a list of the options it accepts and what they do.

Usage statements may seem complicated and difficult to read. They become much simpler to understand once a user becomes familiar with a few basic conventions:

- Square brackets, **[ ]**, surround optional items.
- Anything followed by **...** represents an arbitrary-length list of items of that type.
- Multiple items separated by pipes, **|**, means only *one* of them can be specified.

- Text in angle brackets, `<>`, represents variable data. For example, `<filename>` means “insert the filename you wish to use here”. Sometimes these variables are simply written in capital letters (e.g., `FILENAME`).

Consider the first usage statement for the **date** command:

```
[student@desktopX ~]$ date --help
date [OPTION]... [+FORMAT]
```

This indicates that **date** can take an optional list of options (**[OPTION]...**), followed by an optional format string, prefixed with a plus character, **+**, that defines how the current date should be displayed (**[+FORMAT]**). Since both of these are optional, **date** will work even if it is not given options or arguments (it will print the current date and time using its default format).



## Note

The **man** page for a command has a SYNOPSIS section that provides information about the command's syntax. The **man-pages(7)** man page describes how to interpret all the square brackets, vertical bars, and so forth that users see in SYNOPSIS or a usage message.

When a user is finished using the shell and wants to quit, there are a couple of ways to end the session. The **exit** command terminates the current shell session. Another way to finish a session is by typing **Ctrl+D**.



## References

**intro(1)**, **bash(1)**, **console(4)**, **pts(4)**, and **man-pages(7)** man pages

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

## Practice: Local Console Access Terms

### Quiz

Match the following items to their counterparts in the table.

Argument	Command	Option	Physical console
Prompt	Shell	Terminal	Virtual console

Description	Term
The interpreter that executes commands typed as strings.	Shell
The visual cue that indicates an interactive shell is waiting for the user to type a command.	Prompt
The name of a program to run.	Command
The part of the command line that adjusts the behavior of a command.	Option
The part of the command line that specifies the target that the command should operate on.	Argument
The hardware display and keyboard used to interact with a system.	Physical console
One of multiple logical consoles that can each support an independent login session.	Virtual console

Description	Term
An interface that provides a display for output and a keyboard for input to a shell session.	Terminal

## Solution

Match the following items to their counterparts in the table.

Description	Term
The interpreter that executes commands typed as strings.	Shell
The visual cue that indicates an interactive shell is waiting for the user to type a command.	Prompt
The name of a program to run.	Command
The part of the command line that adjusts the behavior of a command.	Option
The part of the command line that specifies the target that the command should operate on.	Argument
The hardware display and keyboard used to interact with a system.	Physical console
One of multiple logical consoles that can each support an independent login session.	Virtual console
An interface that provides a display for output and a keyboard for input to a shell session.	Terminal

# Configuring SSH Key-based Authentication

## Objective

After completing this section, students should be able to set up SSH to allow secure logins without passwords by using a private authentication key file.

## SSH key-based authentication

Users can authenticate **ssh** logins without a password by using *public key authentication*. **ssh** allows users to authenticate using a private-public key scheme. This means that two keys are generated, a private key and a 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 log into, and is used to verify the private key. The public key does not need to be secret. An SSH server that has the public key can issue a challenge that can only be answered by a system holding your private key. As a result, you can authenticate using the presence of your key. This allows you to access systems in a way that doesn't require typing a password every time, but is still secure.

Key generation is done using the **ssh-keygen** command. This generates the private key `~/.ssh/id_rsa` and the public key `~/.ssh/id_rsa.pub`.



### Note

During key generation, there is the option to specify a passphrase which must be provided in order to access your private key. In the event the private key is stolen, it is very difficult for someone other than the issuer to use it when protected with a passphrase. This adds enough of a time buffer to make a new key pair and remove all references to the old keys before the private key can be used by an attacker who has cracked it.

It is always wise to passphrase-protect the private key since the key allows access to other machines. However, this means the passphrase must be entered whenever the key is used, making the authentication process no longer password-less. This can be avoided using **ssh-agent**, which can be given your passphrase once at the start of the session (using **ssh-add**), so it can provide the passphrase as needed while you stay logged in.

For additional information on the **ssh-agent** command, consult the Red Hat System Administration Guide, Chapter 8.2.4.2.: Configuring ssh-agent.

Once the SSH keys have been generated, they are stored by default in the `.ssh/` directory of your home directory. Permissions should be 600 on the private key and 644 on the public key.

Before key-based authentication can be used, the public key needs to be copied to the destination system. This can be done with **ssh-copy-id**.

```
[student@desktopX ~]$ ssh-copy-id root@desktopY
```

When the key is copied to another system using **ssh-copy-id**, it copies the `~/.ssh/id_rsa.pub` file by default.



## SSH key demonstration

- Use **ssh-keygen** to create a public-private key pair.

```
[student@desktopX ~]$ 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): redhat
Enter same passphrase again: redhat
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:
a4:49:cf:fb:ac:ab:c8:ce:45:33:f2:ad:69:7b:d2:5a student@desktopX.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|
|      . .
|     . *
|    . * S
|   + + .
|  o . E
| o  oo+oo
| .,.**000
+-----+
```

- Use **ssh-copy-id** to copy the public key to the correct location on a remote system. For example:

```
[student@desktopX ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@serverX.example.com
```



## References

Additional information may be available in the chapter on using key-based authentication in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at <http://docs.redhat.com/>

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

# Practice: Using SSH Key-based Authentication

## Guided exercise

In this lab, you will set up SSH key-based authentication.

### Outcomes:

Students will set up SSH user key-based authentication to initiate SSH connections.

- ❑ 1. Create an SSH key pair as **student** on desktopX using no passphrase.

```
[student@desktopX ~]$ 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.
...
```

- ❑ 2. Send the SSH public key to the **student** account on serverX.

```
[student@desktopX ~]$ ssh-copy-id serverX
The authenticity of host 'serverX (172.25.X.11)' can't be established.
ECDSA key fingerprint is 33:fa:a1:3c:98:30:ff:f6:d4:99:00:4e:7f:84:3e:c3.
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@serverX's password: student

Number of key(s) added: 1

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

- ❑ 3. Run the **hostname** command by using **ssh** to display the hostname of the serverX.example.com machine without the need to enter a password.

```
[student@desktopX ~]$ ssh serverX 'hostname'
serverX.example.com
```

# Getting Help From Red Hat

## Objectives

After completing this section, students should be able to view Knowledgebase information and manage support cases from the command line.

## Red Hat Customer Portal

Red Hat Customer Portal (<https://access.redhat.com>) provides customers with access to everything provided with their subscription through one convenient location. Customers can search for solutions, FAQs, and articles through Knowledgebase. Access to official product documentation is provided. Support tickets can be submitted and managed. Subscriptions to Red Hat products can be attached to and detached from registered systems, and software downloads, updates, and evaluations can be obtained. Parts of the site are accessible to everyone, while others are exclusive to customers with active subscriptions. Help with getting access to Customer Portal is available at <https://access.redhat.com/help/>.

Customers can work with Red Hat Customer Portal through a web browser. This section will introduce a command line tool that can also be used to access Red Hat Customer Portal services, **redhat-support-tool**.

### Knowledgebase

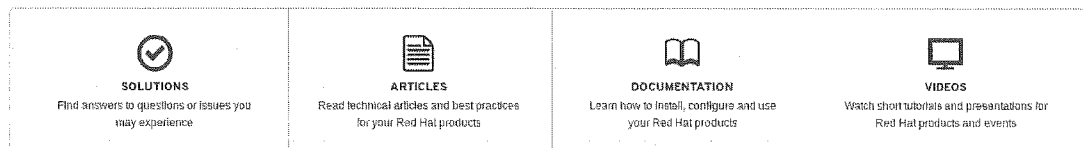


Figure 1.1: Knowledgebase at the Red Hat Customer Portal

## Using redhat-support-tool to search Knowledgebase

The Red Hat Support Tool utility **redhat-support-tool** provides a text console interface to the subscription-based Red Hat Access services. Internet access is required to reach the Red Hat Customer Portal. The **redhat-support-tool** is text-based for use from any terminal or SSH connection; no graphical interface is provided.

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

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

When first invoked, **redhat-support-tool** prompts for required Red Hat Access subscriber login information. To avoid repetitively supplying this information, the tool asks to store account information in the user's home directory (**\$HOME/.redhat-support-tool/redhat-support-tool.conf**). If a Red Hat Access account is shared by many users, the **--global** option can save account information to

`/etc/redhat-support-tool.conf`, along with other systemwide configuration. The tool's `config` command modifies tool configuration settings.

The `redhat-support-tool` allows subscribers to search and display the same Knowledgebase content seen when on the Red Hat Customer Portal. Knowledgebase permits keyword searches, similar to the `man` command. Users 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:

```
[student@desktopX ~]$ 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 Red Hat Network
(RHN) using Red Hat Subscription Manager (RHSM)?
2 [ 17397:VER] What are Flex Guest Entitlements in Red Hat Network?
3 [ 232863:VER] How to register machines and manage subscriptions using Red
Hat Subscription Manager through an invisible HTTP proxy / Firewall?
3 of 43 solutions displayed. Type 'm' to see more, 'r' to start from the beginning
again, or '?' for help with the codes displayed in the above output.
Select a Solution:
```

Specific sections of solution documents may be selected for viewing.

```
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 Red Hat Network (RHN) using Red Hat
Subscription Manager (RHSM)?
URL:      https://access.redhat.com/site/solutions/253273
(END) q
[student@desktopX ~]$
```

### Directly access Knowledgebase articles by document ID

Locate online articles directly using the tool's `kb` command with the Knowledgebase document ID. Returned documents scroll on the screen without pagination, allowing a user to redirect the output using other local commands. This example views the document with the `less` command:

```
[student@desktopX ~]$ redhat-support-tool kb 253273 | less
```

```
Title:   How to register and subscribe a system to Red Hat Network (RHN) using Red Hat
Subscription Manager (RHSM)?
ID:      253273
State:   Verified: This solution has been verified to work by Red Hat Customers and
Support Engineers for the specified product version(s).
URL:     https://access.redhat.com/site/solutions/253273
: q
```

Documents retrieved in unpaginated format are easy to send to a printer, convert to PDF or other document format, or to redirect to a data entry program for an incident tracking or change management system, using other utilities installed and available in Red Hat Enterprise Linux.

## Using redhat-support-tool to manage support cases

One benefit of a product subscription is access to technical support through Red Hat Customer Portal. Depending on the support level of the system's subscription, Red Hat may be contacted through on-line tools or by phone. See [https://access.redhat.com/site/support/policy/support\\_process](https://access.redhat.com/site/support/policy/support_process) for links to detailed information about the support process.

### Preparing a bug report

Before contacting Red Hat Support, gather relevant information for your bug report.

*Define the problem.* Be able to clearly state the problem and its symptoms. Be as specific as possible. Detail the steps which 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**, which will be discussed later in this section. In the case of kernel problems, this could include the **kdump** crash dump from a system or even a digital photo of the monitor of a crashed system which is displaying a kernel backtrace.

*Determine the severity level of your issue.* Red Hat uses four severity levels to classify issues. These four levels follow. Reports of *Urgent* and *High* severity problems should be followed with a phone call to the relevant local support center (see <https://access.redhat.com/site/support/contact/technicalSupport>).

Severity	Description
<i>Urgent</i> (Severity 1):	A problem that severely impacts your use of the software in a production environment (such as loss of production data or in which your production systems are not functioning). The situation halts your business operations and no procedural workaround exists.
<i>High</i> (Severity 2):	A problem where the software is functioning but your use in a production environment is severely reduced. The situation is causing a high impact to portions of your business operations and no procedural workaround exists.
<i>Medium</i> (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-

Severity	Description
	to-low impact on your business, but your business continues to function, including by using a procedural workaround. For development environments, where the situation is causing your project to no longer continue or migrate 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, including by using a procedural workaround.

### Managing a bug report with **redhat-support-tool**

Subscribers may create, view, modify, and close Red Hat Support cases using **redhat-support-tool**. When support cases are opened or maintained, users may include files or documentation, such as diagnostic reports (sosreport). The tool uploads and attaches files to online cases. Case details including *product*, *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, the **--product** and **--version** options are specified, but **redhat-support-tool** would provide a list of choices for those options if the **opencase** command did not specify them.

```
[student@desktopX ~]$ 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 Low
 2 Normal
 3 High
 4 Urgent
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.
```

### Including diagnostic information by attaching a SoS report archive

Including diagnostic information when a support case is first created contributes to quicker problem resolution. 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 is not already prepared, an administrator can generate and attach one later, using the tool's **addattachment** command as advised previously. This section's practice exercise will provide the steps for creating and viewing a current SoS diagnostic report.

Support cases can also be viewed, modified, and closed by you as 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

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

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*, a report of the active stack frames at the point of a crash dump, to provide onsite diagnostics and 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, oVirt, and others, can be parsed to recognize problem symptoms, which can then 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.



## 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/](https://access.redhat.com/site/support/policy/support_process/)

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



# Practice: Creating and Viewing an SoS Report

## Guided exercise

In this lab, you will use the `sosreport` command to generate a SoS report, then view the contents of that diagnostic archive.

### Outcomes

A compressed tar archive of systemwide diagnostic information.

### Before you begin...

Perform the following steps on `serverX` unless directed otherwise.

- ❑ 1. If currently working as a non-root user, switch to root.

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

- ❑ 2. Run the **sosreport** command. This may take many minutes on larger systems.

```
[root@serverX ~]# sosreport

sosreport (version 3.0)

This command will collect system configuration and
diagnostic information from this Red Hat Enterprise Linux
system. An archive containing the collected information
will be generated in /var/tmp and may be provided to a Red
Hat support representative or used for local diagnostic or
recording purposes.

Any information provided to Red Hat will be treated in
strict confidence in accordance with the published support
policies at:

    https://access.redhat.com/support/

The generated archive may contain data considered
sensitive and its content should be reviewed by the
originating organization before being passed to any third party.

No changes will be made to system configuration.

Press ENTER to continue, or CTRL-C to quit. ENTER

Please enter your first initial and last name [serverX.example.com]: yourname
Please enter the case number that you are generating this report for: 01034421
```

Press **Enter**. Provide the requested information. Make up a value for the case number.

```
Running 17/74: general...
Creating compressed archive...

Your sosreport has been generated and saved in:
/var/tmp/sosreport-yourname.01034421-20140129000049.tar.xz
```

```
The checksum is: b2e78125290a4c791162e68da8534887  
Please send this file to your support representative.
```

- 3. Change directory to **/var/tmp**, and unpack the archive.

```
[root@serverX ~]# cd /var/tmp  
[root@serverX tmp]# tar -xvJf sosreport-*.tar.xz
```

- 4. Change directory to the resulting subdirectory and browse the files found there.

```
[root@serverX ~]# cd sosreport-yourname.01034421-20140129000049  
[root@serverX sosreport-yourname.01034421-20140129000049]# ls -lR
```

Open files, list directories, and continue to browse to become familiar with the information included in SoS reports. In the form of the original archived and compressed file, this is the diagnostic information you would be attaching to a **redhat-support-tool** support case. When finished, remove the archive directory and files and return to your home directory.

```
[root@serverX sosreport-yourname.01034421-20140129000049]# cd /var/tmp  
[root@serverX tmp]# rm -rf sosreport*  
[root@serverX tmp]# exit  
[student@serverX ~]$
```

## Summary

### Accessing the Command Line Using the Local Console

Use the physical console to view output and input commands with correct syntax using the **bash** shell.

### Configuring SSH Key-based Authentication

Utilizing key-based SSH authentication adds additional security to remote systems administration.

### Getting Help From Red Hat

Use `redhat-support-tool` to look up Red Hat Knowledgebase articles and manage support cases.





## CHAPTER 2

# FILE SYSTEM NAVIGATION

Overview	
<b>Goal</b>	To copy, move, create, delete, link and organize files while working from the Bash shell prompt.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Identify the purpose for important directories on a Linux system.</li><li>• Create, copy, move, and remove files and directories using command-line utilities.</li><li>• Use hard links and symlinks to make multiple names.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• The Linux File System Hierarchy (and Practice)</li><li>• Managing Files Using Command-Line Tools (and Practice)</li><li>• Making Links Between Files (and Practice)</li></ul>

# The Linux File System Hierarchy

## Objectives

After completing this section, students should be able to understand fundamental file system layout, organization, and the location of key file types.

## 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.

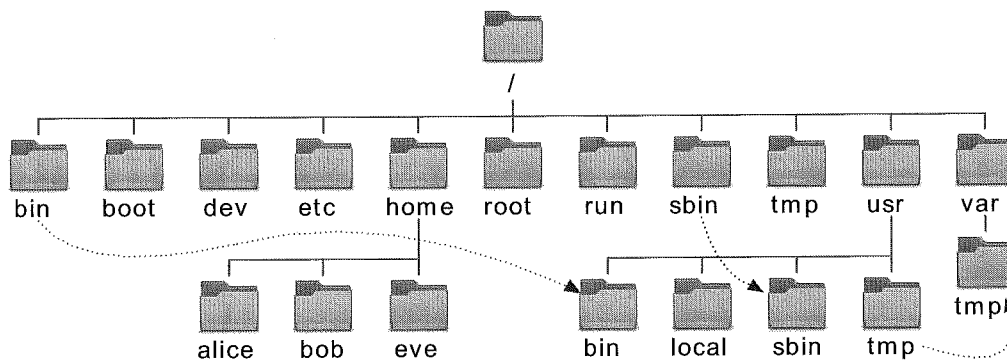


Figure 2.1: Significant file system directories in Red Hat Enterprise Linux 7

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, we could call that directory `/etc`. Likewise, if the `/etc` directory contained a file named **issue**, we 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 are encountered in describing file system directory contents:

- *static* is content that remains unchanged until explicitly edited or reconfigured.
- *dynamic* or *variable* is content typically modified or appended by active processes.
- *persistent* is content, particularly configuration settings, that remain after a reboot.
- *runtime* is process- or system-specific content or attributes cleared during 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
<b>/usr</b>	Installed software, shared libraries, include files, and static read-only program data. Important subdirectories include: - <b>/usr/bin</b> : User commands. - <b>/usr/sbin</b> : System administration commands. - <b>/usr/local</b> : Locally customized software.
<b>/etc</b>	Configuration files specific to this system.
<b>/var</b>	Variable data specific to this system that should persist between boots. Files that dynamically change (e.g. databases, cache directories, log files, printer-spoiled documents, and website content) may be found under <b>/var</b> .
<b>/run</b>	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 <b>/var/run</b> and <b>/var/lock</b> from older versions of Red Hat Enterprise Linux.)
<b>/home</b>	<i>Home directories</i> where regular users store their personal data and configuration files.
<b>/root</b>	Home directory for the administrative superuser, root.
<b>/tmp</b>	A world-writable space for temporary files. Files which are more than 10 days old are deleted from this directory automatically. Another temporary directory exists, <b>/var/tmp</b> , in which files that have not been accessed, changed, or modified in more than 30 days are deleted automatically.
<b>/boot</b>	Files needed in order to start the boot process.
<b>/dev</b>	Contains special <i>device files</i> which are used by the system to access hardware.

**Important**

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

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

In older versions of Red Hat Enterprise Linux, these were distinct directories containing different sets of files. In RHEL 7, the directories in **/** are symbolic links to the matching directories in **/usr**.



## References

**hier(7)** man page

Filesystem Hierarchy Standard  
<http://www.pathname.com/fhs>



## Practice: File System Hierarchy

### Quiz

Match the following items to their counterparts in the table.

/	/etc	/home	/root	/run	/tmp	/usr
/usr/bin	/usr/sbin	/var				

Directory purpose	Location
This directory contains static, persistent system configuration data.	/etc
This is the system's root directory.	/
User home directories are located under this directory.	/home
This is the root account's home directory.	/root
This directory contains dynamic configuration data, such as FTP and websites.	/var
Regular user commands and utilities are located here.	/usr/bin
System administration binaries, for root use, are here.	/usr/sbin
Temporary files are stored here.	/tmp

Directory purpose	Location
Contains dynamic, non-persistent application runtime data.	<i>/run</i>
Contains installed software programs and libraries.	<i>/usr</i>

## Solution

Match the following items to their counterparts in the table.

Directory purpose	Location
This directory contains static, persistent system configuration data.	/etc
This is the system's root directory.	/
User home directories are located under this directory.	/home
This is the root account's home directory.	/root
This directory contains dynamic configuration data, such as FTP and websites.	/var
Regular user commands and utilities are located here.	/usr/bin
System administration binaries, for root use, are here.	/usr/sbin
Temporary files are stored here.	/tmp
Contains dynamic, non-persistent application runtime data.	/run
Contains installed software programs and libraries.	/usr

# Managing Files Using Command-Line Tools

## Objectives

After completing this section, students should be able to create, copy, link, move, and remove files and subdirectories in various directories.

## Command-line file management

File management involves creating, deleting, copying, and moving files. Additionally, directories can be created, deleted, copied, and moved to help organize files logically. When working at the command line, file management requires awareness of the current working directory to choose either absolute or relative path syntax as most efficient for the immediate task.

### File management commands

Activity	Single source <sup>(note)</sup>	Multiple source <sup>(note)</sup>
Copy file	cp file1 file2	cp file1 file2 file3 dir <sup>(5)</sup>
Move file	mv file1 file2 <sup>(2)</sup>	mv file1 file2 file3 dir <sup>(5)</sup>
Remove file	rm file1	rm -f file1 file2 file3 <sup>(6)</sup>
Create directory	mkdir dir	mkdir -p par1/par2/dir <sup>(7)</sup>
Copy directory	cp -r dir1 dir2 <sup>(3)</sup>	cp -r dir1 dir2 dir3 dir4 <sup>(5)</sup>
Move directory	mv dir1 dir2 <sup>(4)</sup>	mv dir1 dir2 dir3 dir4 <sup>(5)</sup>
Remove directory	rm -r dir1 <sup>(3)</sup>	rm -rf dir1 dir2 dir3 <sup>(6)</sup>
Note:	<sup>(1)</sup> The result is one file with two real names. <sup>(2)</sup> The result is a rename. <sup>(3)</sup> The "recursive" option is required to process a source directory. <sup>(4)</sup> If dir2 exists, the result is a move. If dir2 doesn't exist, the result is a rename. <sup>(5)</sup> The last argument must be a directory. <sup>(6)</sup> Use caution with "force" option; you will not be prompted to confirm your action. <sup>(7)</sup> Use caution with "create parent" option; typing errors are not caught.	

### Create directories

The **mkdir** command creates one or more directories or subdirectories, generating errors if the file name already exists or when attempting to create a directory in a parent directory that doesn't exist. The **-p parent** option creates missing parent directories for the requested destination. Be cautious when using **mkdir -p**, since accidental spelling mistakes create unintended directories without generating error messages.

```
[student@desktopX ~]$ mkdir Video/Watched
mkdir: cannot create directory 'Video/Watched': No such file or directory
```

The **mkdir** failed because **Videos** was misspelled. "**Video**" does not exist as a location in which to create the **Watched** subdirectory. If a **-p** were used, the user would not have received an error message and now have two directories, **Video** and **Videos**.

```
[student@desktopX ~]$ mkdir Videos/Watched
```

```
[student@desktopX ~]$ cd Documents
[student@desktopX Documents]$ mkdir ProjectX ProjectY
[student@desktopX Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapter2 Thesis/Chapter3
[student@desktopX Documents]$ cd
[student@desktopX ~]$ ls -R Videos Documents
Documents:
ProjectX ProjectY Thesis thesis_chapter1.odf thesis_chapter2.odf

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:

[student@desktopX ~]$
```

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

### Copy files

The **cp** command copies one or more files to become new, independent files. Syntax allows copying an existing file to a new file in the current or another directory, or copying multiple files into another directory. In any destination, new file names must be unique. If the new file name is not unique, the copy command will overwrite the existing file.

```
[student@desktopX ~]$ cd Videos
[student@desktopX Videos]$ cp blockbuster1.ogg blockbuster3.ogg
[student@desktopX Videos]$ ls -l
total 0
-rw-rw-r--. 1 student student 0 Feb 8 16:23 blockbuster1.ogg
-rw-rw-r--. 1 student student 0 Feb 8 16:24 blockbuster2.ogg
-rw-rw-r--. 1 student student 0 Feb 8 19:02 blockbuster3.ogg
drwxrwxr-x. 2 student student 4096 Feb 8 23:35 Watched
[student@desktopX 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. Conflicting file names that exist at a destination may be overwritten. To protect users from accidentally overwriting directories with contents, multiple file **cp** commands ignore directories specified as a source. Copying non-empty directories, with contents, requires the **-r recursive** option.

```
[student@desktopX Videos]$ cd ../Documents
[student@desktopX Documents]$ cp thesis_chapter1.odf thesis_chapter2.odf Thesis ProjectX
cp: omitting directory `Thesis'
[student@desktopX Documents]$ cp -r Thesis ProjectX
[student@desktopX Documents]$ cp thesis_chapter2.odf Thesis/Chapter2/
[student@desktopX Documents]$ ls -R
```

```

.:
ProjectX ProjectY Thesis thesis_chapter1.odf thesis_chapter2.odf

./ProjectX:
Thesis thesis_chapter1.odf thesis_chapter2.odf

./ProjectX/Thesis:

./ProjectY:

./Thesis:
Chapter1 Chapter2 Chapter3

./Thesis/Chapter1:

./Thesis/Chapter2:
thesis_chapter2.odf

./Thesis/Chapter3:
[student@desktopX Documents]$

```

In the first **cp** command, **Thesis** failed to copy, but **thesis\_chapter1.odf** and **thesis\_chapter2.odf** succeeded. Using the **-r recursive** option, copying **Thesis** succeeded.

### Move files

The **mv** command renames files in the same directory, or relocates files to a new directory. File contents remain unchanged. Files moved to a different file system require creating a new file by copying the source file, then deleting the source file. Although normally transparent to the user, large files may take noticeably longer to move.

```

[student@desktopX Videos]$ cd ../Documents
[student@desktopX Documents]$ ls -l
total 0
-rw-rw-r--. 1 student student 0 Feb  8 16:24 thesis_chapter1.odf
-rw-rw-r--. 1 student student 0 Feb  8 16:24 thesis_chapter2.odf
[student@desktopX Documents]$ mv thesis_chapter2.odf thesis_chapter2_reviewed.odf
[student@desktopX Documents]$ mv thesis_chapter1.odf Thesis/Chapter1
[student@desktopX Documents]$ ls -lR
.:
total 16
drwxrwxr-x. 2 student student 4096 Feb 11 11:58 ProjectX
drwxrwxr-x. 2 student student 4096 Feb 11 11:55 ProjectY
drwxrwxr-x. 5 student student 4096 Feb 11 11:56 Thesis
-rw-rw-r--. 1 student student 0 Feb 11 11:54 thesis_chapter2_reviewed.odf

./ProjectX:
total 0
-rw-rw-r--. 1 student student 0 Feb 11 11:58 thesis_chapter1.odf
-rw-rw-r--. 1 student student 0 Feb 11 11:58 thesis_chapter2.odf

./ProjectX/Thesis:
total 0

./ProjectY:
total 0

./Thesis:
total 12
drwxrwxr-x. 2 student student 4096 Feb 11 11:59 Chapter1
drwxrwxr-x. 2 student student 4096 Feb 11 11:56 Chapter2
drwxrwxr-x. 2 student student 4096 Feb 11 11:56 Chapter3

```

```

./Thesis/Chapter1:
total 0
-rw-rw-r--. 1 student student 0 Feb 11 11:54 thesis_chapter1.odf

./Thesis/Chapter2:
total 0
-rw-rw-r--. 1 student student 0 Feb 11 11:54 thesis_chapter2.odf

./Thesis/Chapter3:
total 0
[student@desktopX Documents]$

```

The first **mv** command is an example of renaming a file. The second causes the file to be relocated to another directory.

### Remove files and directories

Default syntax for **rm** deletes files, but not directories. Deleting a directory, and potentially many subdirectories and files below it, requires the **-r recursive** option. There is no command-line undelete feature, nor a trash bin from which to restore.

```

[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ rm thesis_chapter2_reviewed.odf
[student@desktopX Documents]$ rm Thesis/Chapter1
rm: cannot remove `Thesis/Chapter1': Is a directory
[student@desktopX Documents]$ rm -r Thesis/Chapter1
[student@desktopX Documents]$ ls -l Thesis
total 8
drwxrwxr-x. 2 student student 4096 Feb 11 12:47 Chapter2
drwxrwxr-x. 2 student student 4096 Feb 11 12:48 Chapter3
[student@desktopX 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
[student@desktopX Documents]$

```

After **rm** failed to delete the **Chapter1** directory, the **-r recursive** option succeeded. The last **rm** command parsed into each subdirectory first, individually deleting contained files before removing each now-empty directory. Using **-i** will interactively prompt for each deletion. This is essentially the opposite of **-f** which will force the deletion without prompting the user.

The **rmdir** command deletes directories only if empty. Removed directories cannot be undeleted.

```

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

```

The **rmdir** command failed to delete non-empty **ProjectX**, but **rm -r** succeeded.



## References

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



# Practice: Command-Line File Management

## Guided exercise

In this lab, you will practice efficient techniques for creating and organizing files using directories, file copies, and links.

### Outcomes:

Students will practice creating, rearranging, and deleting files.

### Before you begin...

Log into your student account on serverX. Begin in your home directory.

- 1. In your home directory, create sets of empty practice files to use for the remainder of this lab. If the intended command is not immediately recognized, students are expected to use the guided solution to see and practice how the task is accomplished. Use the shell tab completion to locate and complete path names more easily.

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@serverX ~]$ touch song1.mp3 song2.mp3 song3.mp3 song4.mp3 song5.mp3
song6.mp3
[student@serverX ~]$ touch snap1.jpg snap2.jpg snap3.jpg snap4.jpg snap5.jpg
snap6.jpg
[student@serverX ~]$ touch film1.avi film2.avi film3.avi film4.avi film5.avi
film6.avi
[student@serverX ~]$ ls -l
```

- 2. From your home directory, move the song files into your **Music** subdirectory, the snapshot files into your **Pictures** subdirectory, and the movie files into your **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@serverX ~]$ mv song1.mp3 song2.mp3 song3.mp3 song4.mp3 song5.mp3
song6.mp3 Music
[student@serverX ~]$ mv snap1.jpg snap2.jpg snap3.jpg snap4.jpg snap5.jpg
snap6.jpg Pictures
[student@serverX ~]$ mv film1.avi film2.avi film3.avi film4.avi film5.avi
film6.avi Videos
[student@serverX ~]$ ls -l Music Pictures Videos
```

- 3. In your home directory, create three subdirectories for organizing your files into projects. Call these directories **friends**, **family**, and **work**. Create all three with one command.

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

```
[student@serverX ~]$ mkdir friends family work
[student@serverX ~]$ ls -l
```

- ❑ 4. You will collect some of the new files into the project directories for 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 into this directory. You are making copies, since you will keep the originals after giving these projects to family and friends.

Copy files (all types) containing numbers 1 and 2 to the friends folder.

Copy files (all types) containing numbers 3 and 4 to the family folder.

When collecting files from multiple locations into one location, change to the directory that will contain the *destination* files. Use the simplest path syntax, absolute or relative, to reach the source for each file management task.

```
[student@serverX ~]$ cd friends
[student@serverX friends]$ cp ~/Music/song1.mp3 ~/Music/song2.mp3 ~/Pictures/
snap1.jpg ~/Pictures/snap2.jpg ~/Videos/film1.avi ~/Videos/film2.avi .
[student@serverX friends]$ ls -l
[student@serverX friends]$ cd ../family
[student@serverX family]$ cp ~/Music/song3.mp3 ~/Music/song4.mp3 ~/Pictures/
snap3.jpg ~/Pictures/snap4.jpg ~/Videos/film3.avi ~/Videos/film4.avi .
[student@serverX family]$ ls -l
```

- ❑ 5. For your work project, you will create linked files instead of copies, since these files will remain on this system, not given away to others. Look for the *link count* in the directory listing.

Link files (all types) containing numbers 5 and 6 to the work folder.

Link files look and behave like normal files, but do not use any additional disk space. Creating project directories using linked files organizes files as needed while the files remain stored in their original location. Remember that basic **ln** syntax matches **cp**. Copy or link the *existing* file to *create* a new file.

```
[student@serverX family]$ cd ../work
[student@serverX work]$ ln ~/Music/song5.mp3 ~/Music/song6.mp3 ~/Pictures/
snap5.jpg ~/Pictures/snap6.jpg ~/Videos/film5.avi ~/Videos/film6.avi .
[student@serverX work]$ ls -l
```

- ❑ 6. Your projects are now done. Time to clean up the projects.

Change to your home directory. Attempt to delete both the family and friends projects with a single **rmdir** command.

```
[student@serverX work]$ cd
[student@serverX ~]$ 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 since both directories are non-empty.

- 7. Use another command that will succeed in deleting both the family and friends folders.

```
[student@serverX ~]$ rm -r family friends  
[student@serverX ~]$ ls -l
```

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

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

- 9. Finally, from your home directory, use the **rmdir** command to delete the work directory. The command should succeed now that it is empty.

```
[student@serverX work]$ cd  
[student@serverX ~]$ rmdir work  
[student@serverX ~]$ ls -l
```

# Making Links Between Files

## Objectives

After completing this section, students should be able to use hard links and soft links to make multiple names point to the same file.

## Managing links between files

### Creating hard links

A hard link is a new directory entry with a reference to an existing file on the file system. Every file in a file system has one hard link by default. To save space, instead of copying, a new hard link can be created to reference the same file. A new hard link either needs to have a different file name, if it is created in the same directory as the existing hard link, or it needs to reside in a different directory. All hard links pointing to the same file have the same permissions, link count, user/group ownerships, time stamps, and file content. Hard links pointing to the same file content need to be on the same file system.

The **ls -l** shows the hard link count after the permissions and before the owner of a file.

```
[root@serverX ~]# echo "Hello World" > newfile.txt
[root@serverX ~]# ls -l newfile.txt
-rw-r--r--. 1 root root 0 Mar 11 19:19 newfile.txt
```

The command **ln** creates new hard links to existing files. The command expects an existing file as the first argument, followed by one or more additional hard links. The hard links can reside anywhere as long as they are on the same file system as the existing file. After a new hard link is created, there is no way to tell which of the existing hard links is the original one.

Create a hard link **newfile-link2.txt** for the existing file **newfile.txt** in the same directory.

```
[root@serverX ~]# ln newfile.txt /tmp/newfile-hlink2.txt
[root@serverX ~]# ls -l newfile.txt /tmp/newfile-hlink2.txt
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile.txt
-rw-rw-r--. 2 root root 12 Mar 11 19:19 newfile-hlink2.txt
```

Even if the original file gets deleted, the content of the file is still available as long as at least one hard link exists.

```
[root@serverX ~]# rm -f newfile.txt
[root@serverX ~]# ls -l /tmp/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 /tmp/newfile-link2.txt
[root@serverX ~]# cat /tmp/newfile-link2.txt
Hello World
```



## Important

All hard links referencing the same file have the same permissions, link count, user/group ownerships, time stamps, and file content. If any of that information is changed on one hard link, all other hard links pointing at the same file will show the new information as well.

### Creating soft links

The **ln -s** command creates a soft link, also known as a symbolic link. A soft link is a special file type that points to an existing file or directory. Soft links can point to a file or directory on another file system. Unlike a hard link, a symbolic link can point to a file on a different file system.

```
[root@serverX ~]# ln -s /root/newfile-link2.txt /tmp/newfile-symlink.txt
[root@serverX ~]# ls -l newfile-link2.txt /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 root root 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /root/newfile-link2.txt
-rw-rw-r--. 1 root root 12 Mar 11 19:19 newfile-link2.txt
```

When the original file gets deleted, the soft link is still pointing to the file but the target is gone. A soft link pointing to a missing file is called a "dangling soft link."

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

A soft link can point to a directory. The soft link then acts like a directory. Changing to the soft link directory with **cd** works as expected.

Create a soft link **/root/configfiles** pointing to the **/etc** directory.

```
[root@serverX ~]# ln -s /etc /root/configfiles
[root@serverX ~]# cd /root/configfiles
[root@serverX configfiles]# pwd
/root/configfiles
```



## References

**ln(1)** man page

## Practice: Making Links Between Files

### Guided exercise

In this lab, you will create hard and soft links.

#### Outcomes:

The user creates a hard link and a soft link.

- ❑ 1. Create an additional hard link **/root/qmp-manual.txt** for the existing file **/usr/share/doc/qemu-kvm/qmp-commands.txt** on serverX.
- ❑ 1.1. Create the hard link **/root/qmp-manual.txt**. Link it to the file **/usr/share/doc/qemu-kvm/qmp-commands.txt**.

```
[root@serverX ~]# ln /usr/share/doc/qemu-kvm/qmp-commands.txt /root/qmp-manual.txt
```

- ❑ 1.2. Verify the link count on the newly created link **/root/qmp-manual.txt**.

```
[root@serverX ~]# ls -l /root/qmp-manual.txt
-rw-r--r--. 2 root root 63889 Nov 11 02:58 /root/qmp-manual.txt
```

- ❑ 1.3. Verify the link count on the original file **/usr/share/doc/qemu-kvm/qmp-commands.txt**.

```
[root@serverX ~]# ls -l /usr/share/doc/qemu-kvm/qmp-commands.txt
-rw-r--r--. 2 root root 63889 Nov 11 02:58 /usr/share/doc/qemu-kvm/qmp-commands.txt
```

- ❑ 2. Create the soft link **/root/temppdir** pointing to the directory **/tmp** on serverX.
- ❑ 2.1. Create the soft link **/root/temppdir**. Link it to **/tmp**.

```
[root@serverX ~]# ln -s /tmp /root/temppdir
```

- ❑ 2.2. Verify the newly created link with **ls -l**.

```
[root@serverX ~]# ls -l /root
lrwxrwxrwx. 1 root root 4 Mar 13 08:42 /root/temppdir -> /tmp
```

## Summary

### The Linux File System Hierarchy

Identify the purpose for top-level directories in the Linux hierarchy.

### Managing Files Using Command-Line Tools

Work from the command line to create, move, and delete files and directories.

### Making Links Between Files

Handling links to existing files can save space on the file system.







## CHAPTER 3

# USERS AND GROUPS

Overview	
<b>Goal</b>	To manage Linux users and groups and administer local password policies.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Explain the role of users and groups on a Linux system and how they are understood by the computer.</li><li>• Run commands as the superuser to administer a Linux system.</li><li>• Create, modify, lock, and delete locally defined user accounts.</li><li>• Create, modify, and delete locally defined group accounts.</li><li>• Lock accounts manually or by setting a password-aging policy in the shadow password file.</li><li>• Use centralized identity management services.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Users and Groups (and Practice)</li><li>• Gaining Superuser Access (and Practice)</li><li>• Managing Local User Accounts (and Practice)</li><li>• Managing Local Group Accounts (and Practice)</li><li>• Managing User Passwords (and Practice)</li><li>• Using Identity Management Services (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Managing Local Linux Users and Groups</li></ul>

# Users and Groups

## Objectives

After completing this section, students should be able to explain the role of users and groups on a Linux system and how they are understood by the computer.

## What is a user?

Every process (running program) on the system runs as a particular user. Every file is owned by a particular user. Access to files and directories are restricted by user. The user associated with a running process determines the files and directories accessible to that process.

The **id** command is used to show information about the current logged-in user. Basic information about another user can also be requested by passing in the username of that user as the first argument to the **id** command.

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

To view the user associated with a file or directory, use the **ls -l** command. The third column shows the username:

```
[student@serverX ~]$ ls -l /tmp
drwx-----. 2 gdm      gdm      4096 Jan 24 13:05 orbit-gdm
drwx-----. 2 student student 4096 Jan 25 20:40 orbit-student
-rw-r--r--. 1 root     root     23574 Jan 24 13:05 postconf
```

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. The first column shows the username:

```
[student@serverX ~]$ ps au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        428  0.0  0.7 152768 14400 tty1     Ss+  Feb03   0:04 /usr/bin/Xorg
root        511  0.0  0.0 110012    812 ttyS0    Ss+  Feb03   0:00 /sbin/agetty
root       1805  0.0  0.1 116040  2580 pts/0    Ss   Feb03   0:00 -bash
root       2109  0.0  0.1 178468  2200 pts/0    S    Feb03   0:00 su - student
student    2110  0.0  0.1 116168  2864 pts/0    S    Feb03   0:00 -bash
student    3690  0.0  0.0 123368  1300 pts/0    R+   11:42   0:00 ps au
```

The output of the previous commands displays users by name, but internally the operating system tracks users by a **UID number**. The mapping of names to numbers is defined in databases of account information. By default, systems use a simple "flat file," the **/etc/passwd** file, to store information about local users. The format of **/etc/passwd** follows (seven colon-separated fields):

```
①username:②password:③UID:④GID:⑤GECOS:⑥/home/dir:⑦shell
```

- ① *username* is a mapping of a UID to a name for the benefit of human users.

- ② *password* is where, historically, passwords were kept in an encrypted format. Today, they are stored in a separate file called **/etc/shadow**.
- ③ *UID* is a user ID, a number that identifies the user at the most fundamental level.
- ④ *GID* is the user's primary group ID number. Groups will be discussed in a moment.
- ⑤ *GECOS* field is arbitrary text, which usually includes the user's real name.
- ⑥ */home/dir* is the location of the user's personal data and configuration files.
- ⑦ *shell* is a program that runs as the user logs in. For a regular user, this is normally the program that provides the user's command line prompt.

## What is a group?

Like users, groups have a name and a number (GID). Local groups are defined in **/etc/group**.

### Primary groups

- Every user has exactly one *primary group*.
- For local users, the primary group is defined by the GID number of the group listed in the third field of **/etc/passwd**.
- Normally, the primary group owns new files created by the user.
- Normally, the primary group of a newly created user is a newly created group with the same name as the user. The user is the only member of this *User Private Group* (UPG).

### Supplementary groups

- Users may be a member of zero or more *supplementary groups*.
- The users that are supplementary members of local groups are listed in the last field of the group's entry in **/etc/group**. For local groups, user membership is determined by a comma-separated list of users found in the last field of the group's entry in **/etc/group**:

```
groupname:password:GID:list,of,users,in,this,group
```

- Supplementary group membership is used to help ensure that users have access permissions to files and other resources on the system.



## References

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

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

- Section 29: Users and groups

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

## Practice: User and Group Concepts

### Quiz

Match the items below to their counterparts in the table.

<b>/etc/group</b>	<b>/etc/passwd</b>	GID	UID
home directory	login shell	primary group	

Description	Keyword
A number that identifies the user at the most fundamental level	
The program that provides the user's command line prompt	
Location of local group information	
Location of the user's personal files	
A number that identifies the group at the most fundamental level	
Location of local user account information	
The fourth field of <b>/etc/passwd</b>	

## Solution

Match the items below to their counterparts in the table.

Description	Keyword
A number that identifies the user at the most fundamental level	UID
The program that provides the user's command line prompt	login shell
Location of local group information	<b>/etc/group</b>
Location of the user's personal files	home directory
A number that identifies the group at the most fundamental level	GID
Location of local user account information	<b>/etc/passwd</b>
The fourth field of <b>/etc/passwd</b>	primary group

# Gaining Superuser Access

## Objectives

After completing this section, students should be able to run commands as the superuser to administer a Linux system.

## The root user

Most operating systems have some sort of *superuser*, a user that has all power over the system. This user in Red Hat Enterprise Linux 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. In order to perform tasks such as installing or removing software and to manage system files and directories, a user must escalate privileges to the **root** user.

Most devices can only be controlled by **root**, but there are a few exceptions. For instance, removable devices, such as USB devices, are allowed to be controlled by a normal user. Thus, a non-root user is allowed to add and remove files and otherwise manage a removable device, but only root is allowed to manage "fixed" hard drives by default.

This unlimited privilege, however, comes with responsibility. **root** has unlimited power to damage the system: remove files and directories, remove user accounts, add backdoors, etc. If the **root** account is compromised, someone else would have administrative control of the system. Throughout this course, administrators will be 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 Windows. In Linux, most system administrators log into an unprivileged user account and use various tools to temporarily gain root privileges.



### Warning

One common practice on Windows in the past is for the local Administrator user to log in directly to perform system administrator duties. However, on Linux, it is recommended that system administrators *should not* log in directly as **root**. Instead, system administrators should log in as a non-root user, and use other mechanisms (**su**, **sudo**, or **PolicyKit**, for example) to temporarily gain superuser privileges.

By logging in as the administrative user, 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.

In recent versions of Microsoft Windows, Administrator disabled by default, and features such as User Account Control (UAC) are used to limit administrative privileges for users until actually needed. In Linux, the **PolicyKit** system is the nearest equivalent to UAC.

## Switching users with **su**

The **su** command allows a user to switch to a different user account. If a username is not specified, the *root* account is implied. When invoked as a regular user, a prompt will display asking for the password of the account you are switching to; when invoked as *root*, there is no need to enter the account password.

**su [-] <username>**

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]#
```

The command **su username** starts a *non-login shell*, while the command **su - username** starts a *login shell*. The main distinction is **su -** sets up the shell environment as if this were a clean login as that user, while **su** just starts a shell as that user with the current environment settings.

In most cases, administrators want to run **su -** to get the user's normal 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. See **info su** for details.

## Running commands as root with **sudo**

Fundamentally, Linux implements a very coarse-grained permissions model: *root* can do everything, other users can do nothing (systems-related). The common solution previously discussed is to allow standard users to temporarily “become *root*” using the **su** command. The disadvantage is that while acting as *root*, all the privileges (and responsibilities) of *root* are granted. Not only can the user restart the web server, but they can also remove the entire **/etc** directory. Additionally, all users requiring superuser privilege in this manner must know the *root* password.

The **sudo** command allows a user to be permitted to run a command as *root*, or as another user, based on settings in the **/etc/sudoers** file. Unlike other tools such as **su**, **sudo** requires users to enter their own password for authentication, not the password of the account they are trying to access. This allows an administrator to hand out fine-grained permissions to users to delegate system administration tasks, without having to hand out the *root* password.

For example, when **sudo** has been configured to allow the user *student* to run the command **usermod** as *root*, *student* could run the following command to lock a user account:

```
[student@serverX ~]$ sudo usermod -L username
[sudo] password for student: password
```

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

```
[student@serverX ~]$ sudo tail /var/log/secure
...
Feb 19 15:23:36 localhost sudo: student : TTY=pts/0 ; PWD=/home/student ; USER=root ;
COMMAND=/sbin/usermod -L student
Feb 19 15:23:36 localhost usermod[16325]: lock user 'student' password
Feb 19 15:23:47 localhost sudo: student : TTY=pts/0 ; PWD=/home/student ; USER=root ;
COMMAND=/bin/tail /var/log/secure
```

In Red Hat Enterprise Linux 7, all members of group **wheel** can use **sudo** to run commands as any user, including **root**. The user will be prompted for their own password. This is a change from Red Hat Enterprise Linux 6 and earlier. Users who were members of group **wheel** did not get this administrative access by default in RHEL 6 and earlier.

To enable similar behavior on earlier versions of Red Hat Enterprise Linux, use **visudo** to edit the configuration file and uncomment the line allowing the group **wheel** to run all commands.

```
[root@desktopX ~]# cat /etc/sudoers
...Output omitted...
## Allows people in group wheel to run all commands
%wheel    ALL=(ALL)    ALL

## Same thing without a password
# %wheel  ALL=(ALL)    NOPASSWD: ALL
...Output omitted...
```



## Warning

RHEL 6 did not grant group **wheel** any special privileges by default. Sites which have been using this group may be surprised when RHEL 7 automatically grants all members of **wheel** full **sudo** privileges. This could lead to unauthorized users getting superuser access to RHEL 7 systems.

Historically, membership in group **wheel** has been used by Unix-like systems to grant or control superuser access.

Most system administration applications with a GUI use **PolicyKit** to prompt users for authentication and to manage root access. In Red Hat Enterprise Linux 7, **PolicyKit** may also prompt members of group **wheel** for their own password in order to get **root** privileges when using graphical tools. This is similar to the way in which they can use **sudo** to get those privileges at the shell prompt. **PolicyKit** grants these privileges based on its own configuration settings, separate from **sudo**. Advanced students may be interested in the **pkexec(1)** and **polkit(8)** man pages for details on how this system works, but it is beyond the scope of this course.





## References

**su**(1) and **sudo**(8) man pages

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

- Section 29.2: The Persona of a Process

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

## Practice: Running Commands as root

### Guided exercise

In this lab, you will practice running commands as **root**.

#### Outcomes

Use the **su** with and without login scripts to switch users. Use **sudo** to run commands with privilege.

#### *Before you begin...*

Reset your serverX system.

- ☐ 1. Log into the GNOME desktop on serverX as **student** with a password of **student**.
- ☐ 2. Open a window with a Bash prompt.  
Select **Applications > Utilities > Terminal**.
- ☐ 3. Explore characteristics of the current student login environment.
  - ☐ 3.1. View the user and group information and display the current working directory.

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

- ☐ 3.2. View the variables which specify the home directory and the locations searched for executable files.

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

- ☐ 4. Switch to root without the dash and explore characteristics of the new environment.
  - ☐ 4.1. Become the **root** user at the shell prompt.

```
[student@serverX ~]$ su
Password: redhat
```

- ☐ 4.2. View the user and group information and display the current working directory. Note the identity changed, but not the current working directory.

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

```
/home/student
```

- ❑ 4.3. View the variables which specify the home directory and the locations searched for executable files. Look for references to the **student** and **root** accounts.

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

- ❑ 4.4. Exit the shell to return to the **student** user.

```
[root@serverX student]# exit
exit
```

- ❑ 5. Switch to root with the dash and explore characteristics of the new environment.
  - ❑ 5.1. Become the **root** user at the shell prompt. Be sure all the login scripts are also executed.

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

- ❑ 5.2. View the user and group information and display the current working directory.

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

- ❑ 5.3. View the variables which specify the home directory and the locations searched for executable files. Look for references to the **student** and **root** accounts.

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

- ❑ 5.4. Exit the shell to return to the **student** user.

```
[root@serverX ~]# exit
logout
```

- ❑ 6. Run several commands as student which require root access.
  - ❑ 6.1. View the last 5 lines of the **/var/log/messages**.

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

```
[student@serverX ~]$ sudo tail -5 /var/log/messages
Feb  3 15:07:22 localhost su: (to root) root on pts/0
Feb  3 15:10:01 localhost systemd: Starting Session 31 of user root.
Feb  3 15:10:01 localhost systemd: Started Session 31 of user root.
Feb  3 15:12:05 localhost su: (to root) root on pts/0
Feb  3 15:14:47 localhost su: (to student) root on pts/0
```

- ❑ 6.2. Make a backup of a configuration file in the **/etc** directory.

```
[student@serverX ~]$ cp /etc/motd /etc/motdOLD
cp: cannot create regular file '/etc/motdOLD': Permission denied
[student@serverX ~]$ sudo cp /etc/motd /etc/motdOLD
```

- ❑ 6.3. Remove the **/etc/motdOLD** file that was just created.

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

- ❑ 6.4. Edit a configuration file in the **/etc** directory.

```
[student@serverX ~]$ echo "Welcome to class" >> /etc/motd
-bash: /etc/motd: Permission denied
[student@serverX ~]$ sudo vim /etc/motd
```

# Managing Local User Accounts

## Objectives

After completing this section, students should be able to create, modify, lock, and delete locally defined user accounts.

## Managing local users

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

### useradd creates users

- **useradd *username*** sets reasonable defaults for all fields in **/etc/passwd** when run without options. The **useradd** command does not set any valid password by default, and the user cannot log in until a password is set.
- **useradd --help** will display 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 will not have an effect on any existing users.

### usermod modifies existing users

- **usermod --help** will display the basic options that can be used to modify an account. Some common options include:

usermod options:	
<b>-c, --comment COMMENT</b>	Add a value, such as a full name, to the GECOS field.
<b>-g, --gid GROUP</b>	Specify the primary group for the user account.
<b>-G, --groups GROUPS</b>	Specify a list of supplementary groups for the user account.
<b>-a, --append</b>	Used with the <b>-G</b> option to append the user to the supplemental groups mentioned without removing the user from other groups.
<b>-d, --home HOME_DIR</b>	Specify a new home directory for the user account.
<b>-m, --move-home</b>	Move a user home directory to a new location. Must be used with the <b>-d</b> option.
<b>-s, --shell SHELL</b>	Specify a new login shell for the user account.
<b>-L, --lock</b>	Lock a user account.
<b>-U, --unlock</b>	Unlock a user account.

### userdel deletes users

- **userdel *username*** removes the user from **/etc/passwd**, but leaves the home directory intact by default.
- **userdel -r *username*** removes the user and 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 user ID number. This can also happen when files created by a deleted user exist outside their home directory. This situation can lead to information leakage and other security issues.

The **useradd** command assigns new users the first free UID number above 1000 (unless one is explicitly specified with the **-u UID** option). This is how information leakage can occur: If the first free UID number had been previously assigned to a user account which has since been removed from the system, the old user's UID number 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@serverX ~]# useradd prince
[root@serverX ~]# ls -l /home
drwx----- 3 prince prince 74 Feb  4 15:22 prince
[root@serverX ~]# userdel prince
[root@serverX ~]# ls -l /home
drwx----- 3 1000 1000 74 Feb  4 15:22 prince
[root@serverX ~]# useradd bob
[root@serverX ~]# ls -l /home
drwx----- 3 bob bob 74 Feb  4 15:23 bob
drwx----- 3 bob bob 74 Feb  4 15:22 prince
```

Notice that **bob** now owns all files that **prince** once 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 find "unowned" files and directories by running: **find / -nouser -o -nogroup 2> /dev/null**.

## id displays user information

- **id** will display user information, including the user's UID number and group memberships.
- **id username** will display user information for *username*, including the user's UID number and group memberships.

## passwd sets passwords

- **passwd username** can be used to either set the user's initial password or change that user's password.
- The **root** user can set a password to any value. A message will be 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@serverX ~]# passwd student
Changing password for user student.
New password: redhat123
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary
word
Retype new password: redhat123
```

```
passwd: all authentication tokens updated successfully.
```

- A regular user must choose a password which is at least 8 characters in length and is 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 just the resources they need to function.
- *UID 1000+* is the range available for assignment to regular users.



### Note

Prior to Red Hat Enterprise Linux 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

# Practice: Creating Users Using Command-line Tools

## Guided exercise

In this lab, you will create a number of users on your serverX system, setting and recording an initial password for each user.

### Outcomes

A system with additional user accounts.

### *Before you begin...*

Reset your serverX system.

- ☐ 1. Log into the GNOME desktop on serverX as **student** with a password of **student**.
- ☐ 2. Open a window with a Bash prompt.  
Select **Applications > Utilities > Terminal**.
- ☐ 3. Become the **root** user at the shell prompt.

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

- ☐ 4. Add the user *juliet*.

```
[root@serverX ~]# useradd juliet
```

- ☐ 5. Confirm that *juliet* has been added by examining the **/etc/passwd** file.

```
[root@serverX ~]# tail -2 /etc/passwd  
tcpdump:x:72:72:::/sbin/nologin  
juliet:x:1001:1001::/home/juliet:/bin/bash
```

- ☐ 6. Use the **passwd** command to initialize *juliet*'s password.

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

- ☐ 7. Continue adding the remaining users in the steps below and set initial passwords.
  - ☐ 7.1. romeo

```
[root@serverX ~]# useradd romeo
```



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

## □ 7.2. hamlet

```
[root@serverX ~]# useradd hamlet
[root@serverX ~]# passwd hamlet
```

## □ 7.3. reba

```
[root@serverX ~]# useradd reba
[root@serverX ~]# passwd reba
```

## □ 7.4. dolly

```
[root@serverX ~]# useradd dolly
[root@serverX ~]# passwd dolly
```

## □ 7.5. elvis

```
[root@serverX ~]# useradd elvis
[root@serverX ~]# passwd elvis
```

# Managing Local Group Accounts

## Objectives

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

## Managing supplementary 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.

### groupadd creates groups

- **groupadd** *groupname* without options uses the next available GID from the range specified in the `/etc/login.defs` file.
- The **-g** *GID* option is used to specify a specific GID.

```
[student@serverX ~]$ sudo groupadd -g 5000 ateam
```



### Note

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

- The **-r** option will create a system group using a GID from the range of valid system GID numbers listed in the `/etc/login.defs` file.

```
[student@serverX ~]$ sudo groupadd -r appusers
```

### groupmod modifies existing groups

- The **groupmod** command is used to change a group name to a GID mapping. The **-n** option is used to specify a new name.

```
[student@serverX ~]$ sudo groupmod -n javaapp appusers
```

- The **-g** option is used to specify a new GID.

```
[student@serverX ~]$ sudo groupmod -g 6000 ateam
```

### groupdel deletes a group

- The **groupdel** command will remove a group.

```
[student@serverX ~]$ sudo groupdel javaapp
```

- A group may not be removed if it is the primary group of any existing user. As with **userdel**, check all file systems to ensure that no files remain owned by the group.

## **usermod** alters group membership

- The membership of a group is controlled with user management. Change a user's primary group with **usermod -g *groupname***.

```
[student@serverX ~]$ sudo usermod -g student student
```

- Add a user to a supplementary group with **usermod -aG *groupname username***.

```
[student@serverX ~]$ sudo usermod -aG wheel elvis
```



### Important

The use of the **-a** option makes **usermod** function in "append" mode. Without it, the user would be removed from *all other* supplementary groups.



### References

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

## Practice: Managing Groups Using Command-line Tools

### Guided exercise

In this lab, you will add users to newly created supplementary groups.

#### Outcomes

The **shakespeare** group consists of **juliet**, **romeo**, and **hamlet**. The **artists** group contains **reba**, **dolly**, and **elvis**.

#### Before you begin...

Perform the following steps on serverX unless directed otherwise.

- ☐ 1. Become the **root** user at the shell prompt.

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

- ☐ 2. Create a supplementary group called **shakespeare** with a group ID of **30000**.

```
[root@serverX ~]# groupadd -g 30000 shakespeare
```

- ☐ 3. Create a supplementary group called **artists**.

```
[root@serverX ~]# groupadd artists
```

- ☐ 4. Confirm that *shakespeare* and *artists* have been added by examining the **/etc/group** file.

```
[root@serverX ~]# tail -5 /etc/group  
reba:x:1004:  
dolly:x:1005:  
elvis:x:1006:  
shakespeare:x:30000:  
artists:x:30001:
```

- ☐ 5. Add the *juliet* user to the *shakespeare* group as a supplementary group.

```
[root@serverX ~]# usermod -G shakespeare juliet
```

- ☐ 6. Confirm that *juliet* has been added using the **id** command.

```
[root@serverX ~]# id juliet  
uid=1001(juliet) gid=1001(juliet) groups=1001(juliet),30000(shakespeare)
```

- ☐ 7. Continue adding the remaining users to groups as follows:

- 7.1. Add *romeo* and *hamlet* to the *shakespeare* group.

```
[root@serverX ~]# usermod -G shakespeare romeo
[root@serverX ~]# usermod -G shakespeare hamlet
```

- 7.2. Add *reba*, *dolly*, and *elvis* to the *artists* group.

```
[root@serverX ~]# usermod -G artists reba
[root@serverX ~]# usermod -G artists dolly
[root@serverX ~]# usermod -G artists elvis
```

- 7.3. Verify the supplemental group memberships by examining the **/etc/group** file.

```
[root@serverX ~]# tail -5 /etc/group
reba:x:1004:
dolly:x:1005:
elvis:x:1006:
shakespeare:x:30000: juliet, romeo, hamlet
artists:x:30001: reba, dolly, elvis
```

# Managing User Passwords

## Objectives

After completing this section, students should be able to lock accounts manually or by setting a password-aging policy in the shadow password file.

## Shadow passwords and password policy

In the distant past, 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, or "password hashes," were moved to the more secure `/etc/shadow` file. This new file also allowed password aging and expiration features to be implemented.

There are three pieces of information stored in a modern password hash:

**\$1\$gCjLa2/Z\$6Pu0EK0AzfCjxjv2hoLOB/**

1. **1**: The hashing algorithm. The number 1 indicates an MD5 hash. The number 6 appears when a SHA-512 hash is used.
2. **gCjLa2/Z**: The *salt* used to encrypt the hash. This is originally chosen at random. The salt and the unencrypted password are combined and encrypted to create the encrypted password hash. The use of a salt prevents two users with the same password from having identical entries in the `/etc/shadow` file.
3. **6Pu0EK0AzfCjxjv2hoLOB/**: The encrypted hash.

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 doesn't 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.



### Note

Red Hat Enterprise Linux 6 and 7 support two new strong password hashing algorithms, SHA-256 (algorithm **5**) and SHA-512 (algorithm **6**). Both the salt string and the encrypted hash are longer for these algorithms. The default algorithm used for password hashes can be changed by the **root** user by running the command **authconfig --passalgo** with one of the arguments **md5**, **sha256**, or **sha512**, as appropriate.

Red Hat Enterprise Linux 7 defaults to using SHA-512 encryption.

### `/etc/shadow` format

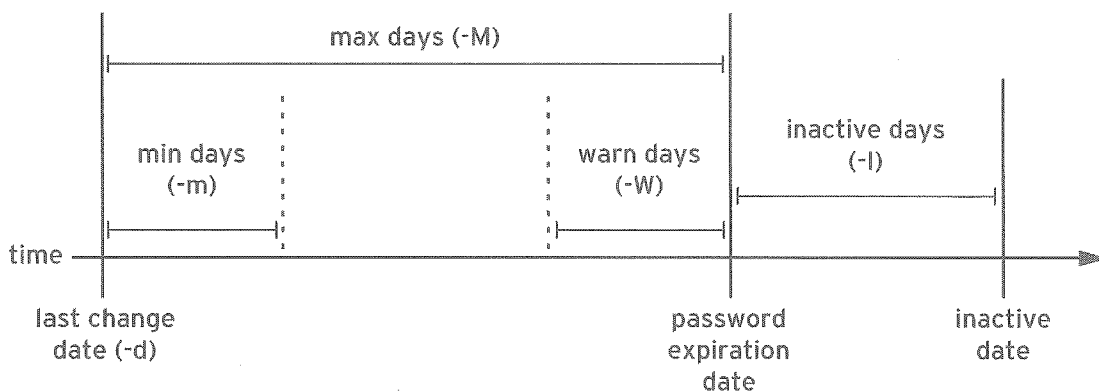
The format of `/etc/shadow` follows (nine colon-separated fields):

1	name:	2	password:	3	lastchange:	4	minage:	5	maxage:	6	warning:	7	inactive:	8	expire:	9	blank
---	-------	---	-----------	---	-------------	---	---------	---	---------	---	----------	---	-----------	---	---------	---	-------

- ❶ The login *name*. This must be a valid account name on the system.
- ❷ The encrypted *password*. A password field which starts with an exclamation mark means that the password is locked.
- ❸ The date of the last *password change*, represented as the number of days since 1970.01.01.
- ❹ The *minimum* number of days before a password may be changed, where 0 means "no minimum age requirement."
- ❺ The *maximum* number of days before a password must be changed.
- ❻ The *warning* period that a password is about to expire. Represented in days, where 0 means "no warning given."
- ❼ The number of days an account remains active after a password has expired. A user may still log into the system and change the password during this period. After the specified number of days, the account is locked, becoming *inactive*.
- ❽ The account *expiration* date, represented as the number of days since 1970.01.01.
- ❾ This *blank* field is reserved for future use.

## Password aging

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



```
# chage -m 0 -M 90 -W 7 -I 14 username
```

**chage -d 0 username** will force a password update on next login.

**chage -l username** will list a username's current settings.

**chage -E YYYY-MM-DD** will expire an account on a specific day.



### Note

The **date** command can be used to calculate a date in the future.

```
[student@serverX ~]$ date -d "+45 days"
Sat Mar 22 11:47:06 EDT 2014
```

## Restricting access

With the **chage** command, an account expiration can be set. Once that date is reached, the user cannot log into the system interactively. The **usermod** command can "lock" an account with the **-L** option.

```
[student@serverX ~]$ sudo usermod -L elvis
[student@serverX ~]$ su - elvis
Password: elvis
su: Authentication failure
```

When a user has left 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.

```
[student@serverX ~]$ sudo usermod -L -e 1 elvis
```

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 USERNAME**. If the account was also expired, be sure to also change the expiration date.

### The **nologin** shell

Sometimes a user needs an account with a password to authenticate to a system, but does not need an interactive shell on 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 into the system directly, the **nologin** "shell" will simply close the connection.

```
[root@serverX ~]# usermod -s /sbin/nologin student
[root@serverX ~]# su - student
Last login: Tue Feb  4 18:40:30 EST 2014 on pts/0
This account is currently not available.
```



### Important

Use of the **nologin** shell prevents interactive use of the system, but does not prevent all access. A user may still be able to authenticate and upload or retrieve files through applications such as web applications, file transfer programs, or mail readers.



### References

**chage**(1), **usermod**(8), **shadow**(5), **crypt**(3) man pages



# Practice: Managing User Password Aging

## Guided exercise

In this lab, you will set unique password policies for users.

### Outcomes

The password for *romeo* must be changed when the user first logs into the system, every 90 days thereafter, and the account expires in 180 days.

### Before you begin...

Perform the following steps on serverX unless directed otherwise.

- ❑ 1. Explore locking and unlocking accounts.

- ❑ 1.1. Lock the **romeo** account.

```
[student@serverX ~]$ sudo usermod -L romeo
```

- ❑ 1.2. Attempt to log in as **romeo**.

```
[student@serverX ~]$ su - romeo
Password: romeo
su: Authentication failure
```

- ❑ 1.3. Unlock the **romeo** account.

```
[student@serverX ~]$ sudo usermod -U romeo
```

- ❑ 2. Change the password policy for **romeo** to require a new password every 90 days.

```
[student@serverX ~]$ sudo chage -M 90 romeo
[student@serverX ~]$ sudo chage -l romeo
Last password change                : Feb 03, 2014
Password expires                    : May 04, 2014
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
```

- ❑ 3. Additionally, force a password change on the first login for the **romeo** account.

```
[student@serverX ~]$ sudo chage -d 0 romeo
```

- ❑ 4. Log in as **romeo** and change the password to **forsooth123**.

```
[student@serverX ~]$ su - romeo
Password: romeo
You are required to change your password immediately (root enforced)
```

```
Changing password for romeo.  
(current) UNIX password: romeo  
New password: forsooth123  
Retype new password: forsooth123  
[romeo@serverX ~]$ exit
```

□ 5. Expire accounts in the future.

□ 5.1. Determine a date 180 days in the future.

```
[student@serverX ~]$ date -d "+180 days"  
Sat Aug 2 17:05:20 EDT 2014
```

□ 5.2. Set accounts to expire on that date.

```
[student@serverX ~]$ sudo chage -E 2014-08-02 romeo  
[student@serverX ~]$ sudo chage -l romeo  
Last password change                : Feb 03, 2014  
Password expires                    : May 04, 2014  
Password inactive                   : never  
Account expires                    : Aug 02, 2014  
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
```

# Using Identity Management Services

## Objectives

After completing this section, students should be able to use centralized identity management services.

## User information and authentication services

### Need for centralized identity management

Modern computer infrastructures tend to consist of many machines, with multiple services running on them. Keeping local user accounts for all these machines and their services in sync is a daunting task, even more so when passwords need to remain synced.

A solution to this is to not store account information on local systems, but instead retrieve this information from a centralized store. Having user information, and the associated authentication information, centralized also allows for something called *Single Sign-On* (SSO). With SSO, a user authenticates once using a password (or other means), and then obtains a form of ticket or cookie that can be used to automatically authenticate to other services.

### User information and authentication

A centralized identity management system will need to provide at least two services:

1. *Account information:* This includes information such as a username, home directory location, UID and GID, group memberships, etc. Popular solutions include *LDAP* (Lightweight Directory Access Protocol), used in multiple products such as Active Directory and IPA Server, and *Network Information Services* (NIS).
2. *Authentication information:* A means for a system to validate that a user is who he/she claims to be. This can be done by providing a cryptographic password hash to the client system, or by sending the (encrypted) password to the server, and receiving a response. An LDAP server can provide authentication information in addition to account information. Kerberos only provides SSO authentication services, and is typically used alongside LDAP user information. Kerberos is used in both IPA Server and Active Directory.

On a Red Hat Enterprise Linux 7 system, local user information is provided by `/etc/passwd`, while authentication information (in the form of a hashed password) is provided by `/etc/shadow`.

## Attaching a system to centralized LDAP and Kerberos servers

### Authconfig

Configuring a Red Hat Enterprise Linux 7 system to use centralized identity management services requires the editing of various files, and the configuration of some daemons. For attaching to central LDAP and Kerberos servers, the following files, at a minimum, would need to be updated:

- `/etc/ldap.conf`: For information about the central LDAP server and its settings.

- **/etc/krb5.conf**: For information about the central Kerberos infrastructure.
- **/etc/sss/sss.conf**: To configure the *system security services daemon (sssd)*, the daemon responsible for retrieving and caching user information and authentication info.
- **/etc/nsswitch.conf**: To indicate to the system which user information and authentication services should be used.
- **/etc/pam.d/\***: Configuring how authentication should be handled for various services.
- **/etc/openldap/cacerts**: To store the root *certificate authorities* (CA) that can validate the SSL certificates used to identify LDAP servers.

The **sssd** daemon will need to be enabled and started before the system can use it.

With this number of files and services to configure, a mistake is easily made. Red Hat Enterprise Linux 7 ships with a suite of tools to automate these configurations: **authconfig**. **authconfig** consists of three related tools that can all perform the same actions:

- **authconfig**: A command-line tool. This tool can be used to automate configurations across a number of systems. The commands used with **authconfig** tend to be very long, with multiple options being passed in. This tool is installed using the *authconfig* package.
- **authconfig-tui**: The interactive version of **authconfig**. Uses a menu-driven text interface. Can be used over **ssh**. This tool is installed using the *authconfig* package.
- **authconfig-gtk**: This version launches a graphical interface. It can also be launched as **system-config-authentication**. This tool is installed using the **authconfig-gtk** package.

### Necessary LDAP parameters

To connect to a central LDAP server for user information, **authconfig** needs a number of settings:

- The host name of the LDAP server(s)
- The *base DN* (Distinguished Name) of the part of the LDAP tree where the system should look for users. This typically looks something like **dc=example, dc=com**, or **ou=People, o=PonyCorp**. This information will be provided by your LDAP server administrator.
- If SSL/TLS is used to encrypt communications with the LDAP server, a root CA certificate that can validate the certificates is offered by the LDAP server.

**Important:** A system will also need some extra packages installed to provide LDAP client functionality. Installing **sssd** will provide all the necessary dependencies.

### Necessary Kerberos parameters

To configure a system to use a centralized Kerberos system for user authentication, **authconfig** will need the following settings:

- The name of the Kerberos *realm* to use. A Kerberos realm is a domain of machines that all use a common set of Kerberos servers and users for authentication.
- One or more *key distribution centers* (KDC). This is the host name of your Kerberos server(s).

- The host name of one or more *admin servers*. This is the machine that clients will talk to when they want to change their password, or perform other user modifications. This is typically the same as the primary KDC, but it can be a different machine.

In addition, an administrator can specify if DNS should be used to look up the realm to use for a specific host name, and to automatically find the KDCs and admin servers. An extra package can be installed to help debug Kerberos issues, and to work with Kerberos tickets from the command line: *krb5-workstation*.

## Using authconfig-gtk

To use **authconfig-gtk** to configure a system for LDAP + Kerberos, use the following steps:

1. Install all the necessary packages:

```
[student@demo ~]$ sudo yum -y install authconfig-gtk sssd krb5-workstation
```

2. Launch **authconfig-gtk**, either from the command line or from **Applications > Sundry > Authentication**. Enter the **root** password when prompted.
3. On the **Identity & Authentication** tab, select **LDAP** from the **User Account Database** **drop-down**. Fill out the **LDAP Search Base DN** and **LDAP Server** fields.

**Authentication Configuration**

Identity & Authentication | Advanced Options | Password Options

**User Account Configuration**

User Account Database: LDAP

LDAP Search Base DN: dc=example,dc=com

LDAP Server: classroom.example.com

☒ Use TLS to encrypt connections

Download CA Certificate...

**Authentication Configuration**

Authentication Method: Kerberos password

Realm: EXAMPLE.COM

KDCs: classroom.example.com

Admin Servers: classroom.example.com

☐ Use DNS to resolve hosts to realms

☐ Use DNS to locate KDCs for realms

Revert | Cancel | Apply

*Figure 3.2: Main authconfig-gtk window*

4. If the LDAP server supports TLS, check the **Use TLS to encrypt connections** box, and use the **Download CA Certificate** button to download the CA certificate.
5. From the **Authentication Method** dropdown, select **Kerberos password**, and fill out the **Realm**, **KDCs**, and **Admin Servers** fields. The last two fields are not available if the **Use DNS to locate KDCs for realms** option is checked.
6. If central home directories are not available, users can create directories on first login by checking the **Create home directories on the first login** box on the **Advanced Options** tab.
7. Click the **Apply** button to save and activate the changes. This will write all relevant configuration files and (re)start the **sssd** service.

#### Testing a configuration

To test the LDAP + Kerberos configuration, an administrator can simply attempt to log into the system (over **ssh**) using the credentials of one of the network users. In addition, the **getent passwd <USERNAME>** command can be used to retrieve information about a network user, in the form **getent passwd <USERNAME>**.

**Important:** In the default configuration, **sssd** will *not* enumerate network users when no username is specified to the **getent** command. This is done to keep the graphical login screen uncluttered and to save valuable network resources and time.

## Attaching a System to an IPA Server

Red Hat provides an integrated solution for configuring LDAP and Kerberos: IPA (Identity, Policy, and Auditing) Server. IPA Server provides LDAP and Kerberos, combined with a suite of both command-line and web-based administration tools. Apart from user and authentication information, IPA Server can centralize **sudo** rules, SSH public keys, SSH host keys, TLS certificates, automounter maps, and much more.

#### Using ipa-client

A Red Hat Enterprise Linux 7 system can be configured to use an IPA server using the **authconfig** suite of tools, but a specialized tool also exists: **ipa-client-install**. This command can be installed from the *ipa-client* package, which pulls in all dependencies (such as **sssd**).

One of the benefits of using **ipa-client-install** is that it can retrieve almost all necessary information from DNS (when configured either by the IPA server or manually by an administrator), as well as create host entries and more on the IPA server. This allows an IPA server administrator to set access policies for that host, create service *principals* (e.g., for NFSv4 exports), and more.

When **ipa-client-install** is run without any arguments, it will first attempt to retrieve information about the IPA server configured for its DNS domain from DNS. If that fails, it will prompt the administrator for the necessary information, such as the domain name of the IPA server and the realm to use. Other information that needs to be provided are the username and password of an account that is allowed to create new machine entries on the IPA server. Unless another account has been created for this, the default IPA server administrator account (**admin**) can be used for this.

The following is an example of a (mostly) DNS driven configuration:

```
[student@desktop ~]$ sudo ipa-client-install
Discovery was successful!
Hostname: desktop.domain0.example.com
Realm: DOMAIN0.EXAMPLE.COM
DNS Domain: server.domain0.example.com
IPA Server: server.domain0.example.com
BasedN: dc=server,dc=domain0,dc=example,dc=com

Continue to configure the system with these values? [no]: yes
User authorized to enroll computers: admin
Synchronizing time with KDC...
Password for admin@DOMAIN0.EXAMPLE.COM: redhat123
Successfully retrieved CA cert
  Subject:      CN=Certificate Authority,O=DOMAIN0.EXAMPLE.COM
  Issuer:       CN=Certificate Authority,O=DOMAIN0.EXAMPLE.COM
  Valid From:   Thu Feb 27 13:31:04 2014 UTC
  Valid Until:  Mon Feb 27 13:31:04 2034 UTC

Enrolled in IPA realm DOMAIN0.EXAMPLE.COM
Created /etc/ipa/default.conf
New SSSD config will be created
Configured /etc/sss/sss.conf
Configured /etc/krb5.conf for IPA realm DOMAIN0.EXAMPLE.COM
Adding SSH public key from /etc/ssh/ssh_host_rsa_key.pub
Adding SSH public key from /etc/ssh/ssh_host_ecdsa_key.pub
SSSD enabled
Configured /etc/openldap/ldap.conf
Configured /etc/ssh/ssh_config
Configured /etc/ssh/sshd_config
Client configuration complete.
```

It is possible to specify all needed information as command-line arguments, allowing for unattended setups as part of an initial system configuration; for example, from a *kickstart*. See the manual page for **ipa-client-install(1)** for more information.

## Joining a system to Active Directory

Red Hat Enterprise Linux 7 features multiple methods of joining a system to Active Directory. Administrators can choose to install the *samba-winbind* package and configure **winbind** through the **authconfig** family of tools, or administrators can install both *sss* and *realmd* packages and use **sss** and the **realm** command.



### Note

The **realm** command can also be used to join Kerberos realms, or IPA server domains, but the final configuration will be slightly different; for example, users will have **@domain** appended to their usernames. **ipa-client-install** is the preferred method to join IPA domains.



### Note

Since there is no Active Directory server running in the classroom, there is no current possibility to practice these steps.

The following is an example of using **realmd** to join an Active Directory domain, and allow Active Directory users to log into the local system. This example assumes that the Active Directory domain is called **domain.example.com**.

1. Install the necessary packages: **realmd**.

```
[student@demo ~]$ yum -y install realmd
```

2. Discover the settings for the **domain.example.com** domain.

```
[student@demo ~]$ sudo realm discover domain.example.com
```

3. Join the Active Directory domain; this will install all necessary packages, and configure **sssd**, **pam**, **/etc/nsswitch.conf**, etc.

```
[student@demo ~]$ sudo realm join domain.example.com
```

This will attempt to join the local system to Active Directory using the **Administrator** account; enter the password for this account when prompted. To use a different account, use the **--user** argument.

4. Active Directory accounts are now usable on the local system, but logins using Active Directory are still disabled. To enable logins, use the following command:

```
[student@demo ~]$ sudo realm permit --realm domain.example.com --all
```

To only allow certain users to log in, replace **--all** with a list of those users. For example:

```
[student@demo ~]$ sudo realm permit --realm domain.example.com DOMAIN\Itchy DOMAIN\Scratchy
```



## Note

By default, domain users must use their fully qualified name to log in; e.g., **ipouser@ipa.example.com** for IPA users, or **DOMAIN\Picard** for Active Directory. To disable this, change the **use\_fully\_qualified\_names** setting in the correct domain block in **/etc/sss/sss.conf** to **False**, or remove it entirely, then restart the **sssd** service.



## References

**authconfig(8)**, **authconfig-tui(8)**, **authconfig-gtk(8)**, **sssd(8)**, **sssd-ipa(8)**, **sssd.conf(5)**, **sssd-ad**, and **realmd(8)** man pages



# Practice: Connecting to a Central LDAP and Kerberos Server

## Guided exercise

In this lab, you will connect your **desktopX** system to become a client of the LDAP server running on **classroom.example.com**. You will configure your **desktopX** system to use the Kerberos infrastructure provided by **classroom.example.com** for additional authentication.

Resources:	
<b>Files:</b>	<code>http://classroom.example.com/pub/example-ca.crt</code>
<b>Machines:</b>	<b>desktopX</b>

### Outcomes:

**desktopX** configured for LDAP user information and Kerberos authentication from **classroom.example.com**.

### Before you begin...

- Reset your **desktopX** system.

To simplify user management, your company has decided to switch to centralized user management. Another team has already set up all the required LDAP and Kerberos services. Centralized home directories are not yet available, so the system should be configured to create local home directories when a user first logs in.

Given the following information, configure your **desktopX** system to use user information from the LDAP server, and authentication services from the Kerberos KDC. DNS service records for the realm have not yet been configured, so you will have to configure Kerberos settings manually.

Name	Value
LDAP server	<code>ldap://classroom.example.com</code>
LDAP base DN	<code>dc=example,dc=com</code>
Use TLS	Yes
Root CA	<code>http://classroom.example.com/pub/example-ca.crt</code>
Kerberos realm	<code>EXAMPLE.COM</code>
Kerberos KDC	<code>classroom.example.com</code>
Kerberos admin server	<code>classroom.example.com</code>

1. Start by installing the necessary packages: `sssd`, `krb5-workstation`, and `authconfig-gtk`.

- 1.1.

```
[student@desktopX ~]$ sudo yum -y install sssd authconfig-gtk krb5-workstation
```

2. Launch the **Authentication Configuration** application, then apply the settings from the table for both LDAP and Kerberos options.

- ❑ 2.1. Either launch **system-config-authentication** from the command line, or launch **Applications > Sundry > Authentication**. Enter the **root** password (**redhat**) when asked.
- ❑ 2.2. Make sure the **Identity & Authentication** tab is open.
- ❑ 2.3. In the **User Account Database**, select **LDAP**.
- ❑ 2.4. Enter **dc=example,dc=com** in the **LDAP Search Base DN** field, and **classroom.example.com** in the **LDAP Server** field.
- ❑ 2.5. Make sure the **Use TLS to encrypt connections** box is checked, then click the **Download CA Certificate...** button.
- ❑ 2.6. Enter **http://classroom.example.com/pub/example-ca.crt** in the **Certificate URL** field, then click **OK**.
- ❑ 2.7. Select **Kerberos password** from the **Authentication Method** dropdown, and uncheck both **Use DNS...** boxes.
- ❑ 2.8. Enter **EXAMPLE.COM** in the **REALM** field, and **classroom.example.com** in both the **KDCs** and **Admin Servers** fields.
- ❑ 2.9. Switch to the **Advanced Options** tab and place a checkmark in the **Create home directories on the first login** box.
- ❑ 2.10. Click the **Apply** button to apply your changes.
- ❑ 3. Use both **getent** and **ssh** to verify your work. You can use the username **ldapuserX** (where **X** is your station number) with the password **kerberos**. Please note that your users will not yet have a home directory mounted.

❑ 3.1. 

```
[student@desktopX ~]$ getent passwd ldapuserX
ldapuserX:*:170X:170X:LDAP Test User X:/home/guests/ldapuserX:/bin/bash
```

❑ 3.2. 

```
[student@desktopX ~]$ ssh ldapuserX@localhost
The authenticity of host 'localhost (::1)' can't be established.
EDCSA key fingerprint is XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
ldapuserX@localhost's password: kerberos
Creating home directory for ldapuserX.
[ldapuserX@desktopX$ ]$ pwd
/home/guests/ldapuserX
[ldapuserX@desktopX$ ]$ ls -a
. .bash_history .bash_profile .cache .mozilla
.. .bash_logout .bashrc .config
[ldapuserX@desktopX$ ]$ logout
```

# Lab: Managing Linux Users and Groups

## Performance checklist

In this lab, you will work on **desktopX** to define a group of local user accounts and configure the machine to use network-defined user accounts. You will define a default password policy, create a supplementary group of three new users, and modify the password policy of one user.

### Outcomes

- A new group on desktopX called **consultants**, including three new user accounts for Sam Spade, Betty Boop, and Dick Tracy.
- All new local accounts should require that passwords be changed at first login and every 30 days thereafter.
- The new consultant accounts should expire at the end of the 90-day contract, and Betty Boop must change her password every 15 days.
- **desktopX** is a client to the IPA server running on serverX and can use the accounts defined in IPA to authenticate.

### Before you begin...

- Reset your **serverX** system.
- Log into your **serverX** system. Please note: this step will take approximately 15 minutes.

```
[student@serverX ~]$ lab ipaclient setup
```

- Reset your **desktopX** system while the setup on **serverX** is running.
- Wait for your serverX system to complete before continuing.

The relevant configuration information for the existing IPA server is:

Name	Value
Realm	<b>SERVERX.EXAMPLE.COM</b> , where <b>X</b> is your station number.
Domain	<b>serverX.example.com</b> , where <b>X</b> is your station number. Note that your <b>desktopX</b> machine is not a part of this DNS domain.
Administrative user	<b>admin</b>
Password	<b>redhat123</b>

A user has already been configured for you to test with. The username is **ipausers**, and the password is **password**. Due to the password policy, this password will need to be changed on first login. Change this password to **redhat123**.

Central home directories have not yet been configured, so for now, configure the system to automatically create a new local home directory when a user first logs in.

When you have completed your work, run **lab combined-users grade** on your **desktopX** machine to verify your work.

1. Ensure that newly created, local users have passwords which must be changed every 30 days.
2. Create a new group named **consultants** with a GID of 900.
3. Create three new users: **sspade**, **bboop**, and **dtracy**, with a password of **default** and add them to the supplementary group **consultants**. The primary group should remain as the user private group.
4. Determine the date 90 days in the future and set each of the three new user accounts to expire on that date.
5. Change the password policy for the **bboop** account to require a new password every 15 days.
6. Additionally, force all users to change their password on first login.
7. Install the *ipa-client* package on your **desktopX** machine.
8. Configure your system, using **ipa-client-install**, to use the IPA server setup for the **serverX.example.com** DNS domain. Home directories should automatically be created, and NTP should not be configured during this process.
9. Verify that you can now successfully log into **desktopX** as the user **ipausers** by using **ssh**. The initial password is **password**, but this should be changed to **redhat123**. Due to the password change requirement, you will have to log in twice.
10. When you finish, run the **lab combined-users grade** evaluation script to confirm you have done everything correctly.

## Solution

In this lab, you will work on **desktopX** to define a group of local user accounts and configure the machine to use network-defined user accounts. You will define a default password policy, create a supplementary group of three new users, and modify the password policy of one user.

### Outcomes

- A new group on desktopX called **consultants**, including three new user accounts for Sam Spade, Betty Boop, and Dick Tracy.
- All new local accounts should require that passwords be changed at first login and every 30 days thereafter.
- The new consultant accounts should expire at the end of the 90-day contract, and Betty Boop must change her password every 15 days.
- **desktopX** is a client to the IPA server running on serverX and can use the accounts defined in IPA to authenticate.

### Before you begin...

- Reset your **serverX** system.
- Log into your **serverX** system. Please note: this step will take approximately 15 minutes.

```
[student@serverX ~]$ lab ipaclient setup
```

- Reset your **desktopX** system while the setup on **serverX** is running.
- Wait for your serverX system to complete before continuing.

The relevant configuration information for the existing IPA server is:

Name	Value
Realm	<b>SERVERX.EXAMPLE.COM</b> , where <i>X</i> is your station number.
Domain	<b>serverX.example.com</b> , where <i>X</i> is your station number. Note that your <b>desktopX</b> machine is not a part of this DNS domain.
Administrative user	<b>admin</b>
Password	<b>redhat123</b>

A user has already been configured for you to test with. The username is **ipausers**, and the password is **password**. Due to the password policy, this password will need to be changed on first login. Change this password to **redhat123**.

Central home directories have not yet been configured, so for now, configure the system to automatically create a new local home directory when a user first logs in.

When you have completed your work, run **lab combined-users grade** on your **desktopX** machine to verify your work.

1. Ensure that newly created, local users have passwords which must be changed every 30 days.

```
[student@desktopX ~]$ sudo vi /etc/login.defs
[student@desktopX ~]$ cat /etc/login.defs
...Output omitted...
PASS_MAX_DAYS 30
PASS_MIN_DAYS 0
PASS_MIN_LEN 5
PASS_WARN_AGE 7
...Output omitted...
```

2. Create a new group named **consultants** with a GID of 900.

```
[student@desktopX ~]$ sudo groupadd -g 900 consultants
[student@desktopX ~]$ tail -5 /etc/group
stapdev:x:158:
pesign:x:989:
tcpdump:x:72:
slocate:x:21:
consultants:x:900:
```

3. Create three new users: **sspade**, **bboop**, and **dtracy**, with a password of **default** and add them to the supplementary group **consultants**. The primary group should remain as the user private group.

```
[student@desktopX ~]$ sudo useradd -G consultants sspade
[student@desktopX ~]$ sudo useradd -G consultants bboop
[student@desktopX ~]$ sudo useradd -G consultants dtracy
[student@desktopX ~]$ tail -5 /etc/group
slocate:x:21:
consultants:x:900:sspade,bboop,dtracy
sspade:x:1001:
bboop:x:1002:
dtracy:x:1003:
[student@desktopX ~]$ sudo passwd sspade
Changing password for user sspade.
New password: default
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: default
passwd: all authentication tokens updated successfully.
[student@desktopX ~]$ sudo passwd bboop
[student@desktopX ~]$ sudo passwd dtracy
```

4. Determine the date 90 days in the future and set each of the three new user accounts to expire on that date.

```
[student@desktopX ~]$ sudo chage -E $(date +%Y-%m-%d -d +90days) sspade
[student@desktopX ~]$ sudo chage -E $(date +%Y-%m-%d -d +90days) bboop
[student@desktopX ~]$ sudo chage -E $(date +%Y-%m-%d -d +90days) dtracy
```

5. Change the password policy for the **bboop** account to require a new password every 15 days.

```
[student@desktopX ~]$ sudo chage -M 15 bboop
[student@desktopX ~]$ chage -l bboop
Last password change          : Feb 04, 2014
Password expires              : Feb 19, 2014
Password inactive             : never
```

```

Account expires                : May 05, 2014
Minimum number of days between password change : 0
Maximum number of days between password change : 15
Number of days of warning before password expires : 7

```

6. Additionally, force all users to change their password on first login.

```

[student@desktopX ~]$ sudo chage -d 0 sspade
[student@desktopX ~]$ sudo chage -d 0 bboop
[student@desktopX ~]$ sudo chage -d 0 dtracy

```

7. Install the *ipa-client* package on your **desktopX** machine.

7.1. 

```
[student@desktopX ~]$ sudo yum -y install ipa-client
```

8. Configure your system, using **ipa-client-install**, to use the IPA server setup for the **serverX.example.com** DNS domain. Home directories should automatically be created, and NTP should not be configured during this process.

8.1. 

```

[student@desktopX ~]$ sudo ipa-client-install --domain=serverX.example.com --no-ntp --mkhomedir
Discovery was successful!
Hostname: desktopX.example.com
Realm: SERVERX.example.com
DNS Domain: serverX.example.com
IPA Server: serverX.example.com
BaseDN: dc=serverX,dc=example.com

Continue to configure the system with these values? [no]: yes
User authorized to enroll computers: admin
Password for admin@SERVERX.EXAMPLE.COM: redhat123
...
Client configuration complete.

```

9. Verify that you can now successfully log into **desktopX** as the user **ipausuer** by using **ssh**. The initial password is **password**, but this should be changed to **redhat123**. Due to the password change requirement, you will have to log in twice.

9.1. 

```

[student@desktopX ~]$ ssh ipausuer@desktopX.example.com
ipausuer@desktopX.example.com's password: password
Password expired. Change your password now.
Creating home directory for ipausuer.
WARNING: Your password has expired.
You must change your password now and login again!
Changing password for user ipausuer.
Current password: password
New password: redhat123
Retype new password: redhat123
passwd: all authentication tokens updated successfully.
Connection to desktopX.example.com closed.
[student@desktopX ~]$ ssh ipausuer@desktopX.example.com
ipausuer@desktopX.example.com's password: redhat123
Last login: Wed Feb 26 05:19:15 2014 from desktopX.example.com
-sh-4.2$ logout

```

10. When you finish, run the **lab combined-users grade** evaluation script to confirm you have done everything correctly.

```
[student@desktopX ~]$ lab combined-users grade
```



## Summary

### Users and Groups

List the roles of users and groups on a Linux system and view the local configuration files.

### Gaining Superuser Access

Escalate privilege to run commands as the superuser.

### Managing Local User Accounts

Add, remove, and modify local users with command-line tools.

### Managing Local Group Accounts

Manage local groups with command-line tools.

### Managing User Passwords

Manage password aging policies for users and manually lock, unlock, and expire accounts.

### Using Identity Management Services

- **authconfig{, -gtk, -tui}** can be used to configure a system to use centralized identity management services.
- **sssd** is configured to retrieve, validate, and cache authentication and user information in the background.





## CHAPTER 4

# FILE PERMISSIONS

Overview	
<b>Goal</b>	To control access to files and directories using permissions and access control lists (acls).
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Change the permissions and ownership of files using command-line tools.</li><li>• Configure a directory in which newly created files are automatically writable by members of the group which owns the directory, using special permissions and default umask settings.</li><li>• Describe POSIX access control lists.</li><li>• Manage POSIX access control lists.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Managing File System Permissions from the Command Line (and Practice)</li><li>• Managing Default Permissions and File Access (and Practice)</li><li>• POSIX Access Control Lists (ACLs) (and Practice)</li><li>• Securing Files with ACLs (and Practice)</li></ul>

# Managing File System Permissions from the Command Line

## Objectives

After completing this section, students should be able to change the permissions and ownership of files using command-line tools.

## Changing file/directory permissions

The command used to change permissions from the command line is **chmod**, short for "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).

### Symbolic method keywords:

```
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, executable)

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, it is possible to change one or more of the existing permissions. In order to accomplish this, use three symbols: **+** to add permissions to a set, **-** to remove permissions from a set, and **=** 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.

### Numeric method:

```
chmod ### file|directory
```

- Each digit represents an access level: user, group, other.
- **#** is sum of r=4, w=2, and x=1.

Using the *numeric* method, permissions are represented by a three-digit (or four, when setting advanced permissions) *octal* number. A single octal digit can represent the numbers **0-7**, exactly the number of possibilities for a three-bit number.

To convert between symbolic and numeric representation of permissions, we need to know how the mapping is done. In the three-digit octal (numeric) representation, each digit stands for one

group of permissions, from left to right: user, group, and other. In each of these groups, start with **0**. If the read permission is present, add **4**. Add **2** if write is present, and **1** for execute.

Numeric permissions are often used by advanced administrators since they are shorter to type and pronounce, while still giving full control over all permissions.

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-----**.

## Examples

- Remove read and write permission for group and other on **file1**:

```
[student@desktopX ~]$ chmod go-rw file1
```

- Add execute permission for everyone on **file2**:

```
[student@desktopX ~]$ chmod a+x file2
```

- Set read, write, and execute permission for user, read, and execute for group, and no permission for other on **sampledir**:

```
[student@desktopX ~]$ chmod 750 sampledir
```



## Note

The **chmod** commands supports the **-R** option for recursively setting permissions on an entire directory tree. When using this option, be sure to use the **X** permissions instead of the **x** permission to indicate that execute permissions should only be set on directories, and not regular files. For example, the following command will recursively set read and write access on **demodir** and all its children for their group owner, but will only apply execute permissions to directories, not regular files:

```
[student@desktopX ~]# chmod -R g+rwX demodir
```

## Changing file/directory user or group ownership

A newly created file is owned by the user who creates the file. By default, the new file has a group ownership which is the primary group of the user creating the file. Since Red Hat Enterprise Linux uses user private groups, this group is often a group with only that user as a

member. To grant access based on group membership, the owner or the group of a file may need to be changed.

File ownership can be changed with the **chown** command. For example, to grant ownership of the file **foofile** to **student**, the following command could be used:

```
[root@desktopX ~]# chown student foofile
```

**chown** can be used with the **-R** option to recursively change the ownership of an entire directory tree. The following command would grant ownership of **foodir** and all files and subdirectories within it to **student**:

```
[root@desktopX ~]# chown -R student foodir
```

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 will change the group **foodir** to **admins**:

```
[root@desktopX ~]# chown :admins foodir
```

The **chown** command can also be used to change both owner and group at the same time by using the syntax **owner:group**. For example, to change the ownership of **foodir** to **visitor** and the group to **guests**, use:

```
[root@desktopX ~]# chown visitor:guests foodir
```

Only **root** can change the ownership of a file. Group ownership, however, can be set by **root** or the file's owner. **root** can grant ownership to any group, while non-**root** users can grant ownership only to groups they belong to.



### Note

Instead of using **chown**, some users change the group ownership by using the **chgrp** command; this command works exactly the same as changing ownership with **chown**, including the use of **-R** to affect entire directory trees.



### References

**ls(1)**, **chmod(1)**, **chown(1)**, and **chgrp(1)** man pages

# Practice: Managing File Security from the Command Line

## Guided exercise

In this lab, you will create a collaborative directory for pre-existing users.

### Outcomes

A directory accessible by all members of the **ateam** group and a file created by Andy that can be modified by Alice.

### Before you begin...

Reset your serverX system.

- ☐ 1. Log into the GNOME desktop on serverX as **student** with a password of **password**.
- ☐ 2. Open a window with a Bash prompt.  
Select **Applications > Utilities > Terminal**.
- ☐ 3. Become the **root** user at the shell prompt.

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

- ☐ 4. Run **lab permissions setup** which will create a shared group, **ateam**, with two new users, **andy** and **alice**. The password for these accounts is **password**

```
[root@serverX ~]# lab permissions setup
```

- ☐ 5. Create a directory in **/home** called **ateam-text**.

```
[root@serverX ~]# mkdir /home/ateam-text
```

- ☐ 6. Change the group ownership of the **ateam-text** directory to **ateam**.

```
[root@serverX ~]# chown :ateam /home/ateam-text
```

- ☐ 7. Ensure the permissions of **ateam-text** allows group members to create and delete files.

```
[root@serverX ~]# chmod g+w /home/ateam-text
```

- ☐ 8. Ensure the permissions of **ateam-text** forbids others from accessing its files.

```
[root@serverX ~]# chmod 770 /home/ateam-text  
[root@serverX ~]$ ls -ld /home/ateam-text  
drwxrwx---. 1 root ateam 6 Jan 23 12:50 /home/ateam-text
```

- ❑ 9. Exit the root shell and switch to the user **andy** with a password of **password**.

```
[root@serverX ~]# exit
[student@serverX ~]$ su - andy
Password: password
```

- ❑ 10. Navigate to the **/home/ateam-text** folder (remember to open a terminal window first).

```
[andy@serverX ~]$ cd /home/ateam-text
```

- ❑ 11. Create an empty file called **andyfile3**.

```
[andy@serverX ateam-text]$ touch andyfile3
```

- ❑ 12. Record the default user and group ownership of the new file and its permissions.

```
[andy@serverX ateam-text]$ ls -l andyfile3
-rw-rw-r--. 1 andy andy 0 Jan 23 12:59 andyfile3
```

- ❑ 13. Change the group ownership of the new file to **ateam** and record the new ownership and permissions.

```
[andy@serverX ateam-text]$ chown :ateam andyfile3
[andy@serverX ateam-text]$ ls -l andyfile3
-rw-rw-r--. 1 andy ateam 0 Jan 23 12:59 andyfile3
```

- ❑ 14. Exit the shell and switch to the user **alice** with a password of **password**.

```
[root@serverX ~]# exit
[student@serverX ~]$ su - alice
Password: password
```

- ❑ 15. Navigate to the **/home/ateam-text** folder.

```
[alice@serverX ~]$ cd /home/ateam-text
```

- ❑ 16. Determine **alice's** privileges to access and/or modify **andyfile3**.

```
[alice@serverX ateam-text]$ echo "text" >> andyfile3
[alice@serverX ateam-text]$ cat andyfile3
text
```



# Managing Default Permissions and File Access

## Objectives

After completing this section, students should be able to configure a directory in which newly created files are automatically writable by members of the group which owns the directory, using special permissions and default umask settings.

## Special permissions

The **setuid** (or **setgid**) permission on an executable file means that the command will run as the **user** (or **group**) of the file, not as the user that ran the command. One example is the **passwd** command:

```
[student@desktopX ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 35504 Jul 16 2010 /usr/bin/passwd
```

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. Users with write permission on the directory cannot force saves to files within the directory owned by other users. An example is **/tmp**:

```
[student@desktopX ~]$ ls -ld /tmp
drwxrwxrwt. 39 root root 4096 Feb 8 20:52 /tmp
```

Lastly, **setgid** on a directory means that files created in the directory will inherit the group affiliation 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.

### Effects of special permissions on files and directories

Special permission	Effect on files	Effect on directories
<b>u+s</b> (suid)	File executes as the user that owns the file, not the user that ran the file.	No effect.
<b>g+s</b> (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.
<b>o+t</b> (sticky)	No effect.	Users with <b>write</b> on the directory can only remove files that they own; they cannot remove or force saves to files owned by other users.

## 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**:

```
[root@desktopX ~]# chmod g+s directory
```

- Set the setgid bit, and read/write/execute for user and group on **directory**:

```
[root@desktopX ~]# chmod 2770 directory
```

## Default file permissions

The default permissions for files are set by the processes that create them. For example, text editors create files so they are readable and writeable, but not executable, by everyone. The same goes for shell redirection. Additionally, binary executables are created executable by the compilers that create them. The **mkdir** command creates new directories with all permissions set—read, write, and execute.

Experience shows that these permissions are not typically set when new files and directories are created. This is because some of the permissions are cleared by the umask of the shell process. The **umask** command without arguments will display the current value of the shell's umask:

```
[student@desktopX ~]$ umask
0002
```

Every process on the system has a umask, which is an octal bitmask that is used to clear the permissions of new files and directories that are created by the process. If a bit is set in the umask, then the corresponding permission is cleared in new files. For example, the previous 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 077 clears all the group and other permissions of newly created files.

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. If it is less than 3 digits, leading zeros are assumed.

The system default umask values for Bash shell users are defined in the **/etc/profile** and **/etc/bashrc** files. Users can override the system defaults in their **.bash\_profile** and **.bashrc** files.

In this example, please follow along with the next steps while your instructor demonstrates the effects of **umask** on new files and directories.

1. Create a new file and directory to see how the default umask affects permissions.

```
[student@desktopX ~]$ touch newfile1
[student@desktopX ~]$ ls -l newfile1
-rw-rw-r--. 1 student student 0 May  9 01:54 newfile1
[student@desktopX ~]$ mkdir newdir1
[student@desktopX ~]$ ls -ld newdir1
drwxrwxr-x. 2 student student 0 May  9 01:54 newdir1
```

- Set the umask value to 0. This setting will not mask any of the permissions of new files. Create a new file and directory to see how this new umask affects permissions.

```
[student@desktopX ~]$ umask 0
[student@desktopX ~]$ touch newfile2
[student@desktopX ~]$ ls -l newfile2
-rw-rw-rw-. 1 student student 0 May  9 01:54 newfile2
[student@desktopX ~]$ mkdir newdir2
[student@desktopX ~]$ ls -ld newdir2
drwxrwxrwx. 2 student student 0 May  9 01:54 newdir2
```

- Set the umask value to 007. This setting will mask all of the "other" permissions of new files.

```
[student@desktopX ~]$ umask 007
[student@desktopX ~]$ touch newfile3
[student@desktopX ~]$ ls -l newfile3
-rw-rw----. 1 student student 0 May  9 01:55 newfile3
[student@desktopX ~]$ mkdir newdir3
[student@desktopX ~]$ ls -ld newdir3
drwxrwx---. 2 student student 0 May  9 01:54 newdir3
```

- Set the umask value to 027. This setting will mask write access for group members and all of the "other" permissions of new files.

```
[student@desktopX ~]$ umask 027
[student@desktopX ~]$ touch newfile4
[student@desktopX ~]$ ls -l newfile4
-rw-r-----. 1 student student 0 May  9 01:55 newfile4
[student@desktopX ~]$ mkdir newdir4
[student@desktopX ~]$ ls -ld newdir4
drwxr-x---. 2 student student 0 May  9 01:54 newdir4
```

- Log in as **root** to change the default umask for unprivileged users to prohibit all access for users not in their group.

Modify **/etc/bashrc** and **/etc/profile** to change the default umask for Bash shell users. Since the default umask for unprivileged users is 0002, look for the **umask** command in these files that sets the umask to that value. Change them to set the umask to 007.

```
[root@desktopX ~]# less /etc/bashrc
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 002
else
    umask 022
fi

# Only display echos from profile.d scripts if we are no login shell
[root@desktopX ~]# vim /etc/bashrc
[root@desktopX ~]# less /etc/bashrc
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
```

```

        umask 022
    fi

    # Only display echos from profile.d scripts if we are no login shell
[root@desktopX ~]# less /etc/profile
    # You could check uidgid reservation validity in
    # /usr/share/doc/setup-*/uidgid file
    if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
        umask 002
    else
        umask 022
    fi

    for i in /etc/profile.d/*.sh ; do
[root@desktopX ~]# vim /etc/profile
[root@desktopX ~]# less /etc/profile
    # You could check uidgid reservation validity in
    # /usr/share/doc/setup-*/uidgid file
    if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
        umask 007
    else
        umask 022
    fi

    for i in /etc/profile.d/*.sh ; do

```

6. Log back in as **student** and confirm that the umask changes you made are persistent.

```

[student@desktopX ~]$ umask
0007

```



## Note

Other shells, such as **tcsh**, may have different system default initialization files in **/etc** and users' home directories.



## References

**bash(1)**, **ls(1)**, **chmod(1)**, and **umask(1)** man pages

# Practice: Controlling New File Permissions and Ownership

## Guided exercise

In this lab, you will control default permissions on new files using the **umask** command and **setgid** permission.

### Outcomes

- Create a shared directory where new files are automatically owned by the group **ateam**.
- Experiment with various umask settings.
- Adjust default permissions for specific users.
- Confirm your adjustment is correct.

### Before you begin...

Reset your serverX system. Run **lab permissions setup** to create the **alice** account. The password for **alice** is **password**.

- ❑ 1. Log in as **alice** on your **serverX** virtual machine and open a window with a Bash prompt. Use the **umask** command without arguments to display Alice's default umask value.

```
[alice@serverX ~]$ umask
0002
```

- ❑ 2. Create a new directory **/tmp/shared** and a new file **/tmp/shared/defaults** to see how the default umask affects permissions.

```
[alice@serverX ~]$ mkdir /tmp/shared
[alice@serverX ~]$ ls -ld /tmp/shared
drwxrwxr-x. 2 alice alice 6 Jan 26 18:43 /tmp/shared
[alice@serverX ~]$ touch /tmp/shared/defaults
[alice@serverX ~]$ ls -l /tmp/shared/defaults
-rw-rw-r--. 1 alice alice 0 Jan 26 18:43 /tmp/shared/defaults
```

- ❑ 3. Change the group ownership of **/tmp/shared** to **ateam** and record the new ownership and permissions.

```
[alice@serverX ~]$ chown :ateam /tmp/shared
[alice@serverX ~]$ ls -ld /tmp/shared
drwxrwxr-x. 2 alice ateam 21 Jan 26 18:43 /tmp/shared
```

- ❑ 4. Create a new file in **/tmp/shared** and record the ownership and permissions.

```
[alice@serverX ~]$ touch /tmp/shared/alice3
[alice@serverX ~]$ ls -l /tmp/shared/alice3
-rw-rw-r--. 1 alice alice 0 Jan 26 18:46 /tmp/shared/alice3
```

- ❑ 5. Ensure the permissions of **/tmp/shared** cause files created in that directory to inherit the group ownership of **ateam**.

```
[alice@serverX ~]$ chmod g+s /tmp/shared
[alice@serverX ~]$ ls -ld /tmp/shared
drwxrwsr-x. 2 alice ateam 34 Jan 26 18:46 /tmp/shared
[alice@serverX ~]$ touch /tmp/shared/alice4
[alice@serverX ~]$ ls -l /tmp/shared/alice4
-rw-rw-r--. 1 alice ateam 0 Jan 26 18:48 /tmp/shared/alice4
```

- ❑ 6. Change the umask for **alice** such that new files are created with read-only access for the group and no access for other users. Create a new file and record the ownership and permissions.

```
[alice@serverX ~]$ umask 027
[alice@serverX ~]$ touch /tmp/shared/alice5
[alice@serverX ~]$ ls -l /tmp/shared/alice5
-rw-r-----. 1 alice ateam 0 Jan 26 18:48 /tmp/shared/alice5
```

- ❑ 7. Open a new Bash shell as **alice** and view the umask.

```
[alice@serverX ~]$ umask
0002
```

- ❑ 8. Change the default umask for **alice** to prohibit all access for users not in their group.

```
[alice@serverX ~]$ echo "umask 007" >> ~/.bashrc
[alice@serverX ~]$ 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
umask 007
```

- ❑ 9. Log out and back into **serverX** as **alice** and confirm that the umask changes you made are persistent.

```
[alice@serverX ~]$ umask
0007
```

# POSIX Access Control Lists (ACLs)

## Objectives

After completing this section, students should be able to:

- Describe ACLs and file system mount options.
- View and interpret ACLs with **ls** and **getfacl**, describe the ACL mask and ACL permission precedence.

## Access control list concepts

Standard Linux file permissions are satisfactory for most situations, but they have limitations. Permissions restricting access to a file are limited to the file owner, membership of a single group, or everyone else. It may not be appropriate for the process (a running program) to be a member of the file's owning group, and even less desirable to grant permission to everyone.

ACLs allow fine-grained permissions to be allocated to a file. Named users or named groups, as well as users and groups identified by a UID or GUID, can be granted permissions, in addition to the standard *file owner*, *group-owner*, and *other* file permissions. The same permission flags apply: **r** - read, **w** - write, and **x** - execute (on files, search for directories).

The file owner can set ACLs on individual files or directories. New files and subdirectories can automatically inherit ACL settings from the parent directory *default* ACLs, if they are set. Similar to normal file access rules, the parent directory hierarchy will need at least the *other* execute permission set to enable named users and named groups to have access.

### File system mount option

The file system needs to be mounted with ACL support enabled. XFS file systems have built-in ACL support. Ext4 file systems created on Red Hat Enterprise Linux 7 have the **acl** option enabled by default, but ext4 file systems created in earlier versions of Red Hat Enterprise Linux may need the **acl** option included with the mount request, or set in the superblock.

## Viewing and interpreting ACL permissions

The **ls -l** command only outputs minimal ACL setting details:

```
[student@serverX steamies]$ ls -l roster.txt
-rwxrw----+ 1 student controller 130 Mar 19 23:56 roster.txt
```

The "+" at the end of the 10-character permission string indicates that there are ACL settings associated with this file. Interpret the *user*, *group*, and *other* "rwx" flags as:

- **user:** Shows the *user* ACL settings, which are the same as the standard *user* file settings; **rwx**.
- **group:** Shows the current ACL *mask* settings, not the *group-owner* settings; **rw**.
- **other:** Shows the *other* ACL settings, which are the same as the standard *other* file settings; no access.



## Important

Changing group permissions on a file with an ACL by using **chmod** does not change the group-owner permissions, but does change the ACL mask. Use **setfacl -m g::perms file** if the intent is to update the file's group-owner permissions.

### View file ACLs

To display ACL settings on a file, use **getfacl file**:

```
[student@serverX steamies]$ getfacl roster.txt
# file: roster.txt
# owner: student
# group: controller
user::rwx
user:james:---
user:1005:rwx      #effective:rw-
group::rwx        #effective:rw-
group:sodor:r--
group:2210:rwx     #effective:rw-
mask::rw-
other:---
```

Take a look at each section of the previous example:

#### Opening comment entries:

```
# file: roster.txt
# owner: student
# group: controller
```

The first three lines are comments that identify the file name, owner (**student**), and group-owner (**controller**). If there are any additional file flags—for example, **setuid** or **setgid**—then a fourth comment line will appear showing which flags are set.

#### User entries:

```
user::rwx           ❶
user:james:---      ❷
user:1005:rwx      #effective:rw-  ❸
```

- ❶ File owner permissions. **student** has **rwX**.
- ❷ Named user permissions. One entry for each named user associated with this file. **james** has NO permissions.
- ❸ Named user permissions. UID **1005** has **rwX**, but the mask limits the effective permissions to **rw** only.

#### Group entries:

```
group::rwx          #effective:rw-  ❶
group:sodor:r--     ❷
group:2210:rwx      #effective:rw-  ❸
```



- ❶ Group-owner permissions. **controller** has **rwX**, but the mask limits the effective permissions to **rw** only.
- ❷ Named group permissions. One entry for each named group associated with this file. **sodor** has **r** only.
- ❸ Named group permissions. GID **2210** has **rwX**, but the mask limits the effective permissions to **rw** only.

Mask entry:

```
mask::rw-
```

Mask settings show the maximum permissions possible for all named users, the group-owner and named groups. UID **1005**, **controller**, and GID **2210** cannot execute this file, even though each entry has the execute permission set.

Other entry:

```
other::---
```

Other or "world" permissions. All other UIDs and GIDs have NO permissions.

### View directory ACLs

To display ACL settings on a directory, use **getfacl /directory**:

```
[student@serverX steamies]$ getfacl .
# file: .
# owner: student
# group: controller
# flags: -s-
user::rwX
user:james:---
user:1005:rwX
group::rwX
group:sodor:r-x
group:2210:rwX
mask::rwX
other:---
default:user::rwX
default:user:james:---
default:group::rwX
default:group:sodor:r-x
default:mask::rwX
default:other:---
```

Take a look at each section of the previous example:

Opening comment entries:

```
# file: .
# owner: student
# group: controller
# flags: -s-
```

The first three lines are comments that identify the directory name, owner (**student**), and group-owner (**controller**). If there are any additional directory flags (**setuid**, **setgid**, **sticky**), then a fourth comment line will appear showing the set flags—in this case, **setgid**.

*Standard ACL entries:*

```
user::rwx
user:james:---
user:1005:rwx
group::rwx
group:sodor:r-x
group:2210:rwx
mask::rwx
other:---
```

The ACL permissions on this directory are the same as the file example earlier, but apply to the directory. The key difference is the inclusion of the execute permission on these entries (when appropriate) to allow directory search permission.

*Default user entries:*

```
default:user::rwx
default:user:james:---
```

- ❶ Default file owner ACL permissions. The file owner will get **rwx**, read/write on new files and execute on new subdirectories.
- ❷ Default named user ACL permissions. One entry for each named user who will automatically get default ACLs applied to new files or subdirectories. **james** will always default to NO permissions.

*Default group entries:*

```
default:group::rwx
default:group:sodor:r-x
```

- ❶ Default group-owner ACL permissions. The file group-owner will get **rwx**, read/write on new files and execute on new subdirectories.
- ❷ Default named group ACL permissions. One entry for each named group which will automatically get default ACLs. **sodor** will get **rx**, read-only on new files, and execute on new subdirectories.

*Default ACL mask entry:*

```
default:mask::rwx
```

Default mask settings show the initial maximum permissions possible for all new files or directories created that have named user ACLs, the group-owner ACL, or named group ACLs: read and write for new files and execute permission on new subdirectories, new files never get execute permission.

*Default other entry:*

```
default:other:---
```

Default *other* or "world" permissions. All other UIDs and GIDs have NO permissions to new files or new subdirectories.

The *default* entries in the previous example do not include the named user (UID **1005**) and named group (GID **2210**); consequently, they will not automatically get initial ACL entries added for them to any new files or new subdirectories. This effectively limits them to files and subdirectories that they already have ACLs on, or if the relevant file owner adds the ACL later using **setfacl**. They can still create their own files and subdirectories.



## Note

The output from **getfacl** can be used as input to **setfacl**. Use **getfacl -R / *directory*** to generate output for the directory and its content. This output can be saved and used for recovery by passing the output to **setfacl --set-file=*file*** to do a mass update.

### The ACL mask

The ACL mask defines the maximum permissions that can be granted to *named users*, the *group-owner*, and *named groups*. It does not restrict the permissions of the *file owner* or *other* users. All files and directories that implement ACLs will have an ACL mask.

The mask can be viewed with **getfacl** and explicitly set with **setfacl**. It will be calculated and added automatically if it is not explicitly set, but it could also be inherited from a parent directory default mask setting. By default, the mask is recalculated whenever any of the affected ACLs is added, modified, or deleted.

### ACL permission precedence

When determining whether a process (a running program) can access a file, file permissions and ACLs are applied as follows:

- If the process is running as the user that owns the file, then the file's user ACL permissions apply.
- If the process is running as a user that is listed in a named user ACL entry, then the named user ACL permissions apply (as long as it is permitted by the mask).
- If the process is running as a group that matches the group-owner of the file, or as a group with an explicit named group ACL entry, then the matching ACL permissions apply (as long as it is permitted by the mask).
- Otherwise, the file's *other* ACL permissions apply.



## References

**acl(5)**, **getfacl(1)**, **ls(1)** man pages

## Practice: Interpret ACLs

### Quiz

Match the following items to their counterparts in the table.

<b>default:m::rx /directory</b>		
<b>default:user:mary:rx /directory</b>	<b>g::rw /directory</b>	
<b>g::rw file</b>	<b>getfacl /directory</b>	
<b>group:hug:rwx /directory</b>	<b>user::rx file</b>	
<b>user:mary:rx file</b>		

Description	ACL operation
Display ACLs on a directory.	
Named user with read, execute permissions for a file.	
File owner with read, execute permissions for a file.	
Read, write permissions for a directory granted to the directory group-owner.	
Read, write permissions for a file granted to the file group-owner.	

Description	ACL operation
Read, write, execute permissions for a directory granted to a named group.	
Read, execute permissions set as the default mask.	
Named user granted initial read permission for new files and read, execute permission for new subdirectories.	

## Solution

Match the following items to their counterparts in the table.

Description	ACL operation
Display ACLs on a directory.	<b>getfacl /directory</b>
Named user with read, execute permissions for a file.	<b>user:mary:rx file</b>
File owner with read, execute permissions for a file.	<b>user::rx file</b>
Read, write permissions for a directory granted to the directory group-owner.	<b>g::rw /directory</b>
Read, write permissions for a file granted to the file group-owner.	<b>g::rw file</b>
Read, write, execute permissions for a directory granted to a named group.	<b>group:hug:rwx /directory</b>
Read, execute permissions set as the default mask.	<b>default:m::rx /directory</b>
Named user granted initial read permission for new files and read, execute permission for new subdirectories.	<b>default:user:mary:rx /directory</b>

# Securing Files with ACLs

## Objectives

After completing this section, students should be able to:

- Change regular ACL file permissions using **setfacl**.
- Control default ACL file permissions for new files and directories.

## Changing ACL file permissions

Use **setfacl** to add, modify, or remove standard ACLs on files and directories.

ACLs use the normal file system representation of permissions, "**r**" for read permission, "**w**" for write permission, and "**x**" for execute permission. A "-" (dash) indicates that the relevant permission is absent. When (recursively) setting ACLs, an uppercase "**X**" can be used to indicate that execute permission should only be set on directories and not regular files, unless the file already has the relevant execute permission. This is the same behavior as **chmod**.

### Adding or modifying an ACL

ACLs can be set via the command line using **-m**, or passed in via a file using **-M** (use "-" (dash) instead of a file name for *stdin*). These two options are the "modify" options; they add new ACL entries or replace specific existing ACL entries on a file or directory. Any other existing ACL entries on the file or directory remain untouched.



### Note

Use the **--set** or **--set-file** options to completely replace the ACL settings on a file.

When first defining an ACL on a file, if the add operation does not include settings for the *file owner*, *group-owner*, or *other* permissions, then they will be set based on the current standard file permissions (these are also known as the *base* ACLs and cannot be deleted), and a new *mask* value will be calculated and added as well.

To add or modify a *user* or *named user* ACL:

```
[student@serverX ~]$ setfacl -m u:name:rX file
```

If *name* is left blank, then it applies to the *file owner*, otherwise *name* can be a username or UID value. In this example, the permissions granted would be read-only, and if already set, execute (unless *file* was a directory, in which case the directory would get the execute permission set to allow directory search).

ACL *file owner* and standard *file owner* permissions are equivalent; consequently, using **chmod** on the *file owner* permissions is equivalent to using **setfacl** on the *file owner* permissions. **chmod** has no effect on named users.

To add or modify a *group* or *named group* ACL:

```
[student@serverX ~]$ setfacl -m g:name:rw file
```

This follows the same pattern for adding or modifying a user ACL. If *name* is left blank, then it applies to the *group-owner*. Otherwise, specify a group name or GID value for a *named group*. The permissions would be read and write in this example.

**chmod** has no effect on any group permissions for files with ACL settings, but it updates the ACL mask.

To add or modify the *other* ACL:

```
[student@serverX ~]$ setfacl -m o::- file
```

*other* only accepts permission settings. It is common for the permission to be set to "-" (dash), which specifies that *other* users have NO permissions, but any of the standard permissions can be specified.

ACL *other* and standard *other* permissions are equivalent, so using **chmod** on the *other* permissions is equivalent to using **setfacl** on the *other* permissions.

Add multiple entries via the same command, and comma-separate each of the entries:

```
[student@serverX ~]$ setfacl -m u::rwx,g:sodor:rX,o::- file
```

This will set the *file owner* to read, write, and execute, set the named group **sodor** to read-only and conditional execute, and restrict all *other* users to NO permissions. The *group-owner* will maintain their existing file or ACL permissions and other "named" entries will remain unchanged.

#### Using getfacl as input

The output from **getfacl** can be used as input to **setfacl**:

```
[student@serverX ~]$ getfacl file-A | setfacl --set-file=- file-B
```

**--set-file** accepts input from a file or *stdin*, and the "-" (dash) specifies the use of *stdin*. In this case, *file-B* will have the same ACL settings as *file-A*.

#### Setting an explicit ACL mask

An ACL mask can be explicitly set on a file or directory to limit the maximum effective permissions for named users, the *group-owner*, and named groups. This restricts any existing permissions that exceed the mask, but does nothing to permissions that are less permissive than the mask.

```
[student@serverX ~]$ setfacl -m m::r file
```

This would add a mask value that restricted any *named users*, the *group-owner*, and any *named groups* to read-only permission, regardless of their existing settings. The *file owner* and *other* users are not impacted by the mask setting.

**getfacl** will show an "effective" comment beside entries that are being restricted by a mask setting.





## Important

By default, the ACL mask is recalculated each time one of the impacted ACL settings (named users, group-owner, or named groups) is modified or deleted, potentially resetting a previous explicit mask setting.

To avoid the mask recalculation, use **-n** or include a mask setting (**-m m::perms**) with any **setfacl** operation that modifies mask-affected ACL settings.

### Recursive ACL modifications

When setting an ACL on a directory, it is common to want to apply the ACL recursively to the directory structure and files. Use the **-R** option to do this. The "X" (capital X) permission is often used with recursion, so that files with the execute permission set retain the setting and directories get the execute permission set to allow directory search. It is considered good practice to also use the uppercase X when non-recursively setting ACLs, as it prevents an administrator from accidentally adding execute permissions to a regular file.

```
[student@serverX ~]$ setfacl -R -m u:name:rX directory
```

This would add the user *name* to the *directory* and all existing files and subdirectories, granting read-only and conditional execute.

### Deleting an ACL

Deleting specific ACL entries follows the same basic format as the modify operation, except the *perms* should not be specified.

```
[student@serverX ~]$ setfacl -x u:name,g:name file
```

This would only remove the named user and the named group from the list of file or directory ACLs. Any other existing ACLs remain active.

It is possible to use the delete (**-x**) and modify (**-m**) operations in the same **setfacl** operation.

The mask can only be deleted if there are no other ACLs set (excluding the *base* ACLs which cannot be deleted), so it must be deleted last. The file will no longer have ACLs and **ls -l** will not show the "+" symbol next to the permissions string. Alternatively, to delete ALL ACLs on a file or directory (including *default* ACLs on directories), use:

```
[student@serverX ~]$ setfacl -b file
```

## Controlling default ACL file permissions

A directory can have *default* ACLs set on it that are automatically inherited by all new files and new subdirectories. There can be *default* ACL permissions set for each of the standard ACL settings, including a default mask.

A directory still requires standard ACLs for access control because *default* ACLs do not implement access control for the directory; they only provide ACL permission inheritance support.

*An example:*

```
[student@serverX ~]$ setfacl -m d:u:name:rx directory
```

This adds a default named user (**d:u:name**) with read-only permission and execute permission on subdirectories.

The **setfacl** command for adding a *default* ACL for each of the ACL types is exactly the same as for standard ACLs, but prefaced with **d:.** Alternatively, use the **-d** option on the command line.



## Important

When setting *default* ACLs on a directory, ensure that users will be able to access the contents of new subdirectories created in it by including the execute permission on the *default* ACL.

Users will not automatically get the execute permission set on newly created regular files because unlike new directories, the ACL *mask* of a new regular file is **rw-**.



## Note

New files and new subdirectories continue to get their owner UID and primary group GID values set from the creating user, except when the parent directory **setgid** flag is enabled, in which case the primary group GID will be the same as the parent directory GID.

### Deleting default ACLs

Deleting a *default* ACL is also the same as deleting a standard ACL; again, preface with **d:**, or use the **-d** option.

```
[student@serverX ~]$ setfacl -x d:u:name directory
```

This removes the *default* ACL that was added in the previous example.

To delete all *default* ACLs on a directory, use **setfacl -k /directory**. To delete ALL ACLs on a directory, use **setfacl -b /directory**.



## References

**acl(5)**, **setfacl(1)** man pages

# Practice: Using ACLs to Grant and Limit Access

## Guided exercise

In this lab, you will add a named group access control list (ACL) and a named user ACL to an existing share folder and its content. You will set up *default* ACLs to ensure future files and directories get the correct permissions.

Resources:	
<b>Files:</b>	/shares/steamies/*, /shares/steamies/display_engines.sh
<b>Machines:</b>	serverX

### Outcomes:

- Members of the **sodor** group will have the same access permissions as the **controller** group on the **steamies** directory, except **james**, who has no access.
- Existing files and directories will be updated to reflect the new **sodor** and **james** ACL permissions.
- New files and directories will automatically get the correct ACL and file permissions.

### Before you begin...

- Reset your serverX system.
- Log into and set up your server system.

```
[student@serverX ~]$ lab acl setup
```

- Open a terminal.
- Switch to **root** using **sudo -i**.

Student is a controller for the Sodor Island Rail network. There is a properly configured share directory located at **/shares/steamies** that hosts files detailing rostering, steam engines, etc.

Currently, only members of the **controller** group have access to this directory, but it has been decided that members of the **sodor** group would benefit from full access to this directory.

James, a member of the **sodor** group, has caused *chaos and confusion* on many occasions, so he is to be denied access to the directory, at least until he shows that he is a *really useful engine*.

Your task is to add appropriate ACLs to the directory and its contents, so that members of the **sodor** group have full access, but deny user **james** any access. Make sure that future files and directories stored in **/shares/steamies** get appropriate ACLs applied.

### Important information:

- **controller** group: **student**

- **sodor** group: **thomas**, **james**
- There is a subdirectory called **engines** and numerous files to test the ACLs. Also, there is an executable script you can test.
- Thomas and James have their passwords set to **redhat**.
- All changes should occur to directory **steamies** and its files; do not adjust the **shares** directory.

❑ 1. Add the named ACLs to the **steamies** directory and all of its content.

- ❑ 1.1. Use **setfacl** to recursively update the **steamies** directory, granting the **sodor** group read, write, and conditional execute permissions.

```
[root@serverX ~]# setfacl -Rm g:sodor:rwX /shares/steamies
```

-R recursive, -m modify/add, :rwX read/write/eXecute (but only on directories and existing executables)

- ❑ 1.2. Use **setfacl** to recursively update the **steamies** directory, denying the user **james** from the **sodor** group any access.

```
[root@serverX ~]# setfacl -Rm u:james:- /shares/steamies
```

-R recursive, -m modify/add, :- no permissions

❑ 2. Add the named ACLs as *default* ACLs to support future file and directory additions.

- ❑ 2.1. Use **setfacl** to add a default access rule for the **sodor** group. Grant read, write, and execute permissions on the **steamies** directory.

```
[root@serverX ~]# setfacl -m d:g:sodor:rwX /shares/steamies
```

-m modify/add, d:g default group, :rwX read/write/execute (needed for proper subdirectory creation and access)

- ❑ 2.2. Use **setfacl** to add a default access rule for the user **james**. Deny all access to the **steamies** directory.

```
[root@serverX ~]# setfacl -m d:u:james:- /shares/steamies
```

-m modify/add, d:u default user, :- no permissions

❑ 3. Verify your ACL changes.

Thomas should be able to read any file, create a new directory with a new file in it, and execute the **display\_engines.sh** script.

James should not be able to read, write, or execute any file; this includes being unable to list the directory contents.

Use **sudo -i -u user** to switch to your test users. Use **exit** or **Ctrl+D** to leave the test user shell.

```
[root@serverX ~]# exit
[student@serverX ~]$ sudo -i -u thomas
[thomas@serverX ~]$ cd /shares/steamies/
```

- 3.1. Use **cat** to check that Thomas can read a file.

```
[thomas@serverX steamies]$ cat roster.txt
James - Shunting at Brendam docks
Percy - Overnight mail run
Henry - Flying Kipper run
Thomas - Annie and Clarabel, Knapford line
```

- 3.2. Use **display\_engines.sh** to check that Thomas can execute a script.

```
[thomas@serverX steamies]$ ./display_engines.sh
They're two, they're four, they're six, they're eight ...
Edward wants to help and share
...
Toby, well let's say, he's square
```

- 3.3. Use **mkdir** to create a directory as Thomas.

Use **echo** to create a file in the new directory as Thomas.

Switch back to **student** when you are finished.

```
[thomas@serverX steamies]$ mkdir tidmouth
[thomas@serverX steamies]$ echo "toot toot" > tidmouth/whistle.txt
[thomas@serverX steamies]$ exit
```

- 3.4. Use **cd** to try and change into the directory as James, and also try **ls** to list the directory. Both commands should fail with **Permission denied**.

You could try one or more of the commands Thomas issued, but as James, to further verify his lack of access. Try prefixing each file with the full path, **/shares/steamies**, because you cannot **cd** into the directory.

Switch back to **student** when you are finished testing **james**.

```
[student@serverX ~]$ sudo -i -u james
[james@serverX ~]$ cd /shares/steamies/
-bash: cd: /shares/steamies/: Permission denied
[james@serverX ~]$ ls /shares/steamies/
ls: cannot open directory /shares/steamies: Permission denied
[james@serverX ~]$ cat /shares/steamies/roster.txt
cat: /shares/steamies/roster.txt: Permission denied
[james@serverX ~]$ exit
```

- 3.5. Use **getfacl** to see all the ACLs on **/shares/steamies** and the ACLs on **/shares/steamies/tidmouth**.



## Note

Use **newgrp controller** to switch *student* to the *controller* group.

The **lab acl setup** script adds *controller* as a supplementary group to *student*; however, unless you have restarted the shell prior to this step, then the current shell does not yet recognize the new membership and **getfacl** on *tidmouth* will get **Permission denied**.

```
[student@serverX ~]$ newgrp controller
[student@serverX ~]$ getfacl /shares/steamies
getfacl: Removing leading '/' from absolute path names
# file: shares/steamies/
# owner: root
# group: controller
# flags: -s-
user::rwx
user:james:---
group::rwx
group:sodor:rwx
mask::rwx
other:---
default:user::rwx
default:user:james:---
default:group::rwx
default:group:sodor:rwx
default:mask::rwx
default:other:---

[student@serverX ~]$ getfacl /shares/steamies/tidmouth
getfacl: Removing leading '/' from absolute path names
# file: shares/steamies/tidmouth
# owner: thomas
# group: controller
# flags: -s-
user::rwx
user:james:---
group::rwx
group:sodor:rwx
mask::rwx
other:---
default:user::rwx
default:user:james:---
default:group::rwx
default:group:sodor:rwx
default:mask::rwx
default:other:---
```

## Summary

### Managing File System Permissions from the Command Line

Modify ownership and permissions of files and directories using **chmod** and **chown**.

### Managing Default Permissions and File Access

Explain how default permissions are set by the system and use **umask** and **SGID** to control automatic access to files.

### POSIX Access Control Lists (ACLs)

- ACLs provide fine-grained access control to files and directories.
- The file system must be mounted with ACL support enabled; XFS has built-in ACL support.
- **ls -l** indicates the presence of ACL settings with the "+" character. The group permissions show the *mask* settings.
- **getfacl file** displays the ACLs on a file or directory; directory ACLs include default ACLs.
- An ACL mask defines the maximum permissions *named users*, the *group-owner*, and *named groups* can have.
- ACL permission precedence is *user*, *named users*, *groups*, and then *others*.

### Securing Files with ACLs

- How to use **setfacl -m acl\_spec** to add or modify.
- How to use **setfacl -x acl\_spec** to delete.
- Default ACLs can be set on a directory; preface the *acl\_spec* with **d:**. Include execute permission to ensure access to new subdirectories.
- How to use **-R** for recursive, **-b** to delete all ACLs, **-k** to delete all default ACLs.
- The *acl\_spec* has the pattern **type:name:perms**.
  - *type* can be **u**, **g**, **o**, or **m**.
  - *name* can be a **username**, **uid**, **group-name**, or **gid**. An empty name implies *file owner* or *group owner*.
  - *perms* are **r**, **w**, **x**, or **X**. "-" means unset.







## CHAPTER 5

# SELINUX PERMISSIONS

Overview	
<b>Goal</b>	To manage the Security Enhanced Linux (SELinux) behavior of a system to keep it secure in case of a network service compromise.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Explain the basics of SELinux permissions.</li><li>• Change SELinux modes with setenforce.</li><li>• Change file contexts with semanage and restorecon.</li><li>• Manage SELinux booleans with setsebool.</li><li>• Examine logs and use sealert to troubleshoot SELinux violations.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Enabling and Monitoring SELinux (and Practice)</li><li>• Changing SELinux Modes (and Practice)</li><li>• Changing SELinux Contexts (and Practice)</li><li>• Changing SELinux Booleans (and Practice)</li><li>• Troubleshooting SELinux (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Managing SELinux Security</li></ul>

# Enabling and Monitoring Security Enhanced Linux (SELinux)

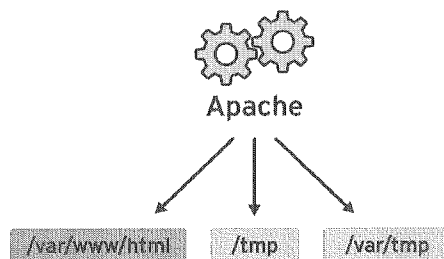
## Objectives

After completing this section, students should be able to:

- Explain the basics of SELinux permissions and context transitions.
- Display the current SELinux mode.
- Correctly interpret the SELinux context of a file.
- Correctly interpret the SELinux context of a process.
- Identify current SELinux Boolean settings.

## Basic SELinux security concepts

Security Enhanced Linux (SELinux) is an additional layer of system security. A 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.



*Figure 5.1: Apache service without SELinux protection*

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, and if they compromise the web server process, gain its permissions: the permissions of the **apache** user and the **apache** group. That user/group has read access to things like the document root (**/var/www/html**), as well as write access to **/tmp**, **/var/tmp**, and any other files/directories that are world-writable.

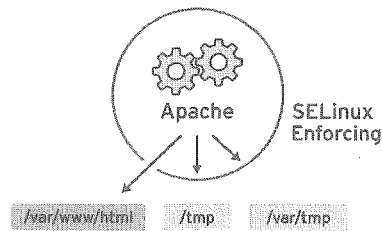


Figure 5.2: Apache service with SELinux protection

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 a SELinux context. A context is a name that is 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`. The type context for the web server is `httpd_t`. The type context for files and directories normally found in `/var/www/html` is `httpd_sys_content_t`. The type 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`.

There is a policy rule that permits Apache (the web server process running as `httpd_t`) to access files and directories with a context normally found in `/var/www/html` and other web server directories (`httpd_sys_content_t`). There is no allow rule in the policy for files normally found in `/tmp` and `/var/tmp`, so access is not permitted. With SELinux, a malicious user could not access the `/tmp` directory. SELinux has rules for remote file systems such as NFS and CIFS, although all files on these file systems are labeled with the same context.

Many commands that deal with files have an option (usually `-Z`) 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@serverX ~]# 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@serverX ~]# systemctl start httpd
[root@serverX ~]# 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@serverX ~]# 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@serverX ~]# 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
```

## SELinux modes

For troubleshooting purposes, SELinux protection can be temporarily disabled using SELinux modes.

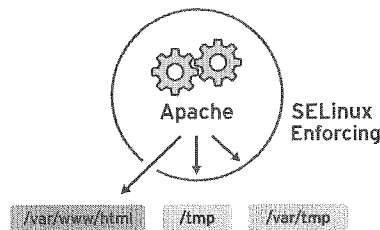


Figure 5.3: SELinux enforcing mode

In *enforcing mode*, SELinux actively denies access to the web server attempting to read files with **tmp\_t** type context. In enforcing mode, SELinux both logs and protects.

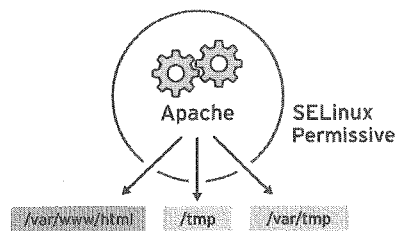


Figure 5.4: SELinux permissive mode

*Permissive mode* is often used to troubleshoot issues. In permissive mode, SELinux allows all interactions, even if there is no explicit rule, and it logs those interactions it would have denied in enforcing mode. This mode can be used to temporarily allow access to content that SELinux is restricting. No reboot is required to go from enforcing to permissive or back again.

A third mode, *disabled*, completely disables SELinux. A system reboot is required to disable SELinux entirely, or to get from disabled mode to enforcing or permissive mode.



### Important

It is better to use permissive mode than to turn off SELinux entirely. One reason for this is that even in permissive mode, the kernel will automatically maintain SELinux file system labels as needed, avoiding the need for an expensive relabeling of the file system when the system is rebooted with SELinux enabled.

To display the current SELinux mode in effect, use the **getenforce** command.

```
[root@serverX ~]# getenforce
```

---

**Enforcing**

## 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 **getsebool** command is used to display SELinux Booleans and their current value. The **-a** option causes this command to list all of the Booleans.

```
[root@serverX ~]# getsebool -a
abrt_anon_write --> off
allow_console_login --> on
allow_corosync_rw_tmpfs --> off
[... Output omitted ...]
```



## References

**selinux(8)**, **getenforce(8)**, **ls(1)**, **ps(1)**, and **getsebool(8)** man pages

## Practice: SELinux Concepts

### Quiz

Match the following items to their counterparts in the table.

Boolean	Context	Disabled mode	Enforcing mode
Permissive mode			

Term	Description
Policy rules are obeyed and violations logged	
Label on processes, files, and ports that determine access	
A reboot is required to transition to this mode	
Switch that enables/disables a set of policy rules	
Policy rule violations only produce log messages	

## Solution

Match the following items to their counterparts in the table.

Term	Description
Policy rules are obeyed and violations logged	Enforcing mode
Label on processes, files, and ports that determine access	Context
A reboot is required to transition to this mode	Disabled mode
Switch that enables/disables a set of policy rules	Boolean
Policy rule violations only produce log messages	Permissive mode

## Changing SELinux Modes

### Objectives

After completing this section, students should be able to:

- Change the current SELinux mode of a system.
- Set the default SELinux mode of a system.

For troubleshooting purposes, SELinux protection can be temporarily disabled using SELinux modes. This section will look at how to change SELinux modes temporarily between enforcing and permissive mode. It will also look at how to set the default SELinux mode that is determined at boot time.

### Changing the current SELinux mode

The **setenforce** command modifies the current SELinux mode:

```
[root@serverX ~]# getenforce
Enforcing
[root@serverX ~]# setenforce
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]
[root@serverX ~]# setenforce 0
[root@serverX ~]# getenforce
Permissive
[root@serverX ~]# setenforce Enforcing
[root@serverX ~]# getenforce
Enforcing
```

Another way to temporarily set the SELinux mode is to pass a parameter to the kernel at boot time. Passing a kernel argument of **enforcing=0** causes the system to boot into permissive mode. A value of **1** would specify enforcing mode. SELinux can be disabled when the **selinux=0** argument is specified. A value of **1** would enable SELinux.

### Setting the default SELinux mode

The configuration file that determines what the SELinux mode is at boot time is **/etc/selinux/config**. Notice that it contains some useful comments:

```
# 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
```

Use **/etc/selinux/config** to change the default SELinux mode at boot time. In the example shown, it is set to enforcing mode.





## Note

In older releases of Red Hat Enterprise Linux, the default SELinux mode was set in a file called **/etc/sysconfig/selinux**. In RHEL7, this file is a symbolic link that points to **/etc/selinux/config**.

Passing the **selinux=** and/or the **enforcing=** kernel arguments overrides any of the default values specified in **/etc/selinux/config**.



## References

**getenforce(1)**, **setenforce(1)**, and **selinux\_config(5)** man pages

## Practice: Changing SELinux Modes

### Guided exercise

In this lab, you will manage SELinux modes, both temporarily and persistently.

#### Resources

**Machines:****serverX****Outcomes:**

You will get practice viewing and setting the current SELinux mode.

- ❑ 1. Log in as **root** on **serverX**. Display the current SELinux mode.

```
[root@serverX ~]# getenforce
Enforcing
```

- ❑ 2. Change the default SELinux mode to permissive and reboot.

```
[root@serverX ~]# vi /etc/selinux/config
[root@serverX ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=permissive
SELINUXTYPE=targeted
[root@serverX ~]# reboot
```

- ❑ 3. When **serverX** comes back up, log in as **root** and display the current SELinux mode.

```
[root@serverX ~]# getenforce
Permissive
```

- ❑ 4. Change the default SELinux mode to enforcing.

```
[root@serverX ~]# vi /etc/selinux/config
[root@serverX ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

- ❑ 5. Set the current SELinux mode to enforcing.

```
[root@serverX ~]# setenforce 1
[root@serverX ~]# getenforce
Enforcing
```

# Changing SELinux Contexts

## Objectives

After completing this section, students should be able to:

- Set the SELinux security context of files in the policy.
- Restore the SELinux security context of files.

## Initial SELinux context

Typically, the SELinux context of a file's parent directory determines its initial SELinux context. The context of the parent directory is assigned to the newly created file. This works for commands like **vim**, **cp**, and **touch**. However, if a file is created elsewhere and the permissions are preserved (as with **mv** or **cp -a**), the original SELinux context will be unchanged.

```
[root@serverX ~]# ls -Zd /var/www/html/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
[root@serverX ~]# touch /var/www/html/index.html
[root@serverX ~]# 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
```

## Changing the SELinux context of a file

There are two commands that are used to change the SELinux context of files: **chcon** and **restorecon**. The **chcon** command changes the context of the file to the context specified as an argument to the command. Often the **-t** option is used to specify only the type component of the context.

The **restorecon** command is the preferred method for changing the SELinux context of a file or directory. Unlike **chcon**, the context is not explicitly specified when using this command. It uses rules in the SELinux policy to determine what the context of the file should be.



### Note

**chcon** should not be used to change the SELinux context of files. Mistakes can be made when specifying the context explicitly. File contexts will be changed back to their default context if the system's file systems are relabeled at boot time.

```
[root@serverX ~]# mkdir /virtual
[root@serverX ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual
[root@serverX ~]# chcon -t httpd_sys_content_t /virtual
[root@serverX ~]# ls -Zd /virtual
drwxr-xr-x. root root unconfined_u:object_r:httpd_sys_content_t:s0 /virtual
[root@serverX ~]# restorecon -v /virtual
restorecon reset /virtual context unconfined_u:object_r:httpd_sys_content_t:s0->
unconfined_u:object_r:default_t:s0
[root@serverX ~]# 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 can be used to display or modify the rules that the **restorecon** command 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.

The **restorecon** command is part of the **policycoreutil** package, and **semanage** is part of the **policycoreutil-python** package.

```
[root@serverX ~]# touch /tmp/file1 /tmp/file2
[root@serverX ~]# ls -Z /tmp/file*
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 /tmp/file1
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
[root@serverX ~]# mv /tmp/file1 /var/www/html/
[root@serverX ~]# cp /tmp/file2 /var/www/html/
[root@serverX ~]# ls -Z /var/www/html/file*
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
-rw-r--r--. root root unconfined_u:object_r:htpd_sys_content_t:s0 /var/www/html/file2
[root@serverX ~]# semanage fcontext -l
...
/var/www(/.*)?      all files    system_u:object_r:htpd_sys_content_t:s0
...
[root@serverX ~]# restorecon -Rv /var/www/
restorecon reset /var/www/html/file1 context unconfined_u:object_r:user_tmp_t:s0
-> system_u:object_r:htpd_sys_content_t:s0
[root@serverX ~]# ls -Z /var/www/html/file*
-rw-r--r--. root root system_u:object_r:htpd_sys_content_t:s0
/var/www/html/file1
-rw-r--r--. root root unconfined_u:object_r:htpd_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@serverX ~]# mkdir /virtual
[root@serverX ~]# touch /virtual/index.html
[root@serverX ~]# ls -Zd /virtual/
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual/
[root@serverX ~]# ls -Z /virtual/
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 index.html
[root@serverX ~]# semanage fcontext -a -t htpd_sys_content_t '/virtual(/.*)?'
[root@serverX ~]# restorecon -RFvv /virtual
[root@serverX ~]# ls -Zd /virtual/
drwxr-xr-x. root root system_u:object_r:htpd_sys_content_t:s0 /virtual/
[root@serverX ~]# ls -Z /virtual/
-rw-r--r--. root root system_u:object_r:htpd_sys_content_t:s0 index.html
```



## References

**chcon(1)**, **restorecon(8)**, and **semanage(8)** man pages

# Practice: Changing SELinux Contexts

## Guided exercise

In this lab, you will persistently change the SELinux context of a directory and its contents.

Resources	
<b>Files:</b>	<code>/etc/httpd/conf/httpd.conf</code>
<b>Machines:</b>	<code>serverX</code>

### Outcomes:

You will have a web server that publishes web content from a non-standard document root.

### Before you begin...

You should have a working RHEL 7 system with SELinux in enforcing mode.

- ❑ 1. Log in as **root** on **serverX**. Use **yum** to install the Apache web server.

```
[root@serverX ~]# yum install -y httpd
```

- ❑ 2. Configure Apache to use a document root in a non-standard location.

- ❑ 2.1. Create the new document root, **/custom**.

```
[root@serverX ~]# mkdir /custom
```

- ❑ 2.2. Create the **index.html** with some recognizable content.

```
[root@serverX ~]# echo 'This is serverX.' > /custom/index.html
```

- ❑ 2.3. Configure Apache to use the new 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@serverX ~]# vi /etc/httpd/conf/httpd.conf
[root@serverX ~]# grep custom /etc/httpd/conf/httpd.conf
DocumentRoot "/custom"
<Directory "/custom">
```

- ❑ 3. Start the Apache web service.

```
[root@serverX ~]# systemctl start httpd
```

- ❑ 4. Open a web browser on **serverX** and try to view the following URL:  
`http://localhost/index.html`. You will get an error message that says you do not have permission to access the file.

- ❑ 5. Define a SELinux file context rule that sets the context type to **httpd\_sys\_content\_t** for **/custom** and all the files below it.

```
[root@serverX ~]# semanage fcontext -a -t httpd_sys_content_t '/custom(/.*)?'
```

- ❑ 6. Use **restorecon** to change their contexts.

```
[root@serverX ~]# restorecon -Rv /custom
restorecon reset /custom context unconfined_u:object_r:default_t:s0-
>unconfined_u:object_r:httpd_sys_content_t:s0
restorecon reset /custom/index.html context unconfined_u:object_r:default_t:s0-
>unconfined_u:object_r:httpd_sys_content_t:s0
```

- ❑ 7. Try to view <http://localhost/index.html> again. You should see the message "This is serverX." displayed.

# Changing SELinux Booleans

## Objectives

After completing this section, students should be able to use SELinux Booleans to make adjustments to policy behavior.

## 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-policy-devel** package provides many manual pages, **\*\_selinux(8)**, which explain the purpose of the Booleans available for various services. If this package has been installed, the **man -k '\_selinux'** command can list these documents.

The **getsebool** command is used to display SELinux Booleans and **setsebool** is used to modify them. **setsebool -P** modifies the SELinux policy to make the modification persistent. **semanage boolean -l** will show whether or not a Boolean is persistent, along with a short description of the Boolean.

```
[root@serverX ~]# 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
...
[root@serverX ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
[root@serverX ~]# setsebool httpd_enable_homedirs on
[root@serverX ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs      (on , off) Allow httpd to enable homedirs
[root@serverX ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
[root@serverX ~]# setsebool -P httpd_enable_homedirs on
[root@serverX ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs      (on , on) Allow httpd to enable homedirs
```

To only list local modifications to the state of the SELinux booleans (any setting that differs from the default in the policy), the command **semanage boolean -l -C** can be used.

```
[root@serverX ~]# 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



# Practice: Changing SELinux Booleans

## Guided exercise

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 will permit Apache to access user home directories.

Resources	
<b>Files:</b>	<code>/etc/httpd/conf.d/userdir.conf</code>
<b>Machines:</b>	<code>serverX</code>

### Outcomes:

You will have a web server that publishes web content from users' home directories.

### Before you begin...

The Apache web server should already be installed and running on `serverX.example.com`.

- ❑ 1. Log in as **root** on **serverX**. Enable the Apache feature that permits users to publish web content from their home directories. Edit the `/etc/httpd/conf.d/userdir.conf` configuration file and change the line with the **UserDir** directive to read as follows:

```
UserDir public_html
```

```
[root@serverX ~]# vi /etc/httpd/conf.d/userdir.conf
[root@serverX ~]# grep '^ *UserDir' /etc/httpd/conf.d/userdir.conf
UserDir public_html
```

- ❑ 2. Restart the Apache web service to make the changes take effect.

```
[root@serverX ~]# systemctl restart httpd
```

- ❑ 3. Create some web content that is published from a user's home directory.

- ❑ 3.1. Log in as **student** in another window and create a **public\_html** directory.

```
[student@serverX ~]$ mkdir ~/public_html
```

- ❑ 3.2. Create some content in a **index.html** file.

```
[student@serverX ~]$ echo 'This is student content on serverX.' > ~/public_html/index.html
```

- ❑ 3.3. Change the permissions on **student**'s home directory so Apache can access the **public\_html** subdirectory.

```
[student@serverX ~]$ chmod 711 ~
```

- ❑ 4. Open a web browser on **serverX** and try to view the following URL: `http://localhost/~student/index.html`. You will get an error message that says you do not have permission to access the file.
- ❑ 5. In your **root** window, use the **getsebool** command to see if there are any Booleans that restrict access to home directories.

```
[root@serverX ~]# getsebool -a | grep home  
[... Output omitted ...]  
httpd_enable_homedirs --> off  
[... Output omitted ...]
```

- ❑ 6. Use **setsebool** to enable home directory access persistently.

```
[root@serverX ~]# setsebool -P httpd_enable_homedirs on
```

- ❑ 7. Try to view `http://localhost/~student/index.html` again. You should see the message "This is student content on serverX."

# Troubleshooting SELinux

## Objectives

After completing this section, students should be able to use SELinux log analysis tools.

## Troubleshooting SELinux issues

What should be done when SELinux prevents access to files on a server? There is a sequence of steps that should be taken when this occurs.

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 isn't 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 a too-restrictive access could be the adjustment of a Boolean. For example, the **ftpd\_anon\_write** Boolean controls whether anonymous FTP users can upload files. This Boolean would have to be turned on if it is desirable to allow 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

The **setroubleshoot-server** package must be installed 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 (**UUIDs**) for SELinux violations that can be used to gather further information. **sealert -l UUID** is used to produce a report for a specific incident. **sealert -a /var/log/audit/audit.log** is used to produce reports for all incidents in that file.

Consider the following sample sequence of commands on a standard Apache web server:

```
[root@serverX ~]# touch /root/file3
[root@serverX ~]# mv /root/file3 /var/www/html
[root@serverX ~]# systemctl start httpd
[root@serverX ~]# 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>
[root@serverX ~]# tail /var/log/audit/audit.log
...
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
...
[root@serverX ~]# tail /var/log/messages
...
Feb 20 19:55:42 serverX 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
[root@serverX ~]# 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                serverX.example.com
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            serverX.example.com
Platform            Linux serverX.example.com 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**, doesn't 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/**.



## References

**sealert(8)** man page

## Practice: Troubleshooting SELinux

### Guided exercise

In this lab, you will learn how to troubleshoot SELinux security denials.

Changing the **DocumentRoot** of an Apache web server introduces SELinux access denials. In this exercise, you will see how that issue could have been identified and resolved.

#### Resources

##### Machines:

serverX

#### Outcomes:

You will get some experience using SELinux troubleshooting tools.

#### Before you begin...

The Apache web server should already be installed and running on serverX.example.com.

You should have completed the steps of the “Changing SELinux Contexts” practice exercise.

- ❑ 1. Log in as **root** on **serverX**. Remove the file context rule created earlier and restore the **/custom** directory structure back to its original SELinux context.
- ❑ 1.1. Remove the file context rule you added in the earlier lab.

```
[root@serverX ~]# semanage fcontext -d -t httpd_sys_content_t '/custom(/.*)?'
```

- ❑ 1.2. Change the file contexts to their original values.

```
[root@serverX ~]# restorecon -Rv /custom
restorecon reset /custom context
unconfined_u:object_r:httpd_sys_content_t:s0
->unconfined_u:object_r:default_t:s0
restorecon reset /custom/index.html context
unconfined_u:object_r:httpd_sys_
content_t:s0->unconfined_u:object_r:default_t:s0
```

- ❑ 2. Open a web browser on **serverX** and try to view the following URL: <http://localhost/index.html>. You will get an error message that says you do not have permission to access the file.
- ❑ 3. View the contents of **/var/log/messages**. You should see some output similar to the following:

```
[root@serverX ~]# tail /var/log/messages
[... Output omitted ...]
Feb 19 12:00:35 serverX setroubleshoot: SELinux is preventing /usr/sbin/httpd
from getattr access on the file . For complete SELinux messages. run
sealert -l 82ead554-c3cb-4664-85ff-e6f256437c6c
[... Output omitted ...]
```

- 4. Run the suggested **sealert** command and see if you can identify the issue and a possible resolution.

```
[root@serverX ~]# sealert -l 82ead554-c3cb-4664-85ff-e6f256437c6c
SELinux is preventing /usr/sbin/httpd from getattr access on the file .

**** Plugin catchall_labels (83.8 confidence) suggests ****

If you want to allow httpd to have getattr access on the file
Then you need to change the label on $FIX_TARGET_PATH
Do
# semanage fcontext -a -t FILE_TYPE '$FIX_TARGET_PATH'
where FILE_TYPE is one of the following: NetworkManager_log_t, ...,
httpd_sys_content_t, httpd_sys_htaccess_t, httpd_sys_ra_content_t,
httpd_sys_rw_content_t, httpd_sys_script_exec_t, httpd_tmp_t, ...
Then execute:
restorecon -v '$FIX_TARGET_PATH'

**** Plugin catchall (17.1 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:default_t:s0
Target Objects          [ file ]
Source                  httpd
Source Path              /usr/sbin/httpd
Port                    <Unknown>
Host                    serverX.example.com
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                serverX.example.com
Platform                Linux serverX.example.com 3.10.0-84.el7.x86_64 #1
                        SMP Tue Feb 4 16:28:19 EST 2014 x86_64 x86_64
Alert Count              9
First Seen              2014-02-19 10:33:06 EST
Last Seen               2014-02-19 12:00:32 EST
Local ID                 82ead554-c3cb-4664-85ff-e6f256437c6c

Raw Audit Messages
type=AVC msg=audit(1392829232.3:1782): avc: denied { getattr } for
pid=11870 comm="httpd" path="/custom/index.html" dev="vda1" ino=11520682
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file

type=SYSCALL msg=audit(1392829232.3:1782): arch=x86_64 syscall=lstat success=no
exit=EACCES a0=7f1854a3b068 a1=7fff493f2ff0 a2=7fff493f2ff0
a3=ffffffffffffffff items=0 ppid=11866 pid=11870 auid=4294967295 uid=48
gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none)
```

```
ses=4294967295 comm=httpd exe=/usr/sbin/httpd
subj=system_u:system_r:httpd_t:s0 key=(null)
```

```
Hash: httpd,httpd_t,default_t,file,getattr
```

- ❑ 5. Read the output from the **sealert** command. Identify which file the Apache web server is having trouble with and look for a possible remedy.

- ❑ 5.1. At the top of the output, a solution is recommended.

```
# semanage fcontext -a -t FILE_TYPE '$FIX_TARGET_PATH'
where FILE_TYPE is one of the following: NetworkManager_log_t, ...,
httpd_sys_content_t, httpd_sys_htaccess_t, httpd_sys_ra_content_t,
httpd_sys_rw_content_t, httpd_sys_script_exec_t, httpd_tmp_t, ...
Then execute:
restorecon -v '$FIX_TARGET_PATH'
```

- ❑ 5.2. Look at the raw AVC message to identify the relevant process and file that is causing the alert.

```
Raw Audit Messages
type=AVC msg=audit(1392829232.3:1782): avc: denied { getattr } for
pid=11870 comm="httpd" path="/custom/index.html" dev="vda1" ino=11520682
scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:default_t:s0 tclass=file
```

- ❑ 5.3. The process involved in the security denial is the **httpd** Apache web server and the file is **/custom/index.html**.
- ❑ 6. Earlier, we resolved this issue using **semanage** and **restorecon**. You must decide if this SELinux violation is a security breach or if it is a legitimate access that requires SELinux to be adjusted to handle a non-standard directory structure.



# 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.

Solve this problem by making adjustments to SELinux. Do not disable SELinux or put it in permissive mode. Do not move the web content or reconfigure Apache in any way.

### Resources

#### Machines:

serverX

### Outcomes:

Launching a web server on **serverX** and pointing it to `http://localhost/lab-content` will display web content instead of an error message.

### Before you begin...

- Reset your **serverX** system.
- Log into and set up your **serverX** system.

```
[student@serverX ~]$ lab selinux setup
```

1. Launch a web browser on **serverX** and browse to `http://localhost/lab-content`. You will see an error message.
2. Research and identify the SELinux issue that is preventing Apache from serving web content.
3. Resolve the SELinux issue that is preventing Apache from serving web content.
4. Verify the SELinux issue has been resolved and Apache is able to serve web content.
5. Run the **lab selinux** grading script to confirm your findings.

## Solution

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.

Solve this problem by making adjustments to SELinux. Do not disable SELinux or put it in permissive mode. Do not move the web content or reconfigure Apache in any way.

### Resources

#### Machines:

serverX

### Outcomes:

Launching a web server on **serverX** and pointing it to `http://localhost/lab-content` will display web content instead of an error message.

### Before you begin...

- Reset your **serverX** system.
- Log into and set up your **serverX** system.

```
[student@serverX ~]$ lab selinux setup
```

1. Launch a web browser on **serverX** and browse to `http://localhost/lab-content`. You will see an error message.
2. Research and identify the SELinux issue that is preventing Apache from serving web content.

Look in `/var/log/messages` for helpful error messages.

```
[root@serverX ~]# tail /var/log/messages
[... Output omitted ...]
Feb 20 13:55:59 serverX dbus-daemon: dbus[427]: [system] Successfully activated service 'org.fedoraproject.Setroubleshootd'
Feb 20 13:55:59 serverX dbus[427]: [system] Successfully activated service 'org.fedoraproject.Setroubleshootd'
Feb 20 13:56:01 serverX setroubleshoot: Plugin Exception restorecon
Feb 20 13:56:01 serverX setroubleshoot: SELinux is preventing /usr/sbin/httpd from open access on the file . For complete SELinux messages. run sealert -l 160daebd-0359-4f72-9dde-46e7fd244e27
```

Especially note the **setroubleshootd** messages. Run **sealert** to get more detailed information about the SELinux error.

```
[root@serverX ~]# sealert -l 160daebd-0359-4f72-9dde-46e7fd244e27
SELinux is preventing /usr/sbin/httpd from open access on the file .

**** Plugin catchall_boolean (89.3 confidence) suggests ****

If you want to allow httpd to read user content
Then you must tell SELinux about this by enabling the 'httpd_read_user_content'
boolean.
You can read 'None' man page for more details.
Do
setsebool -P httpd_read_user_content 1
```

```

**** Plugin catchall (11.6 confidence) suggests ****

If you believe that httpd should be allowed open 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:user_tmp_t:s0
Target Objects                [ file ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          <Unknown>
Host                          serverX.example.com
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                     serverX.example.com
Platform                      Linux serverX.example.com 3.10.0-84.el7.x86_64 #1
                               SMP Tue Feb 4 16:28:19 EST 2014 x86_64 x86_64

Alert Count                   1
First Seen                    2014-02-20 13:55:56 EST
Last Seen                     2014-02-20 13:55:56 EST
Local ID                       160daebd-0359-4f72-9dde-46e7fd244e27

Raw Audit Messages
type=AVC msg=audit(1392922556.862:494): avc: denied { open } for pid=24492
comm="httpd" path="/var/web-content/lab-content/index.html" dev="vda1"
ino=29062705 scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:user_tmp_t:s0 tclass=file

type=SYSCALL msg=audit(1392922556.862:494): arch=x86_64 syscall=open success=no
exit=EACCES a0=7fda4c92eb40 a1=80000 a2=0 a3=0 items=0 ppid=24487 pid=24492
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,user_tmp_t,file,open

```

Looking closely at the raw audit messages, you see that Apache cannot access **/var/web-content/lab-content/index.html**.

3. Resolve the SELinux issue that is preventing Apache from serving web content.

**/var/web-content** is a nonstandard location for Apache web content. Display the SELinux context of **/var/web-content** and the standard document root, **/var/www/html**.

```

[root@serverX ~]# ls -ld -Z /var/web-content /var/www/html
drwxr-xr-x. root root unconfined_u:object_r:var_t:s0 /var/web-content
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html

```

Create a file context rule that will set the default type to **httpd\_sys\_content\_t** for **/var/web-content** and all files below it.

```
[root@serverX ~]# semanage fcontext -a -t httpd_sys_content_t '/var/web-content(/.*)?'
```

Use the **restorecon** command to set the SELinux context for the files in **/var/web-content**.

```
[root@serverX ~]# restorecon -R /var/web-content/
```

4. Verify the SELinux issue has been resolved and Apache is able to serve web content.

Use your web browser to refresh the <http://localhost/lab-content> link. Now you should see some web content.

```
This is the content for the SELinux chapter test.
```

5. Run the **lab selinux** grading script to confirm your findings.

```
[root@serverX ~]# lab selinux grade
Confirming SELinux is in enforcing mode...PASS
Confirming files are in expected location...PASS
Confirming the Apache DocumentRoot is unchanged...PASS
Confirming the web content is accessible...PASS
```

## Summary

### Enabling and Monitoring Security Enhanced Linux (SELinux)

- **getenforce** displays the current SELinux mode, which determines whether SELinux rules are applied.
- The **-Z** option to **ls** and **ps** displays SELinux context labels on files and processes.
- **getsebool -a** displays all SELinux Booleans and their current value.

### Changing SELinux Modes

- **setenforce** changes the current SELinux mode of a system.
- The default SELinux mode of a system is defined in the **/etc/selinux/config** file.

### Changing SELinux Contexts

- The **semanage fcontext** command is used to manage SELinux policy rules that determine the default context for files and directories.
- **restorecon** applies the context defined by the SELinux policy to files and directories.
- Although the **chcon** command can change the SELinux context files, it shouldn't be used because the change may not persist.

### Changing SELinux Booleans

- **setsebool** activates/deactivates SELinux policy rules.
- **semanage boolean -l** displays the persistent value of SELinux Booleans.
- Man pages that end with **\_selinux** often provide useful information about SELinux Booleans.

### Troubleshooting SELinux

- **setroubleshootd** generates log messages in **/var/log/messages**.
- The **sealert** command displays useful information that helps with SELinux troubleshooting.





## CHAPTER 6

# PROCESS MANAGEMENT

Overview	
<b>Goal</b>	To evaluate and control processes running on a Red Hat Enterprise Linux system.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Terminate and control processes using signals.</li><li>• Monitor resource usage and system load due to process activity.</li><li>• Set nice levels on new and existing processes.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Killing Processes (and Practice)</li><li>• Monitoring Process Activity (and Practice)</li><li>• Using nice and renice to Influence Process Priority (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Managing Priority of Linux Processes</li></ul>

## Killing Processes

### Objectives

After completing this section, students 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* (e.g., I/O request or expired timer), or by *explicit request* (e.g., use of a signal-sending command or by 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 typing <b>INTR</b> character ( <b>Ctrl-c</b> ).
3	QUIT	Keyboard quit	Similar to <b>SIGINT</b> , but also produces a process dump at termination. Sent by typing <b>QUIT</b> character ( <b>Ctrl-\</b> ).
9	KILL	Kill, unblockable	Causes abrupt program termination. Cannot be blocked, ignored, or handled; always fatal.
15 <i>default</i>	TERM	Terminate	Causes program termination. Unlike <b>SIGKILL</b> , 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 <b>SIGSTOP</b> , can be blocked, ignored, or handled. Sent by typing <b>SUSP</b> character ( <b>Ctrl-z</b> ).





## 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 Intel x86 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 for expected event signals by implementing handler routines to ignore, replace, or extend a signal's default action.

### Commands for sending signals by explicit request

Users signal their current foreground process by typing a keyboard control sequence to suspend (**Ctrl-z**), kill (**Ctrl-c**), or core dump (**Ctrl-\**) the process. To signal a background process or processes in a different session requires a signal-sending command.

Signals can be specified either by name (e.g., **-HUP** or **-SIGHUP**) or by number (e.g., **-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 ID. Despite its name, the kill command can be used for sending any signal, not just those for terminating programs.

```
[student@serverX ~]$ kill PID
[student@serverX ~]$ kill -signal PID
[student@serverX ~]$ 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 truncated --
```

- Use **killall** to send a signal to one or more processes matching selection criteria, such as a command name, processes owned by a specific user, or all system-wide processes.

```
[student@serverX ~]$ killall command_pattern
[student@serverX ~]$ killall -signal command_pattern
[root@serverX ~]# killall -signal -u username command_pattern
```

- The **pkill** command, like **killall**, can signal multiple processes. **pkill** uses advanced selection criteria, which can include combinations of:
  - **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.

```
[student@serverX ~]$ pkill command_pattern
[student@serverX ~]$ pkill -signal command_pattern
[root@serverX ~]# pkill -G GID command_pattern
[root@serverX ~]# pkill -P PPID command_pattern
[root@serverX ~]# pkill -t terminal_name -U UID command_pattern
```

## Logging users out administratively

The **w** command views users currently logged into the system and their cumulative activities. Use the **TTY** and **FROM** columns to determine the user's location.

All users have a controlling terminal, listed as **pts/N** while working in a graphical environment window (*pseudo-terminal*) or **ttyN** on a system console, alternate console, or other directly connected terminal device. Remote users display their connecting system name in the **FROM** column when using the **-f** option.

```
[student@serverX ~]$ w -f
12:43:06 up 27 min,  5 users,  load average: 0.03, 0.17, 0.66
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
student   :0        :0               12:20    ?xdm?  1:10   0.16s gdm-session-wor
student   pts/0     :0               12:20    2.00s  0.08s  0.01s w -f
root      tty2      :0               12:26    14:58  0.04s  0.04s -bash
bob       tty3      :0               12:28    14:42  0.02s  0.02s -bash
student   pts/1     desktop2.example. 12:41    1:07   0.03s  0.03s -bash
[student@serverX ~]$
```

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, included background tasks and children, are in the **JCPU** column. Current foreground process CPU consumption is in the **PCPU** column.

Users may be forced off a system for security violations, resource overallocation, or other administrative need. Users are expected to quit unnecessary applications, close unused command shells, and exit login sessions when requested.

When situations occur in which users cannot be contacted or have unresponsive sessions, runaway resource consumption, or improper system access, their sessions may need to be administratively terminated using signals.



### Important

Although **SIGTERM** is the default signal, **SIGKILL** is a commonly misused administrator favorite. 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 retry with **SIGKILL** only if a process fails to respond.

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.

```
[root@serverX ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@serverX ~]# pkill -SIGKILL -u bob
[root@serverX ~]# pgrep -l -u bob
[root@serverX ~]#
```

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 which reference the same terminal ID. Unless **SIGKILL** is specified, the session leader (here, the **bash** login shell) successfully handles and survives the termination request, but all other session processes are terminated.

```
[root@serverX ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@serverX ~]# w -h -u bob
bob    tty3      18:37   5:04   0.03s  0.03s  -bash
[root@serverX ~]# pkill -t tty3
[root@serverX ~]# pgrep -l -u bob
7391 bash
[root@serverX ~]# pkill -SIGKILL -t tty3
[root@serverX ~]# pgrep -l -u bob
[root@serverX ~]#
```

The same selective process termination can be applied using parent and child process relationships. Use the **ps tree** 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@serverX ~]# pstree -p bob
bash(8391)─sleep(8425)
           └sleep(8426)
             └sleep(8427)
[root@serverX ~]# pkill -P 8391
[root@serverX ~]# pgrep -l -u bob
bash(8391)
[root@serverX ~]# pkill -SIGKILL -P 8391
[root@serverX ~]# pgrep -l -u bob
bash(8391)
[root@serverX ~]#
```



## 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

# Practice: Killing Processes

## Guided exercise

In this lab, students will use keyboard sequences and signals to manage and stop processes.

### Outcomes:

Experience with observing the results of starting and stopping multiple shell processes.

### Before you begin...

Log in as student to serverX. Start in your home directory.

- ❑ 1. Open two terminal windows, side by side, to be referred to as *left* and *right*.
- ❑ 2. In the left window, start three processes that append text to an output file at one-second intervals. To properly background each process, the complete command set must be contained in parentheses and ended with an ampersand.

```
[student@serverX ~]$ (while true; do echo -n "game " >> ~/outfile; sleep 1; done)
&
[student@serverX ~]$ (while true; do echo -n "set " >> ~/outfile; sleep 1; done)
&
[student@serverX ~]$ (while true; do echo -n "match " >> ~/outfile; sleep 1;
done) &
```

- ❑ 3. In the right window, use **tail** to confirm that all three processes are appending to the file. In the left window, view **jobs** to see all three processes "Running".

```
[student@serverX ~]$ tail -f ~/outfile
[student@serverX ~]$ jobs
[1]  Running                  ( while true; do
    echo -n "game " >> ~/outfile; sleep 1;
done ) &
[2]-  Running                  ( while true; do
    echo -n "set " >> ~/outfile; sleep 1;
done ) &
[3]+  Running                  ( while true; do
    echo -n "match " >> ~/outfile; sleep 1;
done ) &
```

- ❑ 4. Suspend the "game" process using signals. Confirm that the "game" process is "Stopped". In the right window, confirm that "game" output is no longer active.

```
[student@serverX ~]$ kill -SIGSTOP %number
[student@serverX ~]$ jobs
```

- ❑ 5. Terminate the "set" process using signals. Confirm that the "set" process has disappeared. In the right window, confirm that "set" output is no longer active.

```
[student@serverX ~]$ kill -SIGTERM %number
[student@serverX ~]$ jobs
```

- 6. Resume the "game" process using signals. Confirm that the "game" process is "Running". In the right window, confirm that "game" output is again active.

```
[student@serverX ~]$ kill -SIGCONT %number  
[student@serverX ~]$ jobs
```

- 7. Terminate the remaining two jobs. Confirm that no jobs remain and that output has stopped. From the left window, terminate the right window's **tail** command.

Close extra terminal windows.

```
[student@serverX ~]$ kill -SIGTERM %number  
[student@serverX ~]$ kill -SIGTERM %number  
[student@serverX ~]$ jobs  
[student@serverX ~]$ pkill -SIGTERM tail  
[student@serverX ~]$
```

# Monitoring Process Activity

## Objectives

After completing this section, students should be able to:

- Interpret uptime and load averages.
- Monitor real-time processes.

## Load average

The Linux kernel calculates a *load average* metric as an *exponential moving average* of the *load number*, a cumulative CPU count of active system resource requests.

- *Active requests* are counted from per-CPU queues for running threads and threads waiting for I/O, as the kernel tracks process resource activity and corresponding process state changes.
- *Load number* is a calculation routine run every five seconds by default, which accumulates and averages the active requests into a single number for all CPUs.
- *Exponential moving average* is a mathematical formula to smooth out trending data highs and lows, increase current activity significance, and decrease aging data quality.
- *Load average* is the load number calculation routine result. Collectively, it refers to the three displayed values of system activity data averaged for the last 1, 5, and 15 minutes.

### Understanding the Linux load average calculation

The load average represents the perceived system load over a time period. Linux implements the load average calculation as a representation of expected service wait times, not only for CPU but also for disk and network I/O.

- Linux counts not only processes, but threads individually, as separate tasks. CPU request queues for running threads (*nr\_running*) and threads waiting for I/O resources (*nr\_iowait*) reasonably correspond to process states **R** (*Running*) and **D** (*Uninterruptable Sleeping*). Waiting for I/O includes tasks sleeping for expected disk and network responses.
- The load number is a global counter calculation, which is sum-totaled for all CPUs. Since tasks returning from sleep may reschedule to different CPUs, accurate per-CPU counts are difficult, but an accurate cumulative count is assured. Displayed load averages represent all CPUs.
- Linux counts each physical CPU core and microprocessor hyperthread as separate execution units, logically represented and referred to as individual CPUs. Each CPU has independent request queues. View **/proc/cpuinfo** for the kernel representation of system CPUs.

```
[student@serverX ~]$ grep "model name" /proc/cpuinfo
model name : Intel(R) Core(TM) i5 CPU          M 520  @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU          M 520  @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU          M 520  @ 2.40GHz
model name : Intel(R) Core(TM) i5 CPU          M 520  @ 2.40GHz
[student@serverX ~]$ grep "model name" /proc/cpuinfo | wc -l
4
```

- Some UNIX systems only considered CPU utilization or run queue length to indicate system load. Since a system with idle CPUs can experience extensive waiting due to busy disk or network resources, I/O consideration is included in the Linux load average. When experiencing high load averages with minimal CPU activity, examine the disk and network activity.

#### Interpreting displayed load average values

The three values represent the weighted values over the last 1, 5, and 15 minutes. A quick glance can indicate whether system load appears to be increasing or decreasing. Calculate the approximate *per-CPU* load value to determine whether the system is experiencing significant waiting.

- **top**, **uptime**, **w**, and **gnome-system-monitor** display load average values.

```
[student@serverX ~]$ uptime
15:29:03 up 14 min,  2 users,  load average: 2.92, 4.48, 5.20
```

- 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 service waiting times.

```
# From /proc/cpuinfo, 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 ready and waiting thread adds a count of 1. With a total queue count of 1, the resource (CPU, disk, or network) is in use, but no requests spend time waiting. Additional requests increment the count, but since many requests can be processed within the time period, resource *utilization* increases, but not *wait times*.
- Processes sleeping for I/O due to a busy disk or network resource are included in the count and increase the load average. While not an indication of CPU utilization, the queue count still indicates that users and programs are waiting for resource services.
- Until resource saturation, a load average will remain below 1, since tasks will seldom be found waiting in queue. Load average only increases when resource saturation causes requests to remain queued and counted by the load calculation routine. When resource utilization approaches 100%, each additional request starts experiencing service wait time.

## Real-time process monitoring

The **top** 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** = Uninterruptable 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
<b>? or h</b>	Help for interactive keystrokes.
<b>l, t, m</b>	Toggles for load, threads, and memory header lines.
<b>1</b>	Toggle showing individual CPUs or a summary for all CPUs in header.
<b>s</b> <sup>(1)</sup>	Change the refresh (screen) rate, in decimal seconds (e.g., 0.5, 1, 5).
<b>b</b>	Toggle reverse highlighting for <i>Running</i> processes; default is bold only.
<b>B</b>	Enables use of bold in display, in the header, and for <i>Running</i> processes.
<b>H</b>	Toggle threads; show process summary or individual threads.
<b>u, U</b>	Filter for any user name (effective, real).
<b>M</b>	Sorts process listing by memory usage, in descending order.
<b>P</b>	Sorts process listing by processor utilization, in descending order.
<b>k</b> <sup>(1)</sup>	Kill a process. When prompted, enter <b>PID</b> , then <b>signal</b> .
<b>r</b> <sup>(1)</sup>	Renice a process. When prompted, enter <b>PID</b> , then <b>nice_value</b> .
<b>w</b>	Write (save) the current display configuration for use at the next <b>top</b> restart.
<b>q</b>	Quit.
Note:	<sup>(1)</sup> Not available if <b>top</b> started in secure mode. See <b>top(1)</b> .



## References

GNOME System Monitor

- **yelp help:gnome-system-monitor**

**ps(1)**, **top(1)**, **uptime(1)**, and **w(1)** man pages

# Practice: Monitoring Process Activity

## Guided exercise

In this lab, students will use the **top** command to dynamically view, sort, and stop processes.

### Outcomes

Practice with managing processes in real time.

### Before you begin...

Perform the following tasks as **student** on the serverX machine. Run **lab process101 setup** on serverX to prepare for this exercise.

```
[student@serverX ~]$ lab process101 setup
```

- ❑ 1. Open two terminal windows, side by side, to be referred to as *left* and *right*. In the right terminal, run the **top** utility. Size the window to be as tall as possible.

```
[student@serverX ~]$ top
```

- ❑ 2. In the left terminal, determine the number of logical CPUs on this virtual machine.

```
[student@serverX ~]$ grep "model name" /proc/cpuinfo | wc -l
1
```

- ❑ 3. In the left terminal, run a single instance of the **process101** executable.

```
[student@serverX ~]$ process101
```

- ❑ 4. In the right terminal, 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.
- ❑ 5. Note the process ID (PID) for **process101**. View the CPU percentage for the process, which is expected to hover around 25% or 30%.

View the load averages. On a single-CPU virtual machine, for example, 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.

- ❑ 6. In the left terminal, run a second instance of **process101**.

```
[student@serverX ~]$ process101
```

- ❑ 7. In **top**, note the process ID (PID) for the second **process101**. View the CPU percentage for the process, also expected to hover around 25% or 30%.

View the one-minute load average again, which may still be less than 1. Wait up to one minute to allow the calculation to adjust to the new workload.

- ❑ 8. In the left terminal, run a third instance of **process101**.

```
[student@serverX ~]$ process101
```

- ❑ 9. In **top**, note the process ID (PID) for the third **process101**. View the CPU percentage for the process, again expected to hover around 25% or 30%.

View the one-minute load average again, which now is expected to be above 1. Wait up to one minute to allow the calculation to again adjust to the new workload.

- ❑ 10. *Optional:* If this virtual machine has more than one logical CPU, slowly start additional **process101** instances until the one-minute load average equals or exceeds the number of logical CPUs. Divide the load average value by the number of CPUs to determine the estimated load average per CPU.
- ❑ 11. When finished observing the load average values, terminate each of the **process101** processes from within **top**.

- ❑ 11.1. Press **k**. Observe the prompt below the headers and above the columns.

- ❑ 11.2. Type the PID for one of the **process101** instances. Press **Enter**.

- ❑ 11.3. Press **Enter** again to use the default **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.

- ❑ 12. Repeat the previous step for each remaining **process101** instance. Confirm that no **process101** instances remain in **top**.
- ❑ 13. In the right window, press **q** to exit **top**. Close extra terminal windows.

# Using nice and renice to Influence Process Priority

## Objectives

After completing this section, students should be able to:

- Launch processes with a nice level set.
- Modify the nice level on a running process.
- Report on nice levels for processes.

## Reporting on nice levels

The nice levels for existing processes can be viewed in a number of different ways. Most process management tools (like **gnome-system-monitor**) already display the nice level by default, or can be configured to display the nice level.

### Displaying nice levels with top

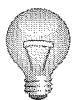
The **top** command can be used to interactively view (and manage) processes. In a default configuration, **top** will display two columns of interest to the nice level: **NI** with the actual nice level, and **PR**, which displays the nice level as mapped to a larger priority queue, with a nice level of **-20** mapping to a priority of **0** and a nice level of **+19** mapping to a priority of **39**.

### Displaying nice levels with ps

The **ps** command can also display nice levels for processes, although it does not do so in most of its default output formats. Users can request exactly the columns they want from **ps**, however, and the name for the nice field is **nice**.

The following example requests a list of all processes, with their pid, name, and nice level, sorted in descending order by nice level:

```
[student@desktopX ~]$ ps axo pid,comm,nice --sort=-nice
PID COMMAND      NI
  74 khugepaged    19
 688 alsactl       19
1953 tracker-miner-f 19
   73 ksmd         5
  714 rtkit-daemon   1
```



## Important

Some processes might report a **-** as their nice level. These processes are being run with a different scheduling policy, and will almost certainly be considered a higher priority by the scheduler. It is possible to display the scheduler policy by requesting the **cls** field from **ps**. A **TS** in this field indicates the process is run under **SCHED\_NORMAL** and can use nice levels; anything else means a different scheduler policy is being used.

## Launching processes with a different nice level

Whenever a process is started, it will normally inherit the nice level from its parent. This means that when a process is started from the command line, it will get the same nice level as the shell process that it was started from. In most cases, this will result in new processes running with a nice level of **0**.

To start a process with a different nice level, both users and system administrators can run their commands using the **nice** tool. Without any other options, running **nice <COMMAND>** will start **<COMMAND>** with a nice level of **10**. Other nice levels can be selected by using the **-n <NICELEVEL>** option to the **nice** command. For example, to start the command **dogecoinminer** with a nice level of **15** and send it to the background immediately, the following command can be used:

```
[student@desktopX ~]$ nice -n 15 dogecoinminer &
```



### Important

Unprivileged users are only allowed to set a positive nice level (**0** to **19**). Only **root** can set a negative nice level (**-20** to **-1**).

## Changing the nice level of an existing process

The nice level of an existing process can be changed from the command line using the **renice** command. The syntax for the **renice** command is as follows:

```
renice -n <NICELEVEL> <PID>...
```

For example, to change the nice level of all **origami@home** processes to **-7**, a system administrator could use the following command (note that more than one PID can be specified at once):

```
[root@desktopX ~]# renice -n -7 $(pgrep origami@home)
```



### Important

Regular users are only allowed to *raise* the nice level on their processes. Only **root** can use **renice** to lower the nice level.

The **top** command can also be used to (interactively) change the nice level on a process. From within **top**, press **r**, followed by the PID to be changed and the new nice level.



## References

**nice(1)**, **renice(1)**, and **top(1)** man pages

# Practice: Discovering Process Priorities

## Guided exercise

In this exercise, you will experience the influence that nice levels have on relative process priorities.

Resources	
<b>Machines:</b>	desktopX

### Outcomes:

An interactive tour of the effects of nice levels.

### Before you begin...

None

- ☐ 1. Log in as **student** to your **desktopX** system.
- ☐ 2. Using the special file **/proc/cpuinfo**, determine the number of CPU cores in your **desktopX** system, then start *two* instances of the command **sha1sum /dev/zero &** for each core.
  - ☐ 2.1. To determine the number of cores using **/proc/cpuinfo**:

```
[student@desktopX ~]$ NCORES=$( grep -c '^processor' /proc/cpuinfo )
```

- ☐ 2.2. Either manually or with a script, start two **sha1sum /dev/zero &** commands for every core in your system.



### Note

The **seq** command prints a list of numbers.

```
[student@desktopX ~]$ for I in $( seq $((NCORES*2)) )
> do
>   sha1sum /dev/zero &
> done
```

- ☐ 3. Verify that you have all the background jobs running that you expected (two for every core in your system).

- ☐ 3.1.

```
[student@desktopX ~]$ jobs
[1]-  Running                  sha1sum /dev/zero &
[2]+  Running                  sha1sum /dev/zero &
...
```

- ☐ 4. Inspect the CPU usage (as a percentage) of all your **sha1sum** processes, using the **ps** and **pgrep** commands. What do you notice?

- ❑ 4.1. 

```
[student@desktopX ~]$ ps u $(pgrep sha1sum)
```
- ❑ 4.2. The CPU percentage for all **sha1sum** processes is about equal.
- ❑ 5. Use the **killall** command to terminate all your **sha1sum** processes.
- ❑ 5.1. 

```
[student@desktopX ~]$ killall sha1sum
```
- ❑ 6. Start two **sha1sum /dev/zero &** commands for each of your cores, but give exactly one of them a nice level of **10**.
- ❑ 6.1. 

```
[student@desktopX ~]$ for I in $( seq $((NCORES*2-1)) )  
> do  
> sha1sum /dev/zero &  
> done  
[student@desktopX ~]$ nice -n10 sha1sum /dev/zero&
```
- ❑ 7. Using the **ps** command, inspect the CPU usage of your **sha1sum** commands. Make sure you include the nice level in your output, as well as the PID and the CPU usage. What do you notice?
- ❑ 7.1. 

```
[student@desktopX ~]$ ps -opid,pcpu,nice,comm $(pgrep sha1sum)
```
- ❑ 7.2. The instance of **sha1sum** with the nice level of **10** gets significantly less CPU than the other instance(s).
- ❑ 8. Use the **renice** command to set the nice level of the **sha1sum** with a nice level of **10** down to **5**. The PID should still be visible in the output of the previous step.  
  
Did this work? Why not?
- ❑ 8.1. 

```
[student@desktopX ~]$ renice -n 5 <PID>  
renice: failed to set priority for <PID> (process ID): Permission denied
```
- ❑ 8.2. Unprivileged users are not allowed to set negative nice values or lower the nice value on an existing process.
- ❑ 9. Using the **sudo** and **renice** commands, set the nice level for the process you identified in the previous step to **-10**.
- ❑ 9.1. 

```
[student@desktopX ~]$ sudo renice -n -10 <PID>
```
- ❑ 10. Start the **top** command as **root**, then use **top** to lower the nice level for the **sha1sum** process using the most CPU back down to **0**. What do you observe afterwards?
- ❑ 10.1. 

```
[student@desktopX ~]$ sudo top
```
- ❑ 10.2. Identify the **sha1sum** process using the most CPU. It will be near the top of the screen.



- ☐ 10.3. Press **r** to enter *renice* mode, then enter the PID you identified, or press **Enter** if the offered default PID is the one you want.
- ☐ 10.4. Enter **0**, then press **Enter**.
- ☐ 10.5. All **sha1sum** commands are again using an (almost) equal amount of CPU.
- ☐ 11. **Important:** Clean up by exiting **top** and killing all your **sha1sum** processes.
  - ☐ 11.1. Press **q** to exit **top**.
  - ☐ 11.2. 

```
[student@desktopX ~]$ killall sha1sum
```

## Lab: Managing Priority of Linux Processes

### Performance checklist

In this lab, you will search for processes with high CPU consumption and adjust their nice levels.

Resources	
<b>Files:</b>	<code>/usr/local/bin/lab nice</code>
<b>Machines:</b>	<code>desktopX</code>

#### Outcomes:

The nice level of the top CPU consumers adjusted to play well with others.

#### Before you begin...

- Reset your **desktopX** system.
- Log into and set up your **desktopX** system.

```
[student@desktopX ~]$ lab nice setup
```

1. Using either **top** or **ps**, identify the two top CPU consumers on your **desktopX** system. If **gnome-shell** is among the top two, ignore it and take the next highest process. Make sure to note the process IDs of these two processes.
2. From the command line, set the nice level of the processes you found in the previous step to **10**.
3. Grade your work by running the following command:

```
[student@desktopX ~]$ lab nice grade
```

4. **Important cleanup:** When you have successfully graded your work, clean up by running the following command:

```
[student@desktopX ~]$ lab nice clean
```

## Solution

In this lab, you will search for processes with high CPU consumption and adjust their nice levels.

Resources	
<b>Files:</b>	/usr/local/bin/lab_nice
<b>Machines:</b>	desktopX

### Outcomes:

The nice level of the top CPU consumers adjusted to play well with others.

### Before you begin...

- Reset your **desktopX** system.
- Log into and set up your **desktopX** system.

```
[student@desktopX ~]$ lab_nice_setup
```

1. Using either **top** or **ps**, identify the two top CPU consumers on your **desktopX** system. If **gnome-shell** is among the top two, ignore it and take the next highest process. Make sure to note the process IDs of these two processes.

- 1.1. Either run **top** and note the two top processes, or run the following:

```
[student@desktopX ~]$ ps aux --sort=pcpu
```

When using the **ps** version, the top CPU consumers will be on the bottom, with their PID listed in the second column.

2. From the command line, set the nice level of the processes you found in the previous step to **10**.

- 2.1. 

```
[student@desktopX ~]$ sudo renice -n 10 <PROCESSPID1> <PROCESSPID2>
```

Make sure to replace **<PROCESSPID1>** and **<PROCESSPID2>** with the process IDs you identified in the previous step.

3. Grade your work by running the following command:

```
[student@desktopX ~]$ lab_nice_grade
```

4. **Important cleanup:** When you have successfully graded your work, clean up by running the following command:

```
[student@desktopX ~]$ lab_nice_clean
```

## Summary

### Killing Processes

Use signals to stop, start, and reload processes and process configurations.

### Monitoring Process Activity

Manage system workload by utilizing load averages and process statistics.

### Using nice and renice to Influence Process Priority

- **nice** is used to set the nice level for new processes.
- **renice** and **top** can be used to modify the nice level on an existing process.
- Both **ps** and **top** can be used to report on nice levels.



## CHAPTER 7

# UPDATING SOFTWARE PACKAGES

Overview	
<b>Goal</b>	To download, install, update, and manage software packages from Red Hat and YUM package repositories.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Register systems with your Red Hat account and entitle them to software updates for installed products.</li><li>• Find, install, and update software packages using the yum command.</li><li>• Enable and disable use of Red Hat or third-party YUM repositories.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Attaching Systems to Subscriptions for Software Updates (and Practice)</li><li>• Managing Software Updates with yum (and Practice)</li><li>• Enabling yum Software Repositories (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Installing and Updating Software Packages</li></ul>

# Attaching Systems to Subscriptions for Software Updates

## Objectives

Register systems with your Red Hat account and entitle them to software updates for installed products.

## 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 which 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).

### Register a system

To register a system with the subscription management service, launch **subscription-manager-gui** by selecting **Applications > Other > Red Hat Subscription Manager** from the main GNOME menu. Enter the password for *root* when prompted to authenticate. This will display the following **Subscription Manager** window.

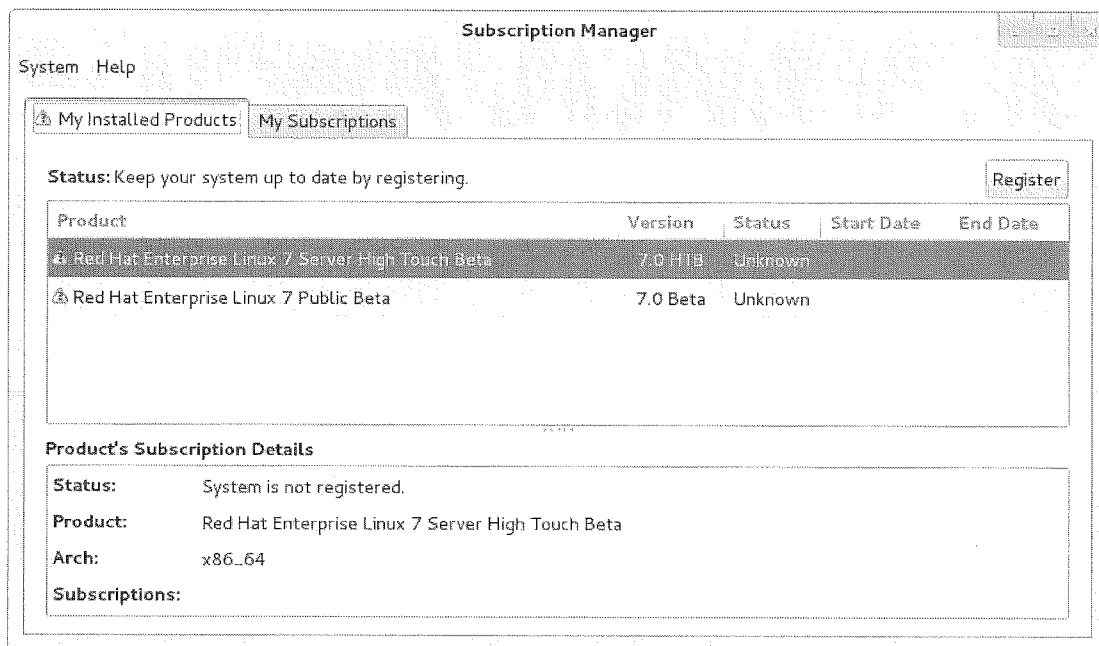


Figure 7.1: The main window of Red Hat Subscription Manager

To register the system, click the **Register** button in the top-right corner of the **Subscription Manager** window. This will bring up the following dialog:

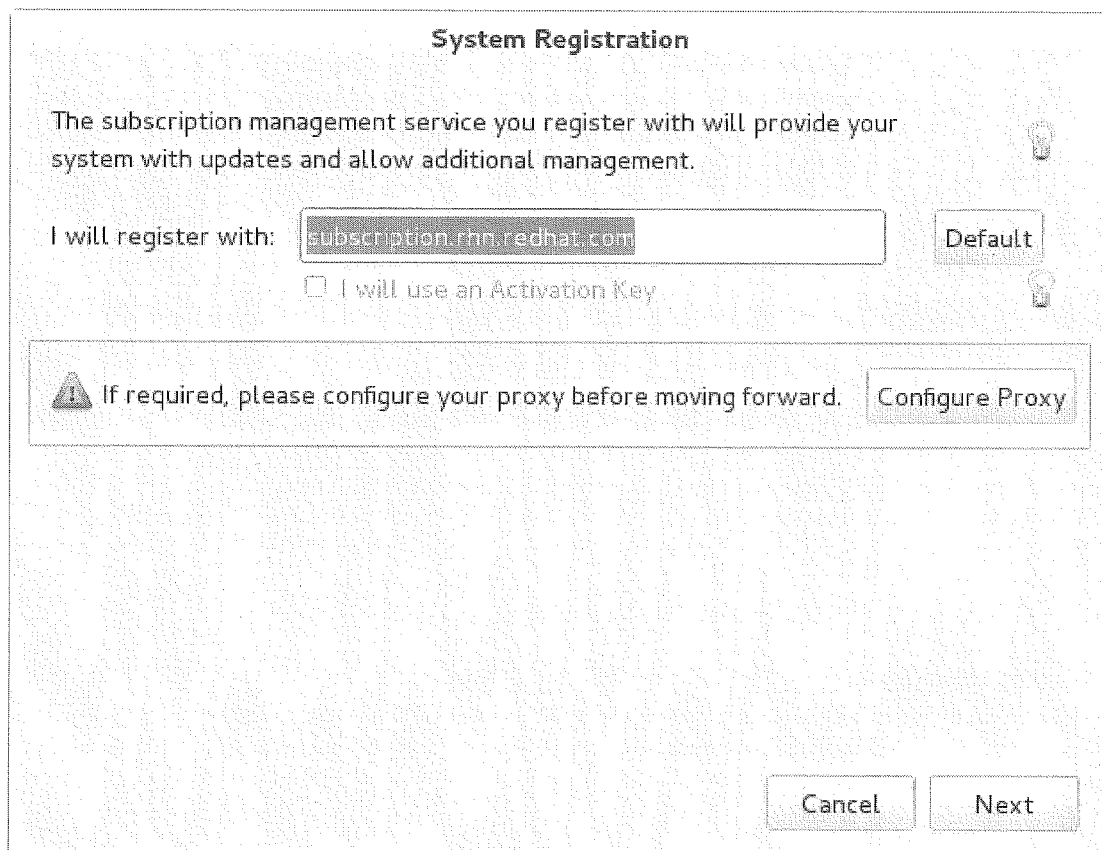


Figure 7.2: The service location dialog of Red Hat Subscription Manager

This dialog box registers a system with a subscription server. The default (subscription.rhn.redhat.com) will register the server to Red Hat's "hosted" content distribution network.

Figure 7.3: The account information dialog of Red Hat Subscription Manager

Click **Next**, then authenticate using the Red Hat account to which the system should be registered.

By default, **Subscription Manager** will try to find the best subscription for this system out of all available subscriptions. If more than one subscription is available, or a specific subscription is required, check the **Manually attach subscriptions after registration** checkbox. With this option checked, **Subscription Manager** will only register the system and not automatically assign any subscriptions.

Click the **Register** button to complete the registration.

### Assigning subscriptions

To assign subscriptions to a system, navigate to the **All Available Subscriptions** tab in the main window of **Subscription Manager**, then click the **Update** button to retrieve a list of available subscriptions.



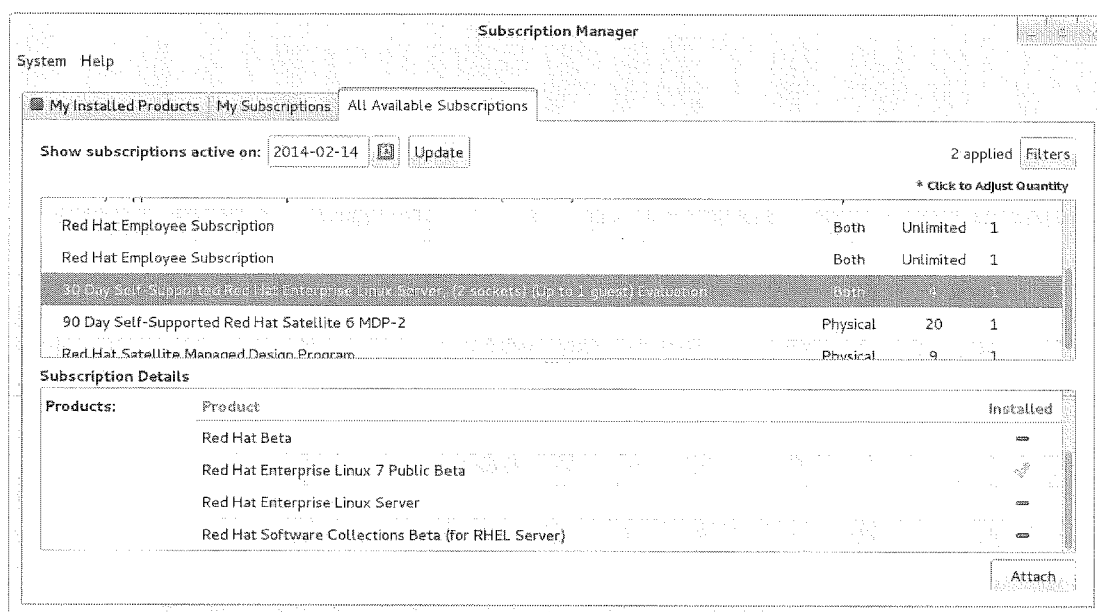


Figure 7.4: All Available Subscriptions tab of Red Hat Subscription Manager

From this list, select one or more subscriptions to assign to this system, then click the **Attach** button.

If there is more than one contract for a specific subscription, a new dialog will open up, asking you to select which contract to use. Note that there are different contracts for *Physical* and *Virtual* systems.

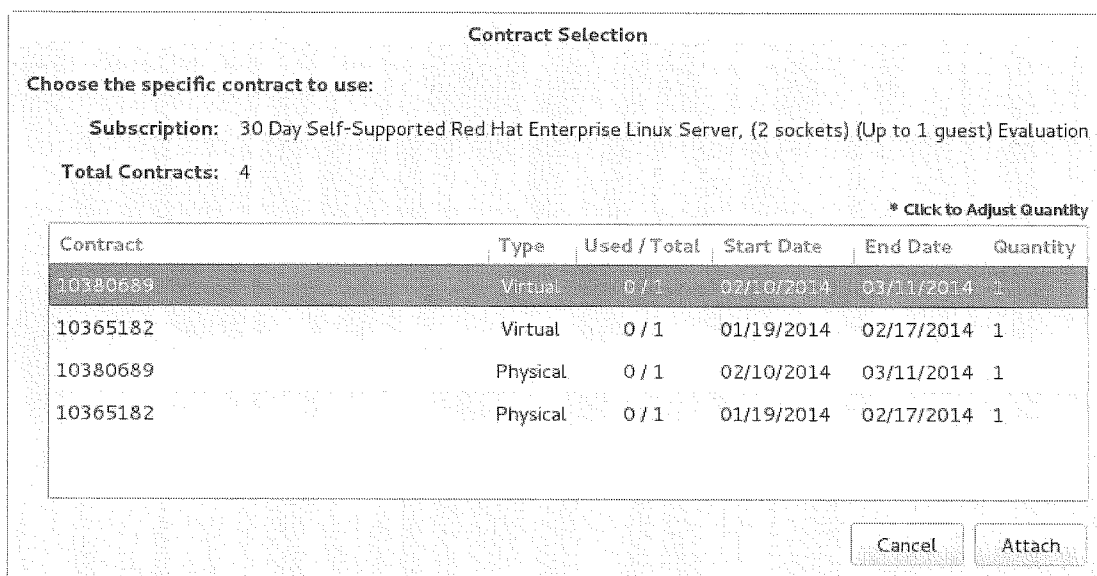


Figure 7.5: Contract selection dialog of Red Hat Subscription Manager

After a subscription has been assigned, close the **Subscription Manager** window. The system is now properly subscribed and ready to receive updates and/or install new software from Red Hat.

## Automating registrations and subscriptions

Use **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:

```
[root:@serverX ~]# subscription-manager register --username=yourusername --password=yourpassword
```

- View available subscriptions:

```
[root:@serverX ~]# subscription-manager list --available | less
```

- Auto-attach a subscription:

```
[root:@serverX ~]# subscription-manager attach --auto
```

- View consumed subscriptions:

```
[root:@serverX ~]# subscription-manager list --consumed
```

- Unregister a system:

```
[root:@serverX ~]# 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 usually issued by an on-premise subscription management service, such as Subscription Asset Manager, and will not be discussed in detail in this course.

## Entitlement certificates

An entitlement is a subscription that's been attached to a system. Digital certificates are used to store current information about entitlements on the local system. Once registered, the entitlement certificates are stored in **/etc/pki** and its subdirectories.

- **/etc/pki/product** contains certificates which indicate Red Hat products installed on the system.
- **/etc/pki/consumer** contains certificates which indicate the Red Hat account to which the system is registered.

- **/etc/pki/entitlement** contains certificates which indicate which subscriptions are attached to the system.

The certificates can be inspected with the **rct** utility directly, but normally the **subscription-manager** tools are a more user-friendly way to examine the subscriptions that are attached to the system.



## Important

Older versions of Red Hat Enterprise Linux originally supported a different subscription management method, *RHN Classic*. RHN Classic is not supported by Red Hat Enterprise Linux 7.

The method covered in this section, *Red Hat Subscription Management*, is the only one used by RHEL 7, and is the default method used by RHEL 6 after RHEL 6.3, and RHEL 5 after RHEL 5.9. RHEL 4 only supports the old method. More information on both methods is available in the references at the end of this section.



## References

**subscription-manager-gui(8)**, **subscription-manager(8)**, and **rct(8)** man pages

Get started with Red Hat Subscription Management  
<https://access.redhat.com/site/articles/433903>

Red Hat Subscription Management: Migrating from RHN and Satellite  
[https://access.redhat.com/site/documentation/en-US/Red\\_Hat\\_Subscription\\_Management/1/html-single/MigratingRHN/](https://access.redhat.com/site/documentation/en-US/Red_Hat_Subscription_Management/1/html-single/MigratingRHN/)

## Practice: Red Hat Subscription Management

### Quiz

Match the following items to their counterparts in the table.

Enable repositories	Register	Review and track	Subscribe
---------------------	----------	------------------	-----------

Description	Task
Determine the number of available subscriptions	
Enable a system to use selected Red Hat products	
Attach a system to a Red Hat account	
Provide software packages	

## Solution

Match the following items to their counterparts in the table.

Description	Task
Determine the number of available subscriptions	Review and track
Enable a system to use selected Red Hat products	Subscribe
Attach a system to a Red Hat account	Register
Provide software packages	Enable repositories

# Managing Software Updates with yum

## Objectives

After completing this section, students should be able to find, install, and update software packages using the **yum** command.

## Working with yum

**yum** is a powerful command-line tool that can be used to more flexibly manage (install, update, remove, and query) software packages. Official Red Hat packages are normally downloaded from Red Hat's content distribution network. Registering a system to the subscription management service automatically configures access to software repositories based on the attached subscriptions.

## Finding software with yum

- **yum help** will display usage information.
- **yum list** displays installed and available packages.

```
[root@serverX ~]# yum list 'http*'
Loaded plugins: langpacks
Available Packages
httpcomponents-client.noarch      4.2.5-4.el7      rhel_dvd
httpcomponents-core.noarch       4.2.4-6.el7      rhel_dvd
httpd.x86_64                     2.4.6-17.el7     rhel_dvd
httpd-devel.x86_64              2.4.6-17.el7     rhel_dvd
httpd-manual.noarch             2.4.6-17.el7     rhel_dvd
httpd-tools.x86_64              2.4.6-17.el7     rhel_dvd
```

- **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**:

```
[root@serverX ~]# yum search all 'web server'
Loaded plugins: langpacks
===== Matched: web server =====
freeradius.x86_64 : High-performance and highly configurable free RADIUS server
hsqldb.noarch : HyperSQL Database Engine
httpd.x86_64 : Apache HTTP Server
libcurl.i686 : A library for getting files from web servers
libcurl.x86_64 : A library for getting files from web servers
mod_revocator.x86_64 : CRL retrieval module for the Apache HTTP server
mod_security.x86_64 : Security module for the Apache HTTP Server
python-paste.noarch : Tools for using a Web Server Gateway Interface stack
```

- **yum info PACKAGENAME** gives detailed information about a package, including the disk space needed for installation.

To get information on the Apache HTTP Server:

```
[root@serverX ~]# yum info httpd
```

```

Loaded plugins: langpacks
Available Packages
Name       : httpd
Arch       : x86_64
Version    : 2.4.6
Release    : 17.el7
Size       : 1.1 M
Repo       : rhel_dvd
Summary    : Apache HTTP Server
URL        : http://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 pathname specified (which often include wildcard characters).

To find packages that provide the `/var/www/html` directory, use:

```

[root@serverX ~]# yum provides /var/www/html
Loaded plugins: langpacks
httpd-2.4.6-17.el7.x86_64 : Apache HTTP Server
Repo                       : rhel_dvd
Matched from:
Filename                   : /var/www/html

1:php-pear-1.9.4-21.el7.noarch : PHP Extension and Application Repository
                                : framework
Repo                       : rhel_dvd
Matched from:
Filename                   : /var/www/html

```

## Installing and removing software with **yum**

- **yum install *PACKAGENAME*** obtains and installs a software package, including any dependencies.

```

[root@serverX ~]# yum install httpd
Loaded plugins: langpacks
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-17.el7 will be installed
--> Processing Dependency: httpd-tools = 2.4.6-17.el7 for package:
    httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package:
    httpd-2.4.6-17.el7.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package:
    httpd-2.4.6-17.el7.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.4.8-3.el7 will be installed
---> Package apr-util.x86_64 0:1.5.2-6.el7 will be installed
---> Package httpd-tools.x86_64 0:2.4.6-17.el7 will be installed
---> Package mailcap.noarch 0:2.1.41-2.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

```

Package	Arch	Version	Repository	Size
Installing:				
httpd	x86_64	2.4.6-17.el7	rhel_dvd	1.1 M
Installing for dependencies:				
apr	x86_64	1.4.8-3.el7	rhel_dvd	100 k
apr-util	x86_64	1.5.2-6.el7	rhel_dvd	90 k
httpd-tools	x86_64	2.4.6-17.el7	rhel_dvd	76 k
mailcap	noarch	2.1.41-2.el7	rhel_dvd	31 k
Transaction Summary				
Install 1 Package (+4 Dependent packages)				
Total download size: 1.4 M				
Installed size: 4.3 M				
Is this ok [y/d/N]:				

- **yum update *PACKAGENAME*** obtains and installs a newer version of the software 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 will install all relevant updates.

```
[root@serverX ~]# 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 will show only the kernel version and release, and the **-a** option will show the kernel release and additional information.

```
[root@serverX ~]# yum list kernel
Loaded plugins: langpacks
Installed Packages
kernel.x86_64           3.10.0-123.0.1.el7      @anaconda/7.0

kernel.x86_64           3.10.0-84.el7           @rhel-7-server-htb-
rpms
[root@serverX ~]# uname -r
3.10.0-123.el7.x86_64
[root@serverX ~]# uname -a
Linux demo.example.com 3.10.0-123.el7.x86_64 #1 SMP Tue Nov 26 16:51:22 EST
2013 x86_64 x86_64 x86_64 GNU/Linux
```

- **yum remove *PACKAGENAME*** removes an installed software package, including any supported packages.



```
[root@serverX ~]# yum remove httpd
```



## Warning

**yum remove** will remove the package(s) listed *and any package that requires the package(s) being removed* (and package(s) which require those packages, and so on). This can lead to unexpected removal of packages, so carefully check 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 7, there are two kinds of groups. Regular groups are collections of packages. *Environment groups* are collections of other groups which include their own packages. The packages or groups provided by a group may be *mandatory* (must be installed if the group is installed), *default* (are normally installed if the group is installed), or *optional* (are not installed when the group is unless asked for specifically).

Like **yum list**, the **yum group list** (or **yum grouplist**) command will show the names of installed and available groups. Some groups are normally installed through environment groups and are hidden by default. These hidden groups can also be listed with the **yum group list hidden** command. If the **ids** option is added, the group ID will also be shown. Groups can be installed, updated, removed, and otherwise queried by name or ID.

```
[root@serverX ~]# yum group list
Loaded plugins: langpacks
Available environment groups:
  Minimal install
  Infrastructure Server
  File and Print Server
  Web Server
  Virtualization Host
  Server with GUI
Installed groups:
  Base
  Desktop Debugging and Performance Tools
  Dial-up Networking Support
  Fonts
  Input Methods
  Internet Browser
  PostgreSQL Database server
  Printing client
  X Window System
Available Groups:
  Additional Development
  Backup Client
  Backup Server
...
```

- Information about a group is displayed with **yum group info** (or **yum groupinfo**). It includes a list of mandatory, default, and optional package names or group IDs. The package names or group IDs may have a marker in front of them.

Marker	Meaning
=	Package is installed, was installed as part of the group
+	Package isn't installed, will be if the group is installed or updated
-	Package isn't installed, will not be if the group is installed or updated
<i>no marker</i>	Package is installed, but was not installed through the group.

```
[root@serverX ~]# yum group info "Identity Management Server"
Loaded plugins: langpacks

Group: Identity Management Server
Group-Id: identity-management-server
Description: Centralized management of users, servers and authentication policies.
Default Packages:
  +389-ds-base
  +ipa-admintools
  +ipa-server
  +pki-ca
Optional Packages:
  +ipa-server-trust-ad
  +nuxwdog
  +slapi-nis
```

- The **yum group install** (or **yum groupinstall**) command will install a group will install its mandatory and default packages and the packages they depend on.

```
[root@serverX ~]# yum group install "Infiniband Support"
...
Transaction Summary
=====
Install 17 Packages (+7 Dependent packages)

Total download size: 9.0 M
Installed size: 33 M
Is this ok [y/d/N]:
...
```



## Important

The behavior of **yum** groups has changed in Red Hat Enterprise Linux 7 from Red Hat Enterprise Linux 6 and earlier. In RHEL 7, 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 will be installed on 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, if any default or optional packages in the group are installed. In RHEL 7, a group is considered to be installed *only* if **yum group install** was used to install it. A new command in RHEL 7, **yum group mark install GROUPNAME** can be used to mark a group as installed, and any missing packages and their dependencies will be installed on 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 command **yum group list** did not.

## Viewing transaction history

- All install and remove transactions are logged in **/var/log/yum.log**.

```
[root@serverX ~]# tail -5 /var/log/yum.log
Feb 16 14:10:41 Installed: libnes-1.1.3-5.el7.x86_64
Feb 16 14:10:42 Installed: libmthca-1.0.6-10.el7.x86_64
Feb 16 14:10:43 Installed: libmlx4-1.0.5-7.el7.x86_64
Feb 16 14:10:43 Installed: libibcm-1.0.5-8.el7.x86_64
Feb 16 14:10:45 Installed: rdma-7.0_3.13_rc8-3.el7.noarch
```

- A summary of install and remove transactions can be viewed with **yum history**.

```
[root@serverX ~]# yum history
Loaded plugins: langpacks
ID      | Login user      | Date and time   | Action(s)      | Altered
-----|-----|-----|-----|-----
6 | Student User <student> | 2014-02-16 14:09 | Install        | 25
5 | Student User <student> | 2014-02-16 14:01 | Install        | 1
4 | System <unset>         | 2014-02-08 22:33 | Install        | 1112 EE
3 | System <unset>         | 2013-12-16 13:13 | Erase          | 4
2 | System <unset>         | 2013-12-16 13:13 | Erase          | 1
1 | System <unset>         | 2013-12-16 13:08 | Install        | 266
history list
```

- A transaction can be reversed with the **history undo** options:

```
[root@serverX ~]# yum history undo 6
Loaded plugins: langpacks
Undoing transaction 6, from Sun Feb 16 14:09:51 2014
Install      dapl-2.0.39-2.el7.x86_64      @rhel-7-server-htb-rpms
Dep-Install  graphviz-2.30.1-18.el7.x86_64 @rhel-7-server-htb-rpms
Dep-Install  graphviz-tcl-2.30.1-18.el7.x86_64 @rhel-7-server-htb-rpms
Install      ibacm-1.0.8-4.el7.x86_64      @rhel-7-server-htb-rpms
Install      ibutils-1.5.7-9.el7.x86_64    @rhel-7-server-htb-rpms
```

```
Dep-Install ibutils-libs-1.5.7-9.el7.x86_64 @rhel-7-server-htb-rpms
...
```

## 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	<b>yum list [NAME-PATTERN]</b>
List installed and available groups	<b>yum grouplist</b>
Search for a package by keyword	<b>yum search KEYWORD</b>
Show details of a package	<b>yum info PACKAGENAME</b>
Install a package	<b>yum install PACKAGENAME</b>
Install a package group	<b>yum groupinstall "GROUPNAME"</b>
Update all packages	<b>yum update</b>
Remove a package	<b>yum remove PACKAGENAME</b>
Display transaction history	<b>yum history</b>



## References

**yum(1)** and **yum.conf(5)** man pages

Additional information on **yum** may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

# Practice: Installing and Updating Software with **yum**

## Guided exercise

In this lab, you will install and remove packages and package groups.

### Outcomes:

Explore installing and removing packages with dependencies.

### *Before you begin...*

Reset your serverX system.

- 1. Search for a specific package.
  - 1.1. Attempt to run the command **gnuplot**. You should find that it is not installed.

```
[root@serverX ~]# gnuplot
bash: gnuplot: command not found...
```

- 1.2. Search for plotting packages.

```
[root@serverX ~]# yum search plot
Loaded plugins: langpacks
===== N/S matched: plot =====
emacs-gnuplot.noarch : Emacs bindings for the gnuplot main application
gnuplot.x86_64 : A program for plotting mathematical expressions and data
gnuplot-common.x86_64 : The common gnuplot parts
python-matplotlib.x86_64 : Python 2D plotting library
texlive-pst-plot.noarch : Plot data using PSTricks

Name and summary matches only, use "search all" for everything.
```

- 1.3. Find out more information about the **gnuplot** package.

```
[root@serverX ~]# yum info gnuplot
Name      : gnuplot
Arch      : x86_64
...
```

- 2. Install the **gnuplot** package.

```
[root@serverX ~]# yum install -y gnuplot
...
Dependencies Resolved

=====
Package             Arch          Version           Repository        Size
=====
Installing:
gnuplot              x86_64        4.6.2-3.el7       rhel_dvd          645 k
Installing for dependencies:
gnuplot-common       x86_64        4.6.2-3.el7       rhel_dvd          595 k
=====
```

```
Transaction Summary
=====
Install 1 Package (+1 Dependent package)
...
```

❑ 3. Remove packages.

- ❑ 3.1. Attempt to remove the **gnuplot** package, but say no. How many packages would be removed?

```
[root@serverX ~]# yum remove gnuplot
...
Removing:
gnuplot      x86_64      4.6.2-3.el7      @rhel_dvd      1.5 M

Transaction Summary
=====
Remove 1 Package

Installed size: 1.5 M
Is this ok [y/N]: n
```

- ❑ 3.2. Attempt to remove the **gnuplot-common** package, but say no. How many packages would be removed?

```
[root@serverX ~]# yum remove gnuplot-common
...
Removing:
gnuplot-common x86_64      4.6.2-3.el7      @rhel_dvd      1.4 M
Removing for dependencies:
gnuplot        x86_64      4.6.2-3.el7      @rhel_dvd      1.5 M

Transaction Summary
=====
Remove 1 Package (+1 Dependent package)

Installed size: 2.9 M
Is this ok [y/N]: n
```

- ❑ 4. Gather information about the "Compatibility Libraries" component group and install it on serverX.

- ❑ 4.1. List all available component groups.

```
[root@serverX ~]# yum grouplist
```

- ❑ 4.2. Find out more information about the *Compatibility Libraries* component group, including a list of included packages.

```
[root@serverX ~]# yum groupinfo "Compatibility Libraries"
Loaded plugins: langpacks

Group: Compatibility Libraries
Group-Id: compat-libraries
```

Description: Compatibility libraries for applications built on previous versions of Red Hat Enterprise Linux.

Mandatory Packages:

```
+compat-db47
+compat-glibc
+compat-libcap1
+compat-libf2c-34
+compat-libgfortran-41
+compat-libtiff3
+compat-openldap
+libpng12
+openssl098e
```

- 4.3. Install the *Compatibility Libraries* component group.

```
[root@serverX ~]# yum groupinstall "Compatibility Libraries"
Loaded plugins: langpacks
Resolving Dependencies
--> Running transaction check
---> Package compat-db47.x86_64 0:4.7.25-27.el7 will be installed
--> Processing Dependency: compat-db-headers = 4.7.25-27.el7 for package:
    compat-db47-4.7.25-27.el7.x86_64
...
Dependencies Resolved
```

Package	Arch	Version	Repository
Installing for group install "Compatibility Libraries":			
compat-db47	x86_64	4.7.25-27.el7	rhel_dvd
libpng12	x86_64	1.2.50-6.el7	rhel_dvd
...			
Installing for dependencies:			
compat-db-headers	noarch	4.7.25-27.el7	rhel_dvd
...			
Transaction Summary			
=====			
Install 9 Packages (+3 Dependent packages)			
Total download size: 5.5 M			
Installed size: 21 M			
Is this ok [y/d/N]: y			
...			
Installed:			
compat-db47.x86_64 0:4.7.25-27.el7			
compat-glibc.x86_64 1:2.12-4.el7			
...			
Dependency Installed:			
compat-db-headers.noarch 0:4.7.25-27.el7			
compat-glibc-headers.x86_64 1:2.12-4.el7			
Complete!			

- 5. Explore the history and undo options of **yum**.

- 5.1. Display recent **yum** history.

```
[root@serverX ~]# yum history
```

```
Loaded plugins: langpacks
ID      | Login user      | Date and time    | Action(s) | Altered
-----|-----|-----|-----|-----
      3 | root <root>     | 2014-06-05 09:33 | Install   | 12
      2 | root <root>     | 2014-06-05 09:30 | Install   | 2
      1 | System <unset>  | 2014-06-02 20:27 | Install   | 1112 EE
history list
```

- 5.2. Confirm that the last transaction is the group installation.

```
[root@serverX ~]# yum history info 3
Loaded plugins: langpacks
Transaction ID : 3
Begin time    : Thu Jun  5 09:33:19 2014
Begin rpmdb   : 1210:7c6b529424621773d5fe147315a53d558f726814
End time      :                09:33:40 2014 (21 seconds)
End rpmdb     : 1222:c283bc776b18b9578b87cdec68853f49b31ca0cc
User          : root <root>
Return-Code   : Success
Command Line   : groupinstall Compatibility Libraries
Transaction performed with:
    Installed   rpm-4.11.1-16.el7.x86_64 installed
    Installed   yum-3.4.3-117.el7.noarch installed
Packages Altered:
    Dep-Install compat-db-headers-4.7.25-27.el7.noarch @rhel_dvd
    Install     compat-db47-4.7.25-27.el7.x86_64      @rhel_dvd
...
history info
```

- 5.3. Use undo options to remove the last set of packages installed.

```
[root@serverX ~]# yum history undo 3
```



# Enabling yum Software Repositories

## Objectives

After completing this section, students should be able to enable and disable the use of Red Hat or third-party yum repositories.

## 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:

```
[root@serverX ~]# yum repolist all
Loaded plugins: langpacks
repo id                      repo name                      status
rhel-7-public-beta-debug-rpms Red Hat Enterprise Linux 7 Public disabled
rhel-7-public-beta-rpms      Red Hat Enterprise Linux 7 Public enabled: 8,520
rhel-7-public-beta-source-rpms Red Hat Enterprise Linux 7 Public disabled
repolist: 8,520
```

Enable and disable repositories with **yum-config-manager**. This will change the **enabled** parameter in the **/etc/yum.repos.d/redhat.repo** file.

```
[root@serverX ~]# yum-config-manager --enable rhel-7-public-beta-debug-rpms
Loaded plugins: langpacks
===== repo: rhel-7-public-beta-debug-rpms =====
[rhel-7-public-beta-debug-rpms]
async = True
bandwidth = 0
base_persistdir = /var/lib/yum/repos/x86_64/7Server
baseurl = https://cdn.redhat.com/content/beta/rhel/everything/7/x86_64/debug
cache = 0
cachedir = /var/cache/yum/x86_64/7Server/rhel-7-public-beta-debug-rpms
cost = 1000
deltarpm_percentage =
enabled = 1
...
```

## Enabling third-party software repositories

Third-party repositories are directories of software package files provided by a non-Red Hat source, which can be accessed by **yum** from a website, FTP server, or local file system. Yum repositories are used by non-Red Hat distributors of software, or for small collections of local packages. (For example, Adobe provides some of its free software for Linux through a yum repository.) The **content.example.com** classroom server actually hosts yum repositories for this class.

Put a file in the **/etc/yum.repos.d/** directory to enable support for a new third-party repository. Repository configuration files must end in **.repo**. 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.

### Using yum-config-manager

If the URL for a yum repository is known, a configuration file can be created with **yum-config-manager**.

```
[root@serverX ~]# yum-config-manager --add-repo="http://dl.fedoraproject.org/pub/epel/
beta/7/x86_64/"
Loaded plugins: langpacks
adding repo from: http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/

[dl.fedoraproject.org_pub_epel_beta_7_x86_64_]
name=added from: http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
baseurl=http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
enabled=1
```

A file was created in the `/etc/yum.repos.d` directory with the output shown. This file can now be modified to provide a customized name and the location of the GPG key. Administrators should download the key to a local file rather than allowing **yum** to retrieve the key from an external source.

```
[EPEL]
name=EPEL 7 Beta
baseurl=http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
```

### RPM configuration package for the repository

Some repositories provide this configuration file and GPG public key as part of an RPM package that can be downloaded and installed using **yum localinstall**. One example of this is the volunteer project EPEL (Extra Packages for Enterprise Linux), which provides software not supported by Red Hat but compatible with Red Hat Enterprise Linux.

Installing the Red Hat Enterprise Linux 7 EPEL repo package:

```
[root@serverX ~]# rpm --import http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-7
[root@serverX ~]# yum install http://dl.fedoraproject.org/pub/epel/beta/7/x86_64/epel-
release-7-0.1.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.

```
[root@serverX ~]# cat /etc/yum.repos.d/epel.repo
[epel]
name=Extra Packages for Enterprise Linux 7 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch
failovermethod=priority
enabled=1
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

[epel-debuginfo]
name=Extra Packages for Enterprise Linux 7 - $basearch - Debug
#baseurl=http://download.fedoraproject.org/pub/epel/7/$basearch/debug
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-7&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
gpgcheck=1

[epel-source]
```

```
name=Extra Packages for Enterprise Linux 7 - $basearch - Source
#baseurl=http://download.fedoraproject.org/pub/epel/7/SRPMS
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-source-7&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
gpgcheck=1
```

The **enabled=0** parameter can be included so that a repository is defined but not searched by default. Repositories can be enabled and disabled persistently with **yum-config-manager** or temporarily with **--enablerepo=PATTERN** and **--disablerepo=PATTERN** options on **yum**.



## Warning

Install the RPM GPG key before installing signed packages. This will verify that the packages belong to a key which has been imported. Otherwise, **yum** will complain about the 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

Additional information may be available in the section on configuring yum and yum repositories in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at <http://docs.redhat.com/>

**yum(1)**, **yum.conf(5)**, and **yum-config-manager(1)** man pages

## Practice: Enabling Software Repositories

### Guided exercise

In this lab, you will configure your server to use a separate **yum** repository to obtain updates, and update your machine.

#### Outcomes:

The system will be configured to obtain software updates from a classroom server and the system will be running the latest Linux kernel.

#### *Before you begin...*

Reset your serverX system.

- 1. Configure the system to obtain software from two classroom repositories:
  - Classroom packages provided at [http://content.example.com/rhel7.0/x86\\_64/rht](http://content.example.com/rhel7.0/x86_64/rht)
  - Updates provided at [http://content.example.com/rhel7.0/x86\\_64/errata](http://content.example.com/rhel7.0/x86_64/errata)
- 1.1. Use **yum-config-manager** to add the classroom packages repository.

```
[root@serverX ~]# yum-config-manager --add-repo="http://
content.example.com/rhel7.0/x86_64/rht"
Loaded plugins: langpacks
adding repo from: http://content.example.com/rhel7.0/x86_64/rht

[content.example.com_rhel7.0_x86_64_rht]
name=added from: http://content.example.com/rhel7.0/x86_64/rht
baseurl=http://content.example.com/rhel7.0/x86_64/rht
enabled=1
```

- 1.2. Create the file **/etc/yum.repos.d/errata.repo** to enable the “Updates” repository with the following content:

```
[updates]
name=Red Hat Updates
baseurl = http://content.example.com/rhel7.0/x86_64/errata
enabled=1
gpgcheck=0
```

- 2. Use **yum-config-manager** to disable the classroom packages repository.

```
[root@serverX ~]# yum-config-manager --disable
content.example.com_rhel7.0_x86_64_rht
Loaded plugins: langpacks
===== repo: content.example.com_rhel7.0_x86_64_rht =====
[content.example.com_rhel7.0_x86_64_rht]
...
enabled = 0
...
```

- 3. Update all relevant software provided using **yum update**.

```
[root@serverX ~]# yum update -y
```

- ❑ 4. Verify that there are two versions of the kernel installed. Which version is currently in use?

```
[root@serverX ~]# yum list kernel
```

```
[root@serverX ~]# uname -r
```

- ❑ 5. Reboot your serverX, then repeat the previous step. Which version is currently in use?

```
[root@serverX ~]# yum list kernel
```

```
[root@serverX ~]# uname -r
```

- ❑ 6. List, then install, the **rht-system** package.

```
[root@serverX ~]# yum list rht*
Loaded plugins: langpacks
Available Packages
rht-system.noarch                1.0.0-2.el7                updates
[root@serverX ~]# yum -y install rht-system
Loaded plugins: langpacks
Resolving Dependencies
...
```

## Lab: Installing and Updating Software Packages

### Performance checklist

In this lab, you will install and update select software packages.

#### Outcomes:

New and updated packages are installed on the system.

#### *Before you begin...*

Reset your serverX system.

1. Create the file `/etc/yum.repos.d/errata.repo`, to enable the "Updates" repository found on the content machine. It should access content found at the following URL: `http://content.example.com/rhel7.0/x86_64/errata`. Do *not* check GPG signatures.
2. Configure serverX to adhere to very specific software requirements. It must have the latest version of the following packages installed. Do not install all updates. Only install updates for the packages listed if they are available.
  - 2.1. **kernel** (existing package w/ an update)
  - 2.2. **xsane-gimp** (new package)
  - 2.3. **rht-system** (new package)
3. For security reasons, it should not have the **wvdial** package installed.
4. When you are ready to check your work, run **lab software grade** on serverX.

## Solution

In this lab, you will install and update select software packages.

### Outcomes:

New and updated packages are installed on the system.

### *Before you begin...*

Reset your serverX system.

1. Create the file `/etc/yum.repos.d/errata.repo`, to enable the "Updates" repository found on the content machine. It should access content found at the following URL: `http://content.example.com/rhel7.0/x86_64/errata`. Do *not* check GPG signatures.

Create the file `/etc/yum.repos.d/errata.repo` with the following content:

```
[updates]
name=Red Hat Updates
baseurl=http://content.example.com/rhel7.0/x86_64/errata
enabled=1
gpgcheck=0
```

2. Configure serverX to adhere to very specific software requirements. It must have the latest version of the following packages installed. Do not install all updates. Only install updates for the packages listed if they are available.

- 2.1. **kernel** (existing package w/ an update)

```
yum update kernel
```

- 2.2. **xsane-gimp** (new package)

```
yum install xsane-gimp
```

- 2.3. **rht-system** (new package)

```
yum install rht-system
```

3. For security reasons, it should not have the **wvdial** package installed.

```
yum remove wvdial
```

4. When you are ready to check your work, run **lab software grade** on serverX.

```
[student@serverX ~]$ lab software grade
```

## Summary

### Attaching Systems to Subscriptions for Software Updates

Registering systems allows access to software updates for installed products.

### Managing Software Updates with **yum**

**yum** is used to install and update software packages.

### Enabling **yum** Software Repositories

Repositories for **yum** are configured in the **/etc/yum.repos.d** directory.





## CHAPTER 8

# CREATING AND MOUNTING FILE SYSTEMS

Overview	
<b>Goal</b>	To create and manage disks, partitions, and filesystems from the command line.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Access the contents of file systems.</li><li>• Manage simple partitions and filesystems.</li><li>• Manage swap space.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Mounting and Unmounting File Systems (and Practice)</li><li>• Adding Partitions, Filesystems, and Persistent Mounts (and Practice)</li><li>• Managing Swap Space (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Adding Disks, Partitions, and Filesystems to a Linux System</li></ul>

# Mounting and Unmounting File Systems

## Objectives

After completing this section, students 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 SATA/PATA or SCSI device needs to be mounted manually 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 target directory where the file system is made available after mounting it. The target directory is referred to as a mount point.

The **mount** command expects the file system argument in one of two different ways:

- The device file of the partition holding the file system, residing in **/dev**.
- The **UUID**, a universal unique identifier of the file system.



### Note

As long as a file system is not recreated, the UUID stays the same. The device file can change; for example, if the order of the devices is changed or if additional devices are added to the system.

The **blkid** command gives an overview of existing partitions with a file system on them and the UUID of the file system, as well as the file system used to format the partition.

```
[root@serverX ~]# blkid
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"
```



### Note

A file system can be mounted on an existing directory. The **/mnt** directory exists by default and provides an entry point for mount points. It is used for manually mounting disks. It is recommended to create a subdirectory under **/mnt** and use that subdirectory as a mount point unless there is a reason to mount the file system in another specific location in the file system hierarchy.

Mount by device file of the partition that holds the file system.

```
[root@serverX ~]# mount /dev/vdb1 /mnt/mydata
```

Mount the file system by universal unique id, or the UUID, of the file system.

```
[root@serverX ~]# mount UUID="46f543fd-78c9-4526-a857-244811be2d88" /mnt/mydata
```



## Note

If the directory acting as mount point is not empty, the files that exist in that directory are not accessible as long as a file system is mounted there. All files written to the mount point directory end up on the file system mounted there.

## Unmounting file systems

To unmount a file system, the **umount** command expects the mount point as an argument.

Change to the **/mnt/mydata** directory. Try to unmount the device mounted on the **/mnt/mydata** mount point. It will fail.

```
[root@serverX ~]# cd /mnt/mydata
[root@serverX mydata]# umount /mnt/mydata
umount: /mnt/mydata: target is busy.
(In some cases useful info about processes that use
the device is found by lsof(8) or fuser(1))
```

Unmounting is not possible if the mount point is accessed by a process. For **umount** to be successful, the process needs to stop accessing the mount point.

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@serverX mydata]# lsof /mnt/mydata
COMMAND  PID USER  FD   TYPE DEVICE SIZE/OFF  NODE NAME
bash      1593 root   cwd   DIR  253,2      6  128 /mnt/mydata
lsof      2532 root   cwd   DIR  253,2     19  128 /mnt/mydata
lsof      2533 root   cwd   DIR  253,2     19  128 /mnt/mydata
```

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@serverX mydata]# cd
[root@serverX ~]# umount /mnt/mydata
```



## Note

A common cause for the file system on the mount point to be busy is if the current working directory of a shell prompt is below the active mount point. The process accessing the mount point is **bash**. Changing to a directory outside the mount point allows the device to be unmounted.

## Accessing removable storage devices

Removable media, such as USB flash devices and drives, get automatically mounted by the graphical desktop environment when plugged in. The mount point for the removable medium is

`/run/media/<user>/<label>`. The `<user>` is the user logged into the graphical environment. The `<label>` is the name given to the file system when it was created.



### Warning

To safely remove USB media from the system, it is required to unmount it before physically removing it from the USB slot to synchronize the file system. Removing a USB storage device without unmounting the file system on it can result in data loss.



### References

`mount(8)`, `umount(8)`, and `lsotf(8)` man pages

# Practice: Mounting and Unmounting File Systems

## Guided exercise

In this lab, you will mount and unmount file systems.

### Outcomes:

The user identifies and mounts a new file system at a specified mount point, then unmounts it.

### Before you begin...

Reset your serverX system. Run the script **lab fs setup** before starting the exercise.

- ❑ 1. A new partition with a file system has been added to the second disk (vdb) on your serverX machine. Mount the newly available partition by UUID at the newly created mount point **/mnt/newspace**.

- ❑ 1.1. Use the **blkid** to discover the UUID of the newly added partition, **vdb1**, on serverX.

```
[root@serverX ~]# blkid
/dev/vda1: UUID="46f543fd-78c9-4526-a857-244811be2d88" TYPE="xfs"
/dev/vdb1: UUID="7c5e3fbb-34eb-4431-a4a5-9b887c1b6866" TYPE="xfs"
```

- ❑ 1.2. Create the mount point **/mnt/newspace** on serverX.

```
[root@serverX ~]# mkdir /mnt/newspace
```

- ❑ 1.3. Mount the file system by UUID on the **/mnt/newspace** directory of the serverX machine.

```
[root@serverX ~]# mount UUID="7c5e3fbb-34eb-4431-a4a5-9b887c1b6866" /mnt/newspace
```

- ❑ 2. Change to the **/mnt/newspace** directory and create a new directory, **/mnt/newspace/newdir**, with an empty file, **/mnt/newspace/newdir/newfile**, on serverX.

- ❑ 2.1. Change to the **/mnt/newspace** directory on serverX.

```
[root@serverX ~]# cd /mnt/newspace
```

- ❑ 2.2. Create a new directory, **/mnt/newspace/newdir**, on serverX.

```
[root@serverX newspace]# mkdir newdir
```

- ❑ 2.3. Create a new empty file, **/mnt/newspace/newdir/newfile**, on serverX.

```
[root@serverX newspace]# touch newdir/newfile
```

- ❑ 3. Unmount the file system mounted on the **/mnt/newspace** directory on serverX.
  - ❑ 3.1. Try to unmount **/mnt/newspace** while the current directory on the shell is still **/mnt/newspace** on serverX.

```
[root@serverX newspace]# umount /mnt/newspace
```

- ❑ 3.2. Change the current directory on the shell to **/root**.

```
[root@serverX newspace]# cd  
[root@serverX ~]#
```

- ❑ 3.3. Successfully unmount **/mnt/newspace** on serverX.

```
[root@serverX ~]# umount /mnt/newspace
```

# Adding Partitions, File Systems, and Persistent Mounts

## Objectives

After completing this section, students should be able to:

- Create and remove disk partitions on disks with an MBR partitioning scheme using **fdisk**.
- Create and remove disk partitions on disks with a GPT partitioning scheme using **gdisk**.
- Format devices with file systems using **mkfs**.
- Mount file systems into the directory tree.

## Disk partitioning

Disk partitioning allows a hard drive to be divided 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. Some examples of situations where disk partitioning is necessary or beneficial are:

- Limit available space to applications or users.
- Allow multibooting of different operating systems from the same disk.
- Separate operating system and program files from user files.
- Create separate area for OS virtual memory swapping.
- Limit disk space usage to improve performance of diagnostic tools and backup imaging.

### MBR partitioning scheme

Since 1982, the *Master Boot Record (MBR)* partitioning scheme has dictated how disks should be 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, administrator can create a maximum of 15 partitions. Since partition size data are stored as 32-bit values, disks partitioned with the MBR scheme have a maximum disk and partition size limit of 2 TiB.

With the advent of hard drives with ever-increasing capacity, the 2 TiB disk and partition size limit of the aged MBR partitioning scheme is no longer a theoretical limit, but rather a real-world problem that is being encountered 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 imposed by the old MBR-based scheme. Per UEFI specifications, GPT defaults to supporting up to 128 partitions. Unlike MBR, which uses 32 bits for storing logical block addresses and size information, GPT allocates 64 bits for logical block addresses. This allows GPT to accommodate partitions and disks of up to 8 *zebibyte (ZiB)*, or 8 billion tebibytes.



## Note

GPT's 8-ZiB limit is based on a 512-byte block size. With hard drive vendors transitioning to 4,096-byte blocks, this limitation will increase to 64 ZiB.

In addition to addressing the limitations of the MBR partitioning scheme, GPT also offers some additional features and benefits. Per its namesake, GPT uses 128-bit GUIDs to uniquely identify each disk and partition. In contrast to MBR, which has a single point of failure, 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. In addition, GPT employs the use of CRC checksum to detect errors and corruption in the GPT header and partition table.

## Managing MBR partitions with fdisk

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. For disks with the MBR partitioning scheme, the **fdisk** partition editor can be used to perform these operations.

### Creating MBR disk partitions

Creating an MBR-style disk partition involves eight steps:

1. Specify the disk device to create the partition on.

As the **root** user, execute the **fdisk** command and specify the disk device name as an argument. This will start the **fdisk** command in interactive mode, and will present a **command** prompt.

```
[root@serverX ~]# fdisk /dev/vdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):
```

2. Request a new primary or extended partition.

Enter **n** to request a new partition and specify whether the partition should be created as a *primary* or *extended* partition. The default selection is the *primary* partition type.

```
Partition type:
  p   primary (0 primary, 0 extended, 4 free)
  e   extended
Select (default p): p
```





## Note

For situations where more than four partitions are needed on a disk, this limit can be bypassed by creating three primary partitions and one extended partition. This extended partition serves as a container within which multiple logical partitions can be created.

### 3. Specify partition number.

This partition number serves as the identification number of the new partition on the disk for use in future partition operations. The default value is the lowest unused partition number.

```
Partition number (1-4, default 1): 1
```

### 4. Specify the first sector on the disk that the new partition will start on.

The default value is the first available sector on the disk.

```
First sector (2048-20971519, default 2048): 2048
```

### 5. Specify the last sector on the disk that the new partition will end on.

The default value is the last of the available, unallocated sectors contiguous to the new partition's first sector.

```
Last sector, +sectors or +size{K,M,G} (6144-20971519, default 20971519): 1050623
```

In addition to the ending sector number, **fdisk** can also accept a number representing the desired size of the partition expressed in sectors.

```
Last sector, +sectors or +size{K,M,G} (6144-20971519, default 20971519): +52488
```

The final, and most user-friendly, input option offered by **fdisk** is to specify the size of the new partition in units of KiB, MiB, or GiB.

```
Last sector, +sectors or +size{K,M,G} (6144-20971519, default 20971519): +512M
```

Once the partition's ending boundary is entered, **fdisk** will then display a confirmation of the partition creation.

```
Partition 1 of type Linux and of size 512 MiB is set
```

### 6. Define partition type.

If the newly created partition should have a type other than *Linux*, enter the **t** command to change a partition's type. Enter the hex code for the new partition type. If needed, a

table of the hex codes for all partition types can be displayed with the **L** command. Setting the partition type correctly is crucial, since some tools rely on it to function properly. For example, when the Linux kernel encounters a partition of type *0xfd*, Linux RAID, it will attempt to autostart the RAID volume.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'
```

7. Save partition table changes.

Issue the **w** command to finalize the partition creation request by writing the changes to the disk's partition table and exiting the **fdisk** program.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource
busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

8. Initiate a kernel re-read of the new partition table.

Run the **partprobe** command with the disk device name as an argument to force a re-read of its partition table.

```
[root@serverX ~]# partprobe /dev/vdb
```



### Important

The **fdisk** program queues all partition table edits and writes them to disk only when the administrator issues the **w** command to write all partition table changes to disk. If the **w** command is not executed prior to exiting the interactive **fdisk** session, all requested changes to the partition table will be discarded and the disk's partition table will remain unchanged. This feature is especially useful when erroneous commands are issued to **fdisk**. To discard the erroneous commands and avoid their unintended consequences, simply exit **fdisk** without saving the partition table changes.

### Removing MBR disk partitions

There are five steps needed to remove a partition from a disk with the MBR partitioning layout using **fdisk**.

1. Specify the disk which contains the partition to be removed.

Execute the **fdisk** command and specify the disk device name as an argument.

```
[root@serverX ~]# fdisk /dev/vdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):
```

2. Identify the partition number of the partition to delete.

Enter **p** to print the partition table and **fdisk** will display information about the disk and its partitions.

```
Command (m for help): p

Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xd2368130

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1           2048     1050623      524288    82  Linux swap / Solaris
```

3. Request the partition deletion.

Enter the **d** command to initiate partition removal and specify the partition number of the partition to be removed.

```
Command (m for help): d
Selected partition 1
Partition 1 is deleted
```

4. Save partition table changes.

Issue the **w** command to finalize the partition removal request by writing the changes to the disk's partition table.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource
busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

5. Initiate a kernel re-read of the new partition table.

Inform the kernel to re-read the partition table with **partprobe**.

```
[root@serverX ~]# partprobe /dev/vdb
```

## Managing GPT partitions with **gdisk**

For disks with the GPT partitioning scheme, the **gdisk** partition editor can be used to manage partitions.



### Warning

While GPT support has been added to **fdisk**, it is still considered experimental, so the **gdisk** command should be used to make partition changes on disks partitioned with the GPT partitioning scheme.

### Creating GPT disk partitions

There are eight steps required to create a GPT-style partition.

1. Specify the disk device to create the partition on.

Execute the **gdisk** command and specify the disk device name as an argument. This will start the **gdisk** command in interactive mode, and will present a **command** prompt.

```
[root@serverX ~]# gdisk /dev/vdb
GPT fdisk (gdisk) version 0.8.6

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries.

Command (? for help):
```

2. Request a new partition.

Enter **n** to create a new partition.

```
Command (? for help): n
```

3. Specify the partition number.

This partition number serves as the identification number of the partition on the disk for use in future partition operations. The default value is the lowest unused partition number.

```
Partition number (1-128, default 1): 1
```

4. Specify the disk location that the new partition will start from.

**gdisk** allows for two different input types. The first input type is an absolute disk sector number representing the first sector of the new partition.

```
First sector (34-20971486, default = 2048) or {+}size{KMGTp}: 2048
```

The second input type indicates the partition's starting sector by its position relative to the first or last sector of the first contiguous block of free sectors on the disk. Using this relative sector position format, input is specified in units of KiB, MiB, GiB, TiB, or PiB.

For example, a value of **+512M** signifies a sector position that is 512 MiB **after** the beginning of the next group of contiguous available sectors. On the other hand, a value of **-512M** denotes a sector positioned 512 MiB **before** the end of this group of contiguous available sectors.

5. Specify the last sector on the disk that the new partition will end on.

The default value is the last of the available, unallocated sectors contiguous to the new partition's first sector.

```
Last sector (2048-20971486, default = 20971486) or {+}size{KMGTp}: 1050623
```

In addition to the absolute ending sector number, **gdisk** also offers the more user-friendly input option of specifying the end boundary of the new partition in units of KiB, MiB, GiB, TiB, or PiB from the beginning or end of the group of contiguous available sectors. A value of **+512M** signifies an ending partition position that is 512 MiB **after** the first sector.

```
Last sector (2048-20971486, default = 20971486) or {+}size{KMGTp}: +512M
```

A value of **-512M** indicates an ending partition position that is 512 MiB **before** the end of the contiguous available sectors.

```
Last sector (2048-20971486, default = 20971486) or {+}size{KMGTp}: -512M
```

6. Define partition type.

New partitions created by **gdisk** default to type *Linux* file system. If a different partition type is desired, enter the corresponding hex code. If needed, a table of the hex codes for all partition types can be displayed with the **L** command.

```
Current type is 'Linux filesystem'
```

```
Hex code or GUID (L to show codes, Enter = 8300): 8e00
```

```
Changed type of partition to 'Linux LVM'
```

7. Save partition table changes.

Issue the **w** command to finalize the partition creation request by writing the changes to the disk's partition table. Enter **y** when **gdisk** prompts for a final confirmation.

```
Command (? for help): w
```

```
Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!
```

```
Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/vdb.
The operation has completed successfully.
```

8. Initiate a kernel re-read of the new partition table.

Run the **partprobe** command with the disk device name as an argument to force a re-read of its partition table.

```
[root@serverX ~]# partprobe /dev/vdb
```



## Important

The **gdisk** program queues all partition table edits and writes them to disk only when the administrator issues the **w** command to write all partition table changes to disk. If the **w** command is not executed prior to exiting the interactive **gdisk** session, all requested changes to the partition table will be discarded and the disk's partition table will remain unchanged. This feature is especially useful when erroneous commands are issued to **gdisk**. To discard the erroneous commands and avoid their unintended consequences, simply exit **gdisk** without saving the partition table changes.

## Removing GPT disk partitions

There are five steps required to remove a partition from a disk with the GPT partitioning scheme using **gdisk**.

1. Specify the disk which contains the partition to be removed.

Execute the **gdisk** command and specify the disk device name as an argument.

```
[root@serverX ~]# gdisk /dev/vdb
GPT fdisk (gdisk) version 0.8.6

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help):
```

2. Identify the partition number of the partition to delete.

Enter **p** to print the partition table. Note the number in the *Number* field for the partition to be deleted.

```
Command (? for help): p
Disk /dev/vdb: 20971520 sectors, 10.0 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): 8B181B97-5259-4C8F-8825-1A973B8FA553
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 20971486
```

```
Partitions will be aligned on 2048-sector boundaries
Total free space is 19922877 sectors (9.5 GiB)
```

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	1050623	512.0 MiB	8E00	Linux LVM

3. Request the partition deletion.

Enter the **d** command to initiate partition removal.

```
Command (? for help): d
Using 1
```

4. Save partition table changes.

Issue the **w** command to finalize the partition removal request by writing the changes to the disk's partition table. Enter **y** when **gdisk** prompts for a final confirmation.

```
Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/vdb.
The operation has completed successfully.
```

5. Initiate a kernel re-read of the new partition table.

Inform the kernel to re-read the partition table with **partprobe**.

```
[root@serverX ~]# partprobe /dev/vdb
```

## Creating file systems

After a block device has been created, the next step is applying a file system format to it. A file system applies a structure to the block device so that data can be stored and retrieved from it. Red Hat Enterprise Linux supports many different file system types, but two common ones are **xfs** and **ext4**. **xfs** is used by default in **anaconda**, the installer for Red Hat Enterprise Linux.

The **mkfs** command can be used to apply a file system to a block device. If no type is specified, an extended type two (ext2) file system will be used, which for many uses is not desirable. To specify the file system type, a **-t** should be used.

```
[root@serverX ~]# mkfs -t xfs /dev/vdb1
meta-data=/dev/vdb1            isize=256    agcount=4, agsize=16384 blks
=                               sectsz=512   attr=2, projid32bit=1
=                               crc=0
data      =                     bsize=4096   blocks=65536, imaxpct=25
=                               sunit=0        swidth=0 blks
naming    =version 2           bsize=4096   ascii-ci=0 ftype=0
log       =internal log       bsize=4096   blocks=853, version=2
=                               sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                extsz=4096   blocks=0, rtextents=0
```

## Mounting file systems

Once the file system format has been applied, the last step to adding a new file system is to attach the file system into the directory structure. When the file system is attached into the directory hierarchy, user space utilities can access or write files on the device.

### Manually mounting file systems

Administrators can use the **mount** command to manually attach the device onto a directory location, or *mount point*, by specifying the device and the mount point, as well as any options that may be desired, to customize the behavior of the device.

```
[root@serverX ~]# mount /dev/vdb1 /mnt
```

The **mount** can also be used to view currently mounted file systems, the mount points, and options.

```
[root@serverX ~]# mount | grep vdb1
/dev/vdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

Manually mounting a file system is an excellent way to verify that a formatted device is accessible or working in the way desired. However, once the system is rebooted, the file system, while it still exists and has intact data, will not be mounted into the directory tree again. If an administrator wants the file system to be persistently mounted, a listing for the file system needs to be added to **/etc/fstab**.

### Persistently mounting file systems

By adding a listing for a device into the **/etc/fstab** file, administrators can configure a device to be mounted to a mount point at system boot.

**/etc/fstab** is a white space-delimited file with six fields per line.

```
[root@serverX ~]# cat /etc/fstab
#
# /etc/fstab
# Created by anaconda on Thu Mar 20 14:52:46 2014
#
# 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
#
UUID=7a20315d-ed8b-4e75-a5b6-24ff9e1f9838 / xfs defaults 1 1
```

The first field specifies the device to be used. In the previous example, the **UUID** is being used to specify the device. Alternatively, the device file could be used; for example, **/dev/vdb1**. The **UUID** is stored in the file system superblock and created when the file system is created.





## 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. The block device file may change, but the **UUID** would remain intact in the superblock of the device.

The **blkid** command can be used to scan the block devices connected to a machine and report on data like the assigned **UUID** and file system format.

```
[root@serverX ~]# blkid /dev/vdb1
/dev/vdb1: UUID="226a7c4f-e309-4cb3-9e76-6ef972dd8600" TYPE="xfs"
```

The second field is the mount point where the device should be attached into the directory hierarchy. The mount point should already exist; if it does not, it can be created with **mkdir**.

The third field contains the file system type that has been applied to the block device.

The fourth field is the list of options that should be applied to the device when mounted to customize the behavior. This field is required, and there is a set of commonly used options called **defaults**. Other options are documented in the **mount** man page.

The last two fields are the dump flag and fsck order. The dump flag is used with the **dump** command to make a backup of the contents of the device. The fsck order field determines if the **fsck** should be run at boot time, in the event that the file system was not unmounted cleanly. The value of the fsck order indicates the order in which file systems should have **fsck** run on them if multiple file systems are required to be checked.

```
UUID=226a7c4f-e309-4cb3-9e76-6ef972dd8600 /mnt xfs defaults 1 2
```



## Note

Having an incorrect entry in **/etc/fstab** may render the machine unbootable. To avoid that situation, an administrator should verify that the entry is valid by unmounting the new file system and using **mount -a**, which reads **/etc/fstab**, to mount the file system back into place. If the **mount -a** command returns an error, it should be corrected before rebooting the machine.



## References

**fdisk(8)**, **gdisk(8)**, **mkfs(8)**, **mount(8)**, **fstab(5)** man pages

## Practice: Adding Partitions, File Systems, and Persistent Mounts

### Guided exercise

In this lab, you will create an MBR partition on a newly allocated disk, format the partition with an ext4 file system, and configure the file system for persistent mounting.

Resources:	
<b>Machines:</b>	serverX

#### Outcomes:

1 GiB ext4 file system on second disk persistently mounted at **/archive**.

#### Before you begin...

- Reset your serverX system.
- Log into serverX.
- Switch to **root** using **sudo -i**.

You have been asked to archive data to a new directory, **/archive**, on serverX. You have been allocated a second disk for this purpose. The **/archive** directory will require 1 GiB of space. To make sure that the **/archive** directory is always available for use, you will need to configure the newly created file system to be persistently mounted at **/archive** even after a server reboot.

Once you have completed your work, reboot your serverX machine and verify that the newly created file system is persistently mounted at **/archive** after the reboot.

- 1. Create a 1 GiB MBR partition on **/dev/vdb** of type **Linux**.

- 1.1. Use **fdisk** to modify the second disk.

```
[root@serverX ~]# fdisk /dev/vdb
```

- 1.2. Display the original partition table, then add a new partition that is 1 GiB in size.

```
Command (m for help): p
Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xfd41a9d3

Device Boot      Start         End      Blocks   Id  System
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
```

```

Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048): Enter
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default
20971519): +1G
Partition 1 of type Linux and of size 1 GiB is set

```

- 1.3. Save the partition table changes.

```

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

```

- 1.4. If **fdisk** issues a warning, then run the **partprobe** command to make the kernel aware of the partition table change. This will not be necessary if the disk device is currently unused.

```
[root@serverX ~]# partprobe
```

- 2. Format the newly created partition with the ext4 file system.

```

[root@serverX ~]# mkfs -t ext4 /dev/vdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

```

- 3. Configure the newly created file system to persistently mount at **/archive**.

- 3.1. Create the **/archive** directory mount point.

```
[root@serverX ~]# mkdir /archive
```

- 3.2. Determine the UUID of the new partition on the second disk.

```

[root@serverX ~]# blkid /dev/vdb1
/dev/vdb1: UUID="5fcb234a-cf18-4d0d-96ab-66a4d1ad08f5" TYPE="ext4"

```

- 3.3. Add an entry to **/etc/fstab**.

```
UUID=5fcb234a-cf18-4d0d-96ab-66a4d1ad08f5 /archive ext4 defaults 0 2
```

- 4. Test mounting the newly created file system.

- 4.1. Execute the **mount** command to mount the new file system using the new entry added to **/etc/fstab**.

```
[root@serverX ~]# mount -a
```

- 4.2. Verify that the new file system is mounted at **/archive**.

```
[root@serverX ~]# mount | grep -w /archive  
/dev/vdb1 on /archive type ext4 (rw,relatime,seclabel,data=ordered)
```

- 5. Reboot serverX. After the server has rebooted, log in and verify that **/dev/vdb1** is mounted at **/archive**.

```
[student@serverX ~]$ mount | grep ^/  
/dev/vda1 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)  
/dev/vdb1 on /archive type ext4 (rw,relatime,seclabel,data=ordered)
```

# Managing Swap Space

## Objectives

After completing this section, students should be able to:

- Create and format a partition for swap space.
- Activate the swap space.

## Swap space concepts

A *swap space* is an area of a disk which can be used with the Linux kernel memory management subsystem. Swap spaces are used to supplement the system RAM by holding inactive pages of memory. The combined system RAM plus swap spaces is called *virtual memory*.

When the memory usage on a system exceeds a defined limit, the kernel will comb through RAM looking for idle memory pages assigned to processes. The kernel writes the idle page to the swap area, and will reassign the RAM page to be used by another process. If a program requires access to a page that has been written to disk, the kernel will locate another idle page of memory, write it to disk, then recall the needed page from the swap area.

Since swap areas reside on disk, swap is incredibly slow when compared with RAM. While it is used to augment system RAM, usage of swap spaces should be kept to a minimum whenever possible.

## Create a swap space

To create a swap space, an administrator needs to do three things:

- Create a partition.
- Set the type of the partition as **82 Linux Swap**.
- Format a swap signature on the device.

### Create a partition

Use a tool, such as **fdisk**, to create a partition of the desired size. In the following example, a 256 MiB partition will be created.

```
[root@serverX ~]# fdisk /dev/vdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table
Building a new DOS disklabel with disk identifier 0x34e4e6d7.

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-20971519, default 2048): Enter
```

```
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519): +256M
Partition 1 of type Linux and of size 256 MiB is set
```

```
Command (m for help): p
```

```
Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x34e4e6d7
```

Device	Boot	Start	End	Blocks	Id	System
/dev/vdb1		2048	526335	262144	83	Linux

### Assign the partition type

After the swap partition has been created, it is recommended practice to change the partition's type, or system ID, to **82 Linux Swap**. In the past, tools looked at the partition type to determine if the device should be activated; however, that is no longer the case. Even though the partition type is not used by utilities any longer, having the type set allows administrators to quickly determine the partition's purpose. The following example continues from within **fdisk**.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'
```

```
Command (m for help): p
```

```
Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x34e4e6d7
```

Device	Boot	Start	End	Blocks	Id	System
/dev/vdb1		2048	526335	262144	82	Linux swap / Solaris

### Format 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 it can be used for storing memory pages.

```
[root@serverX ~]# mkswap /dev/vdb1
Setting up swspace version 1, size = 262140 KiB
no label, UUID=fbd7fa60-b781-44a8-961b-37ac3ef572bf
```

## Activate a swap space

An administrator can use the **swapon** command to activate a formatted swap space. **swapon** can be called on the device, or **swapon -a** will activate all swap spaces listed in the **/etc/fstab** file.

```
[root@serverX ~]# free
```

	total	used	free	shared	buffers	cached
Mem:	1885252	791812	1093440	17092	688	292024
-/+ buffers/cache:		499100	1386152			
Swap:	0	0	0			
[root@serverX ~]# <b>swapon /dev/vdb1</b>						
[root@serverX ~]# <b>free</b>						
	total	used	free	shared	buffers	cached
Mem:	1885252	792116	1093136	17092	692	292096
-/+ buffers/cache:		499328	1385924			
Swap:	<b>262140</b>	<b>0</b>	<b>262140</b>			

### Persistently activate swap space

It is likely that a swap space will be required to automatically activate every time the machine boots. In order for the machine to activate the swap space at every boot, it must be configured in the **/etc/fstab** file.

If needed, an administrator can deactivate a swap space using the **swapoff** command. A **swapoff** will only be successful if any swapped data can be written to other active swap spaces or back into memory. If data cannot be written to other places, the **swapoff** will fail, with an error, and the swap space will stay active.

The following is an example line in **/etc/fstab** adding a previously created swap space.

```
UUID=fbd7fa60-b781-44a8-961b-37ac3ef572bf swap swap defaults 0 0
```

The example uses the **UUID** as the first field. The **UUID** is stored in the swap signature stored on the device, and was part of the output of **mkswap**. If the output of **mkswap** has been lost, the **blkid** command can be used to scan the system and report on all attached block devices. If the administrator does not wish to use the **UUID**, the raw device name can also be used 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 is the placeholder value **swap**.

The third field is the file system type. The file system type for a swap space is **swap**.

The fourth field is for options. In the example, the option **defaults** is used. **defaults** includes the mount option **auto**, which is what causes the swap space to be automatically activated at boot.

The final two fields are the dump flag and fsck order. Swap spaces require neither backing up nor file system checking.



### Note

By default, swap spaces are used in series, meaning that the first activated swap space will be used until it is full, then the kernel will start using the second swap space. Swap space priorities are displayed with **swapon -s**, and can be set with the **pri=** mount option. If swap spaces have the same priority, the kernel will write to them round-robin instead of writing to a single swap space until it is at capacity.



## References

**mkswap(8)**, **swapon(8)**, **swapoff(8)**, **mount(8)**, **fdisk(8)** man pages



# Practice: Adding and Enabling Swap Space

## Guided exercise

In this lab, you will create a swap partition and enable it for use.

### Resources:

#### Machines:

serverX

### Outcomes:

Your serverX host will have 500 MiB of swap space running on its second disk.

### Before you begin...

- Log into serverX.
- Switch to **root** using **sudo -i**.

No swap partition was created during the installation of serverX. During peak usage, the server has been running out of physical memory. You have ordered additional RAM and are anxiously waiting for its arrival. In the meantime, you decide to alleviate the problem by enabling swap space on the second disk. To make sure that the newly added swap space is always available for use, you will also need to configure it to be enabled upon boot.

Once you have completed your work, reboot your serverX machine and verify that the swap space is available after the reboot.

- ❑ 1. Create a 500 MiB partition on **/dev/vdb** of type **Linux swap**.

- ❑ 1.1. Use **fdisk** to modify the second disk.

```
[root@serverX ~]# fdisk /dev/vdb
```

- ❑ 1.2. Print the original partition table, then create a new partition that is 500 MiB in size.

```
Command (m for help): p
Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xfd41a9d3

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1             2048        2099199        1048576   83   Linux

Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (2-4, default 2): 2
First sector (2099200-20971519, default 2099200): Enter
```

```
Using default value 2099200
Last sector, +sectors or +size{K,M,G} (2099200-20971519, default
20971519): +500M
Partition 2 of type Linux and of size 500 MiB is set
```

Command (m for help): **p**

```
Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xfd41a9d3
```

Device	Boot	Start	End	Blocks	Id	System
/dev/vdb1		2048	2099199	1048576	83	Linux
/dev/vdb2		2099200	3123199	512000	83	Linux

- 1.3. Set the newly created partition to type **Linux swap**.

```
Command (m for help): t
Partition number (1,2, default 2): 2
Hex code (type L to list all codes): L
```

```
...
1 FAT12 27 Hidden NTFS Win 82 Linux swap / So c1 DRDOS/sec (FAT-
...
```

```
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'
```

Command (m for help): **p**

```
Disk /dev/vdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xfd41a9d3
```

Device	Boot	Start	End	Blocks	Id	System
/dev/vdb1		2048	2099199	1048576	83	Linux
/dev/vdb2		2099200	3123199	512000	82	Linux swap / Solaris

- 1.4. Save the partition table changes.

```
Command (m for help): w
The partition table has been altered!
```

Calling ioctl() to re-read partition table.

```
WARNING: Re-reading the partition table failed with error 16: Device or
resource busy.
```

```
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

- 1.5. Run **partprobe** to make the kernel aware of the partition table change.

```
[root@serverX ~]# partprobe
```

- 2. Initialize the newly created partition as swap space.

```
[root@serverX ~]# mkswap /dev/vdb2
Setting up swspace version 1, size = 511996 KiB
no label, UUID=74f8f3e1-6af3-4e51-9ab5-c48e52bf4a7b
```

- 3. Enable the newly created swap space.

- 3.1. Creating and initializing swap space does not yet enable it for use, as shown by the **free** and **swapon -s** command.

```
[root@serverX ~]# free
```

	total	used	free	shared	buffers	cached
Mem:	1885252	557852	1327400	17096	1080	246040
-/+ buffers/cache:		310732	1574520			
Swap:	0	0	0			

```
[root@serverX ~]# swapon -s
[root@serverX ~]#
```

- 3.2. Enable the newly created swap space.

```
[root@serverX ~]# swapon /dev/vdb2
```

- 3.3. Verify that the newly created swap space is now available.

```
[root@serverX ~]# swapon -s
```

Filename	Type	Size	Used	Priority
/dev/vdb2				
		partition 511996	0	-1

- 3.4. Disable the swap space.

```
[root@serverX ~]# swapoff /dev/vdb2
```

- 3.5. Verify that the swap space is disabled.

```
[root@serverX ~]# swapon -s
[root@serverX ~]#
```

- 4. Configure the new swap space so that it is enabled upon boot.

- 4.1. Determine the UUID of the new swap partition on the second disk.

```
[root@serverX ~]# blkid /dev/vdb2
/dev/vdb2: UUID="74f8f3e1-6af3-4e51-9ab5-c48e52bf4a7b" TYPE="swap"
```

- 4.2. Add an entry to **/etc/fstab**.

```
UUID=74f8f3e1-6af3-4e51-9ab5-c48e52bf4a7b swap swap defaults 0 0
```

- 4.3. Test enabling the swap space using the entry just added to **/etc/fstab**.

```
[root@serverX ~]# swapon -a
```

- 4.4. Verify that the new swap space was enabled.

```
[root@serverX ~]# swapon -s
```

Filename	Type	Size	Used	Priority
/dev/vdb2	partition	511996	0	-1

- 5. Reboot serverX. After the server has rebooted, log in and verify that swap space is enabled.

```
[student@serverX ~]# swapon -s
```

Filename	Type	Size	Used	Priority
/dev/vdb2	partition	511996	0	-1

# Lab: Adding Disks, Partitions, and File Systems to a Linux System

## Performance checklist

In this lab, you will create a GPT partition on a newly allocated disk, format the partition with an XFS file system, and configure the file system for persistent mounting. You will also create two 512 MiB swap partitions. You will configure one of the swap partitions to have a priority of 1.

### Resources:

#### Machines:

serverX

### Outcomes:

- 2 GiB XFS file system on a GPT partition on the second disk. The file system is persistently mounted at **/backup**.
- A 512 MiB swap partition enabled on the second disk with default priority.
- Another 512 MiB swap partition enabled on the second disk with a priority of 1.

### Before you begin...

- Reset your serverX system.
- Log into serverX.
- Switch to **root** using **sudo -i**.

You have been asked to copy important data from the primary disk on serverX to a separate disk for safekeeping. You have been allocated a second disk on serverX for this purpose. You have decided to create a 2 GiB GPT partition on the second disk and format it with the XFS file system. To ensure that this new file system is always available, you will configure it to persistently mount.

To compensate for the shortage of physical memory on serverX, you want to create and enable some swap space for use. You will create two 512 MiB swap partitions on the second disk and set the priority of one of the swap partitions to 1 so that it is preferred over the other swap partition.

Reboot your serverX machine. Verify that the newly created XFS file system is persistently mounted at **/backup**. Also confirm that two swap spaces are activated upon boot, and one of the swap spaces has the default priority of -1 and the other has a priority of 1.

When you have completed your work, run **lab disk grade** on your serverX machine to verify your work.

1. Create a 2 GiB GPT partition on **/dev/vdb** of type **Linux**.
2. Create two 512 MiB partitions on **/dev/vdb** of type **Linux swap**.
3. Format the newly created partitions. Format the 2 GiB partition with an XFS file system. Initialize the two 512 MiB partitions as swap space.
4. Configure the newly created XFS file system to persistently mount at **/backup**.

5. Configure the newly created swap spaces to be enabled at boot. Set one of the swap spaces to be preferred over the other.
6. Reboot serverX. After the server has rebooted, log in and verify that **/dev/vdb1** is mounted at **/backup**. Also verify that two 512 MiB swap partitions are enabled, and that one has default priority and the other has a priority of 1.
7. When you have completed your work, run **lab disk grade** on the serverX machine to verify your work.

## Solution

In this lab, you will create a GPT partition on a newly allocated disk, format the partition with an XFS file system, and configure the file system for persistent mounting. You will also create two 512 MiB swap partitions. You will configure one of the swap partitions to have a priority of 1.

### Resources:

#### Machines:

serverX

### Outcomes:

- 2 GiB XFS file system on a GPT partition on the second disk. The file system is persistently mounted at **/backup**.
- A 512 MiB swap partition enabled on the second disk with default priority.
- Another 512 MiB swap partition enabled on the second disk with a priority of 1.

### Before you begin...

- Reset your serverX system.
- Log into serverX.
- Switch to **root** using **sudo -i**.

You have been asked to copy important data from the primary disk on serverX to a separate disk for safekeeping. You have been allocated a second disk on serverX for this purpose. You have decided to create a 2 GiB GPT partition on the second disk and format it with the XFS file system. To ensure that this new file system is always available, you will configure it to persistently mount.

To compensate for the shortage of physical memory on serverX, you want to create and enable some swap space for use. You will create two 512 MiB swap partitions on the second disk and set the priority of one of the swap partitions to 1 so that it is preferred over the other swap partition.

Reboot your serverX machine. Verify that the newly created XFS file system is persistently mounted at **/backup**. Also confirm that two swap spaces are activated upon boot, and one of the swap spaces has the default priority of -1 and the other has a priority of 1.

When you have completed your work, run **lab disk grade** on your serverX machine to verify your work.

1. Create a 2 GiB GPT partition on **/dev/vdb** of type **Linux**.

- 1.1. Use **gdisk** to modify the second disk.

```
[root@serverX ~]# gdisk /dev/vdb
```

- 1.2. Add a new partition that is 2 GiB in size.

```
Command (? for help): n
Partition number (1-128, default 1): 1
First sector (34-20971486, default = 2048) or {+-}size{KMGT}: Enter
Last sector (2048-20971486, default = 20971486) or {+-}size{KMGT}: +2G
Current type is 'Linux filesystem'
```

1.3. Set the new partition to type **Linux**.

```
Hex code or GUID (L to show codes, Enter = 8300): Enter
Changed type of partition to 'Linux filesystem'
```

2. Create two 512 MiB partitions on **/dev/vdb** of type **Linux swap**.

## 2.1. Add a partition that is 512 MiB.

```
Command (? for help): n
Partition number (2-128, default 2): 2
First sector (34-20971486, default = 4196352) or {+}size{KMGTP}: Enter
Last sector (4196352-20971486, default = 20971486) or {+}size{KMGTP}: +512M
Current type is 'Linux filesystem'
```

2.2. Set the partition to type **Linux swap**.

```
Hex code or GUID (L to show codes, Enter = 8300): L
...
8200 Linux swap          8300 Linux filesystem      8301 Linux reserved
...
Hex code or GUID (L to show codes, Enter = 8300): 8200
Changed type of partition to 'Linux swap'
```

2.3. Add another partition that is 512 MiB, and set its type to **Linux swap**.

```
Command (? for help): n
Partition number (3-128, default 3): 3
First sector (34-20971486, default = 5244928) or {+}size{KMGTP}: Enter
Last sector (5244928-20971486, default = 20971486) or {+}size{KMGTP}: +512M
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 8200
Changed type of partition to 'Linux swap'
```

## 2.4. Verify the partitions.

```
Command (? for help): p
Disk /dev/vdb: 20971520 sectors, 10.0 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): 9918D507-7344-406A-9902-D2503FA028EF
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 20971486
Partitions will be aligned on 2048-sector boundaries
Total free space is 14679997 sectors (7.0 GiB)
```

Number	Start (sector)	End (sector)	Size	Code	Name
1	2048	4196351	2.0 GiB	8300	Linux filesystem
2	4196352	5244927	512.0 MiB	8200	Linux swap
3	5244928	6293503	512.0 MiB	8200	Linux swap

## 2.5. Save the changes to the partition table.

```
Command (? for help): w
```



```
Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!
```

```
Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/vdb.
The operation has completed successfully.
```

2.6. Run **partprobe** to make the kernel aware of the partition table change.

```
[root@serverX ~]# partprobe
```

3. Format the newly created partitions. Format the 2 GiB partition with an XFS file system. Initialize the two 512 MiB partitions as swap space.

3.1. Format the newly created partition with the XFS file system.

```
[root@serverX ~]# mkfs -t xfs /dev/vdb1
meta-data=/dev/vdb1             isize=256    agcount=4, agsize=131072 blks
      =                       sectsz=512   attr=2, projid32bit=1
      =                       crc=0
data     =                       bsize=4096   blocks=524288, imaxpct=25
      =                       sunit=0      swidth=0 blks
naming   =version 2             bsize=4096   ascii-ci=0 ftype=0
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
```

3.2. Initialize the other two partitions as swap space.

```
[root@serverX ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 524284 KiB
no label, UUID=d00554b7-dfac-4034-bdd1-37b896023f2c
```

```
[root@serverX ~]# mkswap /dev/vdb3
Setting up swapspace version 1, size = 524284 KiB
no label, UUID=af30cbb0-3866-466a-825a-58889a49ef33
```

4. Configure the newly created XFS file system to persistently mount at **/backup**.

4.1. Create the **/backup** directory mount point.

```
[root@serverX ~]# mkdir /backup
```

4.2. Determine the UUID of the first partition on the second disk.

```
[root@serverX ~]# blkid /dev/vdb1
/dev/vdb1: UUID="748ca35a-1668-4a2f-bfba-51ebe550f6f0" TYPE="xfs"
PARTLABEL="Linux filesystem" PARTUUID="83b18afb-9c12-48bf-a620-7f8a612df5a8"
```

4.3. Add an entry to **/etc/fstab**.

```
UUID=748ca35a-1668-4a2f-bfba-51ebe550f6f0 /backup xfs defaults 0 2
```

5. Configure the newly created swap spaces to be enabled at boot. Set one of the swap spaces to be preferred over the other.
  - 5.1. Add entries to **/etc/fstab** using the UUIDs generated by the previous **mkswap** steps. Set the priority on one of the swap spaces to 1.

```
UUID=d00554b7-dfac-4034-bdd1-37b896023f2c swap swap defaults 0 0
UUID=af30cbb0-3866-466a-825a-58889a49ef33 swap swap pri=1 0 0
```

6. Reboot serverX. After the server has rebooted, log in and verify that **/dev/vdb1** is mounted at **/backup**. Also verify that two 512 MiB swap partitions are enabled, and that one has default priority and the other has a priority of 1.

```
[student@serverX ~]$ mount | grep ^/
/dev/vda1 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
/dev/vdb1 on /backup type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

```
[student@serverX ~]$ free
              total        used         free       shared    buffers     cached
Mem:          1885252      563528      1321724         17096          696       245224
-/+ buffers/cache:      317608      1567644
Swap:          1048568           0       1048568
```

```
[student@serverX ~]$ swapon -s
Filename                                Type       Size    Used Priority
/dev/vdb2                               partition  524284    0     -1
/dev/vdb3                               partition  524284    0      1
```

7. When you have completed your work, run **lab disk grade** on the serverX machine to verify your work.

```
[student@serverX ~]$ lab disk grade
```

## Summary

### Mounting and Unmounting File Systems

Accessing contents of file systems on internal and external storage devices is important.

### Adding Partitions, File Systems, and Persistent Mounts

- **fdisk** can be used to add, modify, and remove partitions on disks with MBR partitioning schemes.
- **gdisk** can be used to add, modify, and remove partitions on disks with GPT partitioning schemes.
- File systems are created on disk partitions using **mkfs**.
- To make file system mounts persistent, they must be added to **/etc/fstab**.

### Managing Swap Space

Create and activate swap spaces.





## CHAPTER 9

# SERVICE MANAGEMENT AND BOOT TROUBLESHOOTING

Overview	
<b>Goal</b>	To control and monitor system daemons and troubleshoot the Red Hat Enterprise Linux boot process.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• List system daemons and network services started by the systemd service and socket units.</li><li>• Control system daemons and network services using <code>systemctl</code>.</li><li>• Describe the Red Hat Enterprise Linux boot process.</li><li>• Repair common boot issues.</li><li>• Repair file system issues at boot.</li><li>• Repair boot loader problems.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Identifying Automatically Started System Processes (and Practice)</li><li>• Controlling System Services (and Practice)</li><li>• The Red Hat Enterprise Linux Boot Process (and Practice)</li><li>• Repairing Common Boot Issues (and Practice)</li><li>• Repairing File System Issues at Boot (and Practice)</li><li>• Repairing Boot Loader Issues (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Controlling Services and Daemons</li></ul>

# Identifying Automatically Started System Processes

## Objectives

After completing this section, students should be able to list system daemons and network services started by the **systemd** service and socket units.

## Introduction to **systemd**

System startup and server processes are managed by the *systemd System and Service Manager*. This program provides a method for activating system resources, server daemons, and other processes, both at boot time and on a running system.

*Daemons* are processes that 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. By convention, the names of many daemon programs end in the letter "d".

To listen for connections, a daemon uses a *socket*. This is the primary communication channel with local or remote clients. Sockets may be created by daemons or may be separated from the daemon and be created by another process, such as **systemd**. The socket is passed to the daemon when a connection is established by the client.

A *service* 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**).

### A bit of history

For many years, process ID 1 of Linux and UNIX systems has been the **init** process. This process was responsible for activating other services on the system and is the origin of the term "init system." Frequently used daemons were started on systems at boot time with *System V* and *LSB* init scripts. These are shell scripts, and may vary from one distribution to another. Less frequently used daemons were started on demand by another service, such as **initd** or **xinetd**, which listens for client connections. These systems have several limitations, which are addressed with **systemd**.

In Red Hat Enterprise Linux 7, process ID 1 is **systemd**, the new init system. A few of the new features provided by **systemd** include:

- Parallelization capabilities, 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, such as by not starting a network service when the network is not available.
- A method of tracking related processes together by using Linux control groups.



## Note

With **systemd**, shell-based service scripts are used only for a few legacy services. Therefore, configuration files with shell variables, such as those found in **/etc/sysconfig**, are being replaced. Those still in use are included as **systemd** environment files and read as **NAME=VALUE** pairs. They are no longer sourced as a shell script.

## **systemctl** and **systemd** units

The **systemctl** command is used to manage different types of **systemd** objects, called *units*. A list of available unit types can be displayed with **systemctl -t help**.



## Important

The **systemctl** may abbreviate or "ellipsize" unit names, process tree entries, and unit descriptions unless run with the **-l** option.

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. Control of the socket will be passed to a daemon or newly started service when a client connection is made. Socket units are used to delay the start of a service at boot time and to start less frequently used services on demand. These are similar in principle to services which use the **xinetd** superserver to start 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.

## Service states

The status of a service can be viewed 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@serverX ~]# systemctl status sshd.service
sshd.service - OpenSSH server daemon
Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)
Active: active (running) since Thu 2014-02-27 11:51:39 EST; 7h ago
Main PID: 1073 (sshd)
CGroup: /system.slice/sshd.service
        └─1073 /usr/sbin/sshd -D

Feb 27 11:51:39 server0.example.com systemd[1]: Started OpenSSH server daemon.
Feb 27 11:51:39 server0.example.com sshd[1073]: Could not load host key: /et...y
Feb 27 11:51:39 server0.example.com sshd[1073]: Server listening on 0.0.0.0 ....
Feb 27 11:51:39 server0.example.com sshd[1073]: Server listening on :: port 22.
Feb 27 11:53:21 server0.example.com sshd[1270]: error: Could not load host k...y
Feb 27 11:53:22 server0.example.com sshd[1270]: Accepted password for root f...2
Hint: Some lines were ellipsized, use -l to show in full.
```

Several keywords indicating the state of the service can be found in the status output:

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	Will be started at boot time.
disabled	Will not be started at boot time.
static	Can not 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 previous versions of Red Hat Enterprise Linux.

## Listing unit files with systemctl

In this example, please follow along with the next steps while your instructor demonstrates obtaining status information of services.



### Note

Notice that the **systemctl** command will automatically paginate the output with **less**.

1. Query the state of all units to verify a system startup.

```
[root@serverX ~]# systemctl
```

2. Query the state of only the service units.

```
[root@serverX ~]# systemctl --type=service
```

3. Investigate any units which are in a failed or maintenance state. Optionally, add the **-l** option to show the full output.

```
[root@serverX ~]# systemctl status rngd.service -l
```

4. The **status** argument may also be used to determine if a particular unit is active and show if the unit is enabled to start at boot time. Alternate commands can also easily show the active and enabled states:

```
[root@serverX ~]# systemctl is-active sshd
[root@serverX ~]# systemctl is-enabled sshd
```



5. List the active state of all loaded units. Optionally, limit the type of unit. The **--all** option will add inactive units.

```
[root@serverX ~]# systemctl list-units --type=service
[root@serverX ~]# systemctl list-units --type=service --all
```

6. View the enabled and disabled settings for all units. Optionally, limit the type of unit.

```
[root@serverX ~]# systemctl list-unit-files --type=service
```

7. View only failed services.

```
[root@serverX ~]# systemctl --failed --type=service
```



## References

**systemd(1)**, **systemd.unit(5)**, **systemd.service(5)**, **systemd.socket(5)**, and **systemctl(1)** man pages

Additional information may be available in the chapter on managing services with **systemd** in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

## Practice: Identify the Status of systemd Units

### Guided exercise

In this lab, you will identify installed and running services on the system.

#### Outcomes:

A list of active and enabled services on the system.

#### *Before you begin...*

Reset your serverX system.

- ☐ 1. List all service units on the system.

```
[student@serverX ~]$ sudo systemctl list-units --type=service
```

- ☐ 2. List all socket units, active and inactive, on the system.

```
[student@serverX ~]$ sudo systemctl list-units --type=socket --all
```

- ☐ 3. Explore the status of the **chronyd** service. This service is used for network time synchronization (NTP).

- ☐ 3.1. Display the status of the **chronyd** service. Note the process ID of any active daemons.

```
[student@serverX ~]$ sudo systemctl status chronyd
```

- ☐ 3.2. Confirm that the listed daemons are running.

```
[student@serverX ~]$ ps -p PID
```

- ☐ 4. Explore the status of the **sshd** service. This service is used for secure encrypted communication between systems.

- ☐ 4.1. Determine if the **sshd** service is enabled to start at system boot.

```
[student@serverX ~]$ sudo systemctl is-enabled sshd
```

- ☐ 4.2. Determine if the **sshd** service is active without displaying all of the status information.

```
[student@serverX ~]$ sudo systemctl is-active sshd
```

- ☐ 4.3. Display the status of the **sshd** service.

```
[student@serverX ~]$ sudo systemctl status sshd
```

- 5. List the enabled or disabled states of all service units.

```
[student@serverX ~]$ sudo systemctl list-unit-files --type=service
```

# Controlling System Services

## Objectives

After completing this section, students should be able to control system daemons and network services using **systemctl**.

## Starting and stopping system daemons on a running system

Changes to a configuration file or other updates to a service may require that the service be restarted. A service that is no longer used may be stopped before removing the software. A service that is not frequently used may be manually started by an administrator only when it is needed.

In this example, please follow along with the next steps while your instructor demonstrates managing services on a running system.

1. View the status of a service.

```
[root@serverX ~]# systemctl status sshd.service
```

2. Verify that the process is running.

```
[root@serverX ~]# ps -up PID
```

3. Stop the service and verify the status.

```
[root@serverX ~]# systemctl stop sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

4. Start the service and view the status. The process ID has changed.

```
[root@serverX ~]# systemctl start sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

5. Stop, then start, the service in a single command.

```
[root@serverX ~]# systemctl restart sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

6. Issue instructions for a service to read and reload its configuration file without a complete stop and start. The process ID will not change.

```
[root@serverX ~]# systemctl reload sshd.service  
[root@serverX ~]# systemctl status sshd.service
```

### Unit dependencies

Services may be started as dependencies of other services. If a socket unit is enabled and the service unit with the same name is not, the service will automatically be started when a request is made on the network socket. Services may also be triggered by path units when a file system condition is met. For example, a file placed into the print spool directory will cause the **cups** print service to be started if it is not running.

```
[root@serverX ~]# 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 will disable the dependencies.

The **systemctl list-dependencies UNIT** command can be used to print out a tree of what other units must be started if the specified unit is started. Depending on the exact dependency, the other unit may need to be running before or after the specified unit starts. The **--reverse** option to this command will show what units need to have the specified unit started in order to run.

### Masking services

At times, a system may have conflicting services installed. For example, there are multiple methods to manage networks (network and NetworkManager) and firewalls (iptables and firewalld). To prevent an administrator from accidentally starting a service, that service may be *masked*. Masking will create a link in the configuration directories so that if the service is started, nothing will happen.

```
[root@serverX ~]# systemctl mask network
ln -s '/dev/null' '/etc/systemd/system/network.service'
[root@serverX ~]# systemctl unmask network
rm '/etc/systemd/system/network.service'
```



### Important

A disabled service will not be started automatically at boot or by other unit files, but can be started manually. A masked service can not be started manually or automatically.

## Enabling system daemons to start or stop at boot

Starting a service on a running system does not guarantee that the service will be started when the system reboots. Similarly, stopping a service on a running system will not keep it from starting again when the system reboots. Services are started at boot time when links are created in the appropriate **systemd** configuration directories. These links are created and removed with **systemctl** commands.

In this example, please follow along with the next steps while your instructor demonstrates enabling and disabling services.

1. View the status of a service.

```
[root@serverX ~]# systemctl status sshd.service
```

2. Disable the service and verify the status. Note that disabling a service does not stop the service.

```
[root@serverX ~]# systemctl disable sshd.service
[root@serverX ~]# systemctl status sshd.service
```

3. Enable the service and verify the status.

```
[root@serverX ~]# systemctl enable sshd.service
[root@serverX ~]# systemctl is-enabled sshd.service
```

## Summary of systemctl commands

Services can be started and stopped on a running system and enabled or disabled for automatic start at boot time.

Task:	Command:
View detailed information about a unit state.	<b>systemctl status <i>UNIT</i></b>
Stop a service on a running system.	<b>systemctl stop <i>UNIT</i></b>
Start a service on a running system.	<b>systemctl start <i>UNIT</i></b>
Restart a service on a running system.	<b>systemctl restart <i>UNIT</i></b>
Reload configuration file of a running service.	<b>systemctl reload <i>UNIT</i></b>
Completely disable a service from being started, both manually and at boot.	<b>systemctl mask <i>UNIT</i></b>
Make a masked service available.	<b>systemctl unmask <i>UNIT</i></b>
Configure a service to start at boot time.	<b>systemctl enable <i>UNIT</i></b>
Disable a service from starting at boot time.	<b>systemctl disable <i>UNIT</i></b>
List units which are required and wanted by the specified unit.	<b>systemctl list-dependencies <i>UNIT</i></b>



## References

**systemd(1)**, **systemd.unit(5)**, **systemd.service(5)**, **systemd.socket(5)**, and **systemctl(1)** man pages

Additional information may be available in the chapter on managing services with **systemd** in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

# Practice: Using **systemctl** to Manage Services

## Guided exercise

In this lab, you will manage a service unit that is already installed on the system.

### Outcomes:

The **chronyd** service is disabled and no longer running on the system.

### *Before you begin...*

Reset your serverX system.

- ❑ 1. Observe the results of **systemctl restart** and **systemctl reload** commands.

- ❑ 1.1. Display the status of the **sshd** service. Note the process ID of the daemon.

```
[student@serverX ~]$ sudo systemctl status sshd
```

- ❑ 1.2. Restart the **sshd** service and view the status. The process ID of the daemon has changed.

```
[student@serverX ~]$ sudo systemctl restart sshd  
[student@serverX ~]$ sudo systemctl status sshd
```

- ❑ 1.3. Reload the **sshd** service and view the status. The process ID of the daemon has not changed and connections have not been interrupted.

```
[student@serverX ~]$ sudo systemctl reload sshd  
[student@serverX ~]$ sudo systemctl status sshd
```

- ❑ 2. Verify that the **chronyd** service is running.

```
[student@serverX ~]$ sudo systemctl status chronyd
```

- ❑ 3. Stop the **chronyd** service and view the status.

```
[student@serverX ~]$ sudo systemctl stop chronyd  
[student@serverX ~]$ sudo systemctl status chronyd
```

- ❑ 4. Determine if the **chronyd** service is enabled to start at system boot.

```
[student@serverX ~]$ sudo systemctl is-enabled chronyd
```

- ❑ 5. Reboot the system, then view the status of the **chronyd** service.

```
[student@serverX ~]$ sudo systemctl status chronyd
```

- ❑ 6. Disable the **chronyd** service so that it does not start at system boot, then view the status of the service.

```
[student@serverX ~]$ sudo systemctl disable chronyd  
[student@serverX ~]$ sudo systemctl status chronyd
```

- ❑ 7. Reboot the system, then view the status of the **chronyd** service.

```
[student@serverX ~]$ sudo systemctl status chronyd
```



# The Red Hat Enterprise Linux Boot Process

## Objectives

After completing this section, students should be able to describe and influence the Red Hat Enterprise Linux boot process.

## The Red Hat Enterprise Linux 7 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 (graphical) 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 7. The list for **x86\_64** virtual machines is roughly the same, but some of the hardware-specific steps are handled in software by the hypervisor.

1. The machine is powered on. The system firmware (either modern UEFI or more old-fashioned BIOS) runs a *Power On Self Test* (POST), and starts to initialize some of the hardware.

Configured using: The system BIOS/UEFI configuration screens, typically reached by pressing a certain key combination—e.g., **F2**—early during the boot process.

2. 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/UEFI configuration screens, typically reached by pressing a certain key combination—e.g., **F2**—early during the boot process.

3. The system firmware reads a *boot loader* from disk, then passes control of the system to the boot loader. On a Red Hat Enterprise Linux 7 system, this will typically be **grub2**.

Configured using: **grub2-install**

4. The boot loader loads its configuration from disk, and presents the user with a menu of possible configurations to boot.

Configured using: **/etc/grub.d/**, **/etc/default/grub**, and (not manually) **/boot/grub2/grub.cfg**.

5. After the user has made a choice (or an automatic timeout has happened), the boot loader loads the configured kernel and *initramfs* from disk and places them in memory. An *initramfs* is a **gzip-ed cpio** archive containing kernel modules for all hardware necessary at boot, init scripts, and more. On Red Hat Enterprise Linux 7, the *initramfs* contains an entire usable system by itself.

Configured using: **/etc/dracut.conf**

6. The boot loader hands control of the system 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: `/etc/grub.d/`, `/etc/default/grub`, and (not manually) `/boot/grub2/grub.cfg`.

7. 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 7, the **initramfs** contains a working copy of **systemd** as `/sbin/init`, as well as a **udev** daemon.

Configured using: **init=** command-line parameter.

8. The **systemd** instance from the **initramfs** executes all units for the **initrd.target** target. This includes mounting the actual root file system on `/sysroot`.

Configured using: `/etc/fstab`

9. The kernel root file system is switched (pivoted) from the **initramfs** root file system to the system root file system that was previously mounted on `/sysroot`. **systemd** then re-executes itself using the copy of **systemd** installed on the system.
10. **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 its essence, a **systemd** target is a set of units that should be activated to reach a desired system state. These targets will typically include at least a text-based login or a graphical login screen being spawned.

Configured using: `/etc/systemd/system/default.target`, `/etc/systemd/system/`

## Boot, reboot, and shut down

To power off or reboot a running system from the command line, administrators can use the **systemctl** command.

**systemctl poweroff** will stop all running services, unmount all file systems (or remount them read-only when they cannot be unmounted), and then power down the system.

**systemctl reboot** will stop all running services, unmount all file systems, and then reboot the system.

For the ease of backward compatibility, the **poweroff** and **reboot** commands still exist, but in Red Hat Enterprise Linux 7 they are symbolic links to the **systemctl** tool.



### Important

**systemctl halt** and **halt** are also available to stop the system, but unlike their **poweroff** equivalents, these commands do *not* power off the system; they bring a system down to a point where it is safe to manually power it off.

## Selecting a systemd target

A **systemd** target is a set of **systemd** units that should be started to reach a desired state. The most important of these targets are listed in the following table.

Target	Purpose
<b>graphical.target</b>	System supports multiple users, graphical and text-based logins.
<b>multi-user.target</b>	System supports multiple users, text-based logins only.
<b>rescue.target</b>	<b>sulogin</b> prompt, basic system initialization completed.
<b>emergency.target</b>	<b>sulogin</b> prompt, <b>initramfs</b> pivot complete and system root mounted on / read-only. <i>SWAP is INIT-RAU</i>

It is possible for a target to be a part of another target; for example, the **graphical.target** includes **multi-user.target**, which in turn depends on **basic.target** and others. These dependencies can be viewed from the command line with the following command:

```
[root@serverX ~]# systemctl list-dependencies graphical.target | grep target
```

An overview of all available targets can be viewed with:

```
[root@serverX ~]# systemctl list-units --type=target --all
```

An overview of all targets installed on disk can be viewed with:

```
[root@serverX ~]# systemctl list-unit-files --type=target --all
```

### Selecting a target at runtime

On a running system, administrators can choose to switch to a different target using the **systemctl isolate** command; for example:

```
[root@serverX ~]# systemctl isolate multi-user.target
```

Isolating a target will stop all services not required by that target (and its dependencies), and start any required services that have not yet been started.



### Note

Not all targets can be isolated. Only targets that have **AllowIsolate=yes** set in their unit files can be isolated; for example, the **graphical.target** target can be isolated, but the **cryptsetup.target** target cannot.

### Setting a default target

When the system starts, and control is passed over to **systemd** from the **initramfs**, **systemd** will try to activate the **default.target** target. Normally the **default.target** target will be a symbolic link (in **/etc/systemd/system/**) to either **graphical.target** or **multi-user.target**.

Instead of editing this symbolic link by hand, the **systemctl** tool comes with two commands to manage this link: **get-default** and **set-default**.

```
[root@serverX ~]# systemctl get-default
```

```
multi-user.target
[root@serverX ~]# systemctl set-default graphical.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/graphical.target' '/etc/systemd/system/default.target'
[root@serverX ~]# systemctl get-default
graphical.target
```

### Selecting a different target at boot time

To select a different target at boot time, a special option can be appended to the kernel command line from the boot loader: **systemd.unit=**.

For example, to boot the system into a rescue shell where configuration changes can be made without (almost) any service running, the following can be appended from the interactive boot loader menu before starting:

```
systemd.unit=rescue.target
```

This configuration change will only affect 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 for Red Hat Enterprise Linux 7 systems:

1. (Re)boot the system.
2. Interrupt the boot loader menu countdown by pressing any key.
3. Move the cursor to the entry to be started.
4. Press **e** to edit the current entry.
5. Move the cursor to the line that starts with **linux16**. This is the kernel command line.
6. Append **systemd.unit=desired.target**.
7. Press **Ctrl+x** to boot with these changes.



## References

**bootup(7)**, **dracut.bootup(7)**, **systemd.target(5)**, **systemd.special(7)**, **sulogin(8)**, and **systemctl(1)** man pages

**info grub2** (*GNU GRUB Manual*)

# Practice: Selecting a Boot Target

## Guided exercise

In this lab, you will configure your **serverX** system to boot into different targets.

### Resources:

#### Machines:

serverX

### Outcomes:

A system booted into different targets.

### Before you begin...

- Reset your **serverX** system.

- On your **serverX** system, switch to the **multi-user** target manually without rebooting.

- ```
[student@serverX ~]$ sudo systemctl isolate multi-user.target
```

- Log into a text-based console as **root**.

- Configure your **serverX** to automatically boot into the **multi-user** target after a reboot, then reboot your **serverX** system to verify.

- ```
[root@serverX ~]# systemctl set-default multi-user.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/multi-user.target' '/etc/systemd/system/default.target'
```

- ```
[root@serverX ~]# systemctl reboot
```

- Reboot your **serverX** system, then from within the boot loader menu, boot into the **rescue** target.

- Reboot your **serverX** machine.

- ```
[root@serverX ~]# systemctl reboot
```

- Interrupt the boot loader when the menu appears by pressing any key.
- Move the selection to the default entry (the first one) using the cursor keys.
- Press **e** to edit the current entry.
- Move the cursor to the line that starts with **linux16**.
- Move the cursor to the end of the line (using the **End** key), and append the following text:

```
systemd.unit=rescue.target
```

- 4.7. Press **Ctrl+x** to boot using the modified configuration.
- 4.8. When prompted for the **root** password, enter **redhat**.
- 5. Set the default **systemd** target back to the graphical target.

```
[root@serverX ~]# systemctl set-default graphical.target
```

- 6. Press **Ctrl+d** to continue booting into the (new) default target.

# Repairing Common Boot Issues

## Objectives

After completing this section, students should be able to repair common boot issues.

## Recovering the root password

One task that every system administrator should be able to accomplish is recovering 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.

A number of 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 will explore a method that does not require the use of external media.



### Note

On Red Hat Enterprise Linux 6 and earlier, an administrator could boot the system into *runlevel 1*, and be presented with a root prompt. The closest analogs to runlevel 1 on a Red Hat Enterprise Linux 7 machine are the **rescue.target** and **emergency.target** targets, both of which require the **root** password to log in.

On Red Hat Enterprise Linux 7, 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. While this is mostly meant for debugging, it can also be used to recover a lost **root** password:

1. Reboot the system.
2. Interrupt the boot loader countdown by pressing any key.
3. Move the cursor to the entry that needs to be booted.
4. Press **e** to edit the selected entry.
5. Move the cursor to the kernel command line (the line that starts with **linux16**).
6. Append **rd.break** (this will break just before control is handed from the **initramfs** to the actual system).



### Note

The **initramfs** prompt will show up on whatever console is specified *last* on the kernel commandline.

7. Press **Ctrl+x** to boot with the changes.

At this point, a **root** shell will be presented, with the root file system for the actual system mounted read-only on **/sysroot**.



## Important

SELinux is not yet enabled at this point, so any new files being created will not have an SELinux context assigned to them. Keep in mind that some tools (such as **passwd**) first create a new file, then move it in place of the file they are intended to edit, effectively creating a new file without an SELinux context.

To recover 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-4.2# passwd root
```

4. Make sure that all unlabeled files (including **/etc/shadow** at this point) get relabeled during boot.

```
sh-4.2# touch /.autorelabel
```

5. Type **exit** twice. The first will exit the **chroot** jail, and the second will exit the **initramfs** debug shell.

At this point, the system will continue booting, perform a full SELinux relabel, then reboot again.

## Using journalctl

It can be useful to look at the logs of previous (failed) boots. If the **journald** log has been made persistent, this can be done with the **journalctl** tool.

First make sure that you have persistent **journald** logging enabled:

```
[root@serverX ~]# mkdir -p -m2775 /var/log/journal
[root@serverX ~]# chown :systemd-journal /var/log/journal
[root@serverX ~]# killall -USR1 systemd-journald
```

To inspect the log files for a previous boot, use the **-b** option to **journalctl**. Without any arguments, the **-b** option will filter output only to messages pertaining to this boot, but with a negative number as an argument, it will filter on previous boots. For example:

```
[root@serverX ~]# journalctl -b-1 -p err
```



This command will show all messages rated as an error or worse from the previous boot.

## Diagnose and repair systemd boot issues

If there are problems during the starting of services, there are a couple of tools available to system administrators that can help with debugging and/or troubleshooting:

### Early debug shell

By running **systemctl enable debug-shell.service**, a **root** shell will be spawned on **TTY9 (Ctrl+Alt+F9)** early during the boot sequence. This shell is automatically logged in as **root**, so that an administrator can use some of the other debugging tools while the system is still booting.



### Warning

Do not forget to disable the **debug-shell.service** service when you are done debugging, as it leaves an unauthenticated root shell open to anyone with local console access.

### 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 will spawn into a special 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 **rescue.target** waits for **sysinit.target** to complete first so that more of the system will be initialized (e.g., logging, file systems, etc.).

These shells can be used 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 will continue with the regular boot process.

### Stuck jobs

During startup, **systemd** spawns a number of jobs. If some of these jobs cannot complete, they will block other jobs from running. To inspect the current job list, an administrator can use the command **systemctl list-jobs**. Any jobs listed as **running** must complete before the jobs listed as **waiting** can continue.



### References

**dracut.cmdline(7)**, **systemd-journald(8)**, **journalctl(1)**, **sushell(8)**, and **systemctl(1)** man pages

**/usr/lib/systemd/system/debug-shell.service**

## Practice: Resetting a Lost root Password

### Guided exercise

In this lab, you will recover a lost root password.

#### Resources:

#### Machines:

serverX

#### Outcomes:

A recovered root password.

#### Before you begin...

- Reset your **serverX** system.
- Log in and set up your **serverX** system:

```
[student@serverX ~]$ lab rootpw setup
```

The **lab rootpw setup** script has just reset your root password to a random string and rebooted your system. Without using **sudo**, break into your own system and reset the **root** password back to **redhat**.

- ❑ 1. Reboot your system, and interrupt the countdown in the boot loader menu.
  - ❑ 1.1. 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.
- ❑ 2. Edit the default boot loader entry (in memory) to abort the boot process just after all file systems have been mounted, but before control is handed over to **systemd**, then 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. Using the cursor keys, navigate to the line that starts with **linux16**.
  - ❑ 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 config.
- ❑ 3. At the **switch\_root** prompt, remount the **/sysroot** file system read-write, then use **chroot** to go into a **chroot** jail at **/sysroot**.
  - ❑ 3.1.
 

```
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
```
- ❑ 4. Change the **root** password back to **redhat**.

❑ 4.1. `sh-4.2# echo redhat | passwd --stdin root`

- ❑ 5. Configure the system to automatically perform a full SELinux relabel after boot. This is necessary since the **passwd** tool re-created the **/etc/shadow** file without an SELinux context.

❑ 5.1. `sh-4.2# touch /.autorelabel`

- ❑ 6. Type **exit** twice to continue booting your system as normal. The system will run an SELinux relabel, then reboot again by itself.
- ❑ 7. Verify your work by running the following command:

`[student@serverX ~]$ lab rootpw grade`

## Repairing File System Issues at Boot

### Objectives

After completing this section, students should be able to repair file system issues during boot.

Errors in **/etc/fstab** and corrupt file systems can stop a system from booting. In most cases, **systemd** will actually continue to boot after a timeout, or drop to an emergency repair shell that requires the **root** password.

The following table lists some common errors and their results.

Problem	Result
Corrupt file system	<b>systemd</b> will attempt a <b>fsck</b> . If the problem is too serious for an automatic fix, the user will be prompted to run <b>fsck</b> manually from an emergency shell.
Non-existent device/UUID referenced in <b>/etc/fstab</b>	<b>systemd</b> will wait for a set amount of time, waiting for the device to become available. If the device does not become available, the user is dropped to an emergency shell after the timeout.
Non-existent mount point in <b>/etc/fstab</b>	<b>systemd</b> creates the mount point if possible; otherwise, it drops to an emergency shell.
Incorrect mount option specified in <b>/etc/fstab</b>	The user is dropped to an emergency shell.

In all cases, an administrator can also utilize the **emergency.target** target to diagnose and fix the issue, since no file systems will be mounted before the emergency shell is displayed.



### Note

When using the automatic recovery shell during file system issues, do not forget to issue a **systemctl daemon-reload** after editing **/etc/fstab**. Without this reload, **systemd** will continue using the old version.



### References

**systemd-fsck(8)**, **systemd-fstab-generator(3)**, and **systemd.mount(5)** man pages

# Practice: Repairing Boot Problems

## Guided exercise

In this lab, you will recover from an error in `/etc/fstab`.

### Resources:

#### Machines:

serverX

### Outcomes:

After completing this exercise, your machine should boot normally again, without user intervention.

### Before you begin...

- Reset your **serverX** system.
- Log in and set up your **serverX** system:

```
[student@serverX ~]$ lab bootbreakfs setup
```

You *had* a new admin in your team, but it was decided that it would be in everybody's best interest if that admin pursued a different career.

Now that your staffing issue has been solved, there are a couple of remaining issues. One of them is a machine that had been "fixed" by this admin.

- ❑ 1. Take a good look at the console of your **serverX** machine. It seems it is stuck early on.

Take a minute to speculate about a possible cause for this behavior, then reboot the machine and interrupt the boot loader menu countdown. (If you wait long enough, the system will eventually spawn a rescue shell by itself, but that can take a while.)

- ❑ 1.1. Usually you would send a **Ctrl+Alt+Del** to your system to reboot it. This particular boot problem causes that key sequence to retry the boot sequence again without rebooting. In this case, either wait for the task to timeout or use the power switch to force a reboot.
- ❑ 1.2. When the boot loader menu appears after the BIOS self-test, press any key to interrupt the countdown.
- ❑ 2. Looking at the error you had during the previous boot, it appears that at least parts of the system are still functioning. Since 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. Using the cursor keys, navigate to the line that starts with **linux16**.
  - ❑ 2.4. Press **End** to move the cursor to the end of the line.

- ❑ 2.5. Append **emergency** to the end of the line.
- ❑ 2.6. Press **Ctrl+x** to boot using the modified config.
- ❑ 3. Log into the emergency mode. Pay close attention to any errors you might receive.
  - ❑ 3.1. At the **Give root password for maintenance** prompt, enter **redhat**.
- ❑ 4. Inspect what file systems are currently mounted.
  - ❑ 4.1. 

```
[root@localhost ~]# mount
...
/dev/vda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
```
- ❑ 5. It appears that the root file system is mounted read-only; mount it read-write.
  - ❑ 5.1. 

```
[root@localhost ~]# mount -o remount,rw /
```
- ❑ 6. Attempt to mount all the other file systems:
  - ❑ 6.1. 

```
[root@localhost ~]# mount -a
mount: mount point /RemoveMe does not exist
```
- ❑ 7. Open **/etc/fstab** in an editor and fix the issue.
  - ❑ 7.1. 

```
[root@localhost ~]# vi /etc/fstab
```
  - ❑ 7.2. Remove the invalid line (the one with **RemoveMe**).
  - ❑ 7.3. Save your changes, then exit your editor.
- ❑ 8. Verify that your **/etc/fstab** is now correct by attempting to mount all entries.
  - ❑ 8.1. 

```
[root@localhost ~]# mount -a
```
- ❑ 9. Exit your **emergency** shell and reboot the system by typing **reboot**. Your system should now boot normally.

# Repairing Boot Loader Issues

## Objectives

After completing this section, students should be able to fix boot loader issues.

The boot loader used by default on Red Hat Enterprise Linux 7 is **grub2**, the second major version of the *GRand Unified Bootloader*.

**grub2** can be used to boot on both BIOS and UEFI systems, and supports booting almost any operating system that runs on modern hardware.

The main configuration file for **grub2** is **/boot/grub2/grub.cfg**, but administrators are not supposed to edit this file directly. Instead, a tool called **grub2-mkconfig** is used to generate that configuration using a set of different configuration files, and the list of installed kernels.

**grub2-mkconfig** will look at **/etc/default/grub** for options such as the default menu timeout and kernel command line to use, then use a set of scripts in **/etc/grub.d/** to generate a configuration file.

To make permanent changes to the boot loader configuration, an administrator needs to edit the configuration files listed previously, then run the following command:

```
[root@serverX ~]# grub2-mkconfig > /boot/grub2/grub.cfg
```

In those cases where major changes have been made, an administrator might prefer to run that command without the redirection so that the results can be inspected first.

### Important directives

To troubleshoot a broken **grub2** configuration, an administrator will need to understand the syntax of **/boot/grub2/grub.cfg** first. Actual bootable entries are encoded inside **menuentry** blocks. In these blocks, **linux16** and **initrd16** lines point to the kernel to be loaded from disk (along with the kernel command line) and the **initramfs** to be loaded. During interactive editing at boot, **Tab** completion is available to find these files.

The **set root** lines inside those blocks do not point to the root file system for the Red Hat Enterprise Linux 7 system, but instead point to the file system from which **grub2** should load the kernel and initramfs files. The syntax is **harddrive,partition**, where **hd0** is the first hard drive in the system, **hd1** is the second, etc. The partitions are indicated as **msdos1** for the first MBR partition, or **gpt1** for the first GPT partition on that drive.

### Reinstalling the boot loader

In those cases where the boot loader itself has become corrupted, it can be reinstalled using the **grub2-install** command. On BIOS systems, the disk where **grub2** should be installed in the MBR should be provided as an argument. On UEFI systems, no arguments are necessary when the EFI system partition is mounted on **/boot/efi**.



## References

**info grub2** (*GNU GRUB Manual*)

**info grub2-install** (*GNU GRUB Manual*)

- Chapter 28: "Invoking **grub2-install**"



# Practice: Repairing a Boot Loader Problem

## Guided exercise

In this lab, you will repair an issue with the boot loader configuration on one of your machines.

### Resources:

#### Machines:

serverX

### Outcomes:

A machine that boots normally without user intervention.

#### Before you begin...

- Reset your **serverX** system.
- Log in and set up your **serverX** system:

```
[student@serverX ~]$ lab bootbreakgrub setup
```

One of your *former* co-workers was experimenting with speeding up the boot process on one of your machines. After a number of failed attempts, you have now been tasked with repairing the damage done.

- ❑ 1. Look at the console of your **serverX** machine, then reboot the machine and interrupt the boot loader countdown timer.
  - ❑ 1.1. 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.
- ❑ 2. Move the cursor to the default boot entry, then press **e** to edit that entry. Inspect the configuration closely, looking for anything that seems out of the ordinary.
- ❑ 3. Find the line that is blocking the boot process, modify it, then boot with these changes.
  - ❑ 3.1. **os16** is not a valid **grub** directive. Change it to **linux16**.
  - ❑ 3.2. Press **Ctrl+x** to boot your system with the modified configuration.
- ❑ 4. Wait for the system to boot, log in as **student**, elevate your privileges to **root**, then generate a new **grub2** configuration. Do not immediately overwrite the existing configuration, but inspect the new config first.

```
❑ 4.1. [student@serverX ~]$ sudo -i
[student@serverX ~]$ sudo -i
[root@serverX ~]# grub2-mkconfig
```

- ❑ 4.2. Scroll through the output to see if it looks like a valid **grub2** configuration.
- ❑ 4.3. Commit the configuration to disk.

```
[root@serverX ~]# grub2-mkconfig > /boot/grub2/grub.cfg
```

- 5. Reboot your machine, and check if it boots normally again without user intervention.

- 5.1. 

```
[root@serverX ~]# systemctl reboot
```

## Lab: Controlling Services and Daemons

### Performance checklist

In this lab, you will manage a service unit that is already installed on the system.

#### Outcomes:

The **psacct** service is enabled and running on the system, and the **rsyslog** service is disabled and no longer running on the system.

#### *Before you begin...*

Reset your serverX system.

1. Start the **psacct** service.
2. Configure the **psacct** service so that it starts at system boot.
3. Stop the **rsyslog** service.
4. Configure the **rsyslog** service so that it does not start at system boot.
5. Reboot the system, then run **lab services grade** to verify the configuration.

## Solution

In this lab, you will manage a service unit that is already installed on the system.

### Outcomes:

The **psacct** service is enabled and running on the system, and the **rsyslog** service is disabled and no longer running on the system.

### *Before you begin...*

Reset your serverX system.

1. Start the **psacct** service.

```
[student@serverX ~]$ sudo systemctl start psacct
[student@serverX ~]$ sudo systemctl status psacct
```

2. Configure the **psacct** service so that it starts at system boot.

```
[student@serverX ~]$ sudo systemctl enable psacct
[student@serverX ~]$ sudo systemctl status psacct
```

3. Stop the **rsyslog** service.

```
[student@serverX ~]$ sudo systemctl stop rsyslog
[student@serverX ~]$ sudo systemctl status rsyslog
```

4. Configure the **rsyslog** service so that it does not start at system boot.

```
[student@serverX ~]$ sudo systemctl disable rsyslog
[student@serverX ~]$ sudo systemctl status rsyslog
```

5. Reboot the system, then run **lab services grade** to verify the configuration.

```
[student@serverX ~]$ lab services grade
```

## Summary

### Identifying Automatically Started System Processes

Determine the status of system daemons and network services started by **systemd**.

### Controlling System Services

Start, stop, and enable services using **systemctl**.

### The Red Hat Enterprise Linux Boot Process

- The Red Hat Enterprise Linux 7 boot process can be broken down into four steps:
  1. Hardware (BIOS/UEFI)
  2. Boot loader (**grub2**)
  3. **kernel** and **initramfs**
  4. **systemd**
- **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.
- **systemd.unit=** on the kernel command line selects a different target at boot.

### Repairing Common Boot Issues

- Use **rd.break** on the kernel command line to interrupt the boot process before control is handed over from the **initramfs**. The system will be mounted (read-only) under **/sysroot**.
- **journalctl** can be used to filter for specific boots with the **-b** option.
- The **debug-shell.service** service can be used to get an automatic **root** shell early during boot.

### Repairing File System Issues at Boot

- **systemd** will display an emergency shell in most cases dealing with file system issues.
- The **emergency.target** target can also be used to diagnose and fix file system issues.

### Repairing Boot Loader Issues

- Use **e** and **Ctrl+x** to edit boot loader entries in memory, then boot.
- Use **grub2-mkconfig > /boot/grub2/grub.cfg** to regenerate the boot loader configuration.
- **grub2-install** is used to reinstall the boot loader.





## CHAPTER 10

# NETWORK CONFIGURATION

Overview	
<b>Goal</b>	To configure basic IPv4 networking on Red Hat Enterprise Linux systems.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Test and review current network configuration with basic utilities.</li><li>• Manage network settings and devices with <code>nmcli</code> and NetworkManager.</li><li>• Modify network settings by editing the configuration files.</li><li>• Configure and test system hostname and name resolution.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Validating Network Configuration (and Practice)</li><li>• Configuring Networking with <code>nmcli</code> (and Practice)</li><li>• Editing Network Configuration Files (and Practice)</li><li>• Configuring Host Names and Name Resolution</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Managing Red Hat Enterprise Linux Networking</li></ul>

# Validating Network Configuration

## Objectives

After completing this section, students should be able to test and review current network configuration with basic utilities.

## Displaying IP addresses

The `/sbin/ip` command is used to show device and address information.

```
[student@demo ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,①UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    ②link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    ③inet 172.25.0.11/16 brd ④172.25.255.255 scope global eth0
        valid_lft forever preferred_lft forever
    ⑤inet6 fe80::5054:ff:fe00:b/64 scope link
        valid_lft forever preferred_lft forever
```

- ① An active interface has the status of **UP**.
- ② The link line specifies the hardware (MAC) address of the device.
- ③ The inet line shows the IPv4 address and prefix.
- ④ The broadcast address, scope, and device name are also on this line.
- ⑤ The inet6 line shows IPv6 information.

The `ip` command may also be used to show statistics about network performance. The received (RX) and transmitted (TX) packets, errors, and dropped counters can be used to identify network issues caused by congestion, low memory, and overruns.

```
[student@demo ~]$ ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    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
```

## Troubleshooting routing

The `/sbin/ip` command is also used to show routing information.

```
[student@demo ~]$ ip route
default via 172.25.0.254 dev eth0 proto static metric 1024
172.25.0.0/16 dev eth0 proto kernel scope link src 172.25.0.11
10.0.0.0/8 dev eth1 proto kernel scope link src 10.0.0.11
```

All packets destined for the 10.0.0.0/8 network will be sent directly to the destination through the device eth1. All packets destined for the 172.25.0.0/16 network will be sent directly to the destination through the device eth0. All other packets will be sent to the default router located at 172.25.0.254, and also through device eth0.



The **ping** command is used to test connectivity. The command will continue to run until a **Control-C** is pressed unless options are given to limit the number of packets sent.

```
[student@demo ~]$ ping -c3 172.25.0.254
```

To trace the path to a remote host, use either **traceroute** or **tracepath**. Both commands can be used to trace a path with UDP packets; 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, but may not be installed by default.

```
[student@demo ~]$ tracepath access.redhat.com
...
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.atlmgmq46w-bcr00.tbone.rr.com 107.286ms asymm 12
...
```

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.

## 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. It is similar to the **netstat** command, which is also available but may not be installed by default.

```
[student@demo ~]$ ss -ta
State      Recv-Q  Send-Q      Local Address:Port      Peer Address:Port
LISTEN     0        128          *:sunrpc                 *:
LISTEN     0        128          ①*:ssh                   *:
LISTEN     0        100          ②127.0.0.1:smtp          *:
LISTEN     0        128          *:36889                  *:
ESTAB      0        0          ③172.25.0.11:ssh         172.25.254.254:59392
LISTEN     0        128          :::sunrpc                :::*
LISTEN     0        128          ④:::ssh                  :::*
LISTEN     0        100          ⑤:::1:smtp               :::*
LISTEN     0        128          :::34946                  :::*
```

- ① The port used for SSH is listening on all IPv4 addresses. The "\*" is used to represent "all" when referencing IPv4 addresses or ports.
- ② The port used for SMTP is listening on the 127.0.0.1 IPv4 loopback interface.
- ③ The established SSH connection is on the 172.25.0.11 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.
-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.

**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

Additional information may be available in the chapter on configuring networking in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

|| <http://docs.redhat.com/>

# Practice: Examining Network Configuration

## Guided exercise

In this lab, you will examine the network configuration of the current system.

### Outcomes:

Identify the current network interfaces and basic network addresses.

### Before you begin...

Reset your serverX system.

- ❑ 1. Display the current IP address and netmask for all interfaces.

```
[student@serverX ~]$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    qlen 1000
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    inet 172.25.X.11/24 brd 172.25.X.255 scope global dynamic eth0
        valid_lft 12704sec preferred_lft 12704sec
    inet6 fe80::5054:ff:fe00:b/64 scope link
        valid_lft forever preferred_lft forever
```

- ❑ 2. Display the statistics for the eth0 interface.

```
[student@serverX ~]$ ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    mode DEFAULT qlen 1000
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    418398    4588      0       0        0        0
    TX: bytes  packets  errors  dropped carrier collsns
    360733    1730      0       0        0        0
```

- ❑ 3. Display the routing information.

```
[student@serverX ~]$ ip route
default via 172.25.X.254 dev eth0 proto static metric 1024
172.25.X.0/24 dev eth0 proto kernel scope link src 172.25.X.11
```

- ❑ 4. Verify that the router is accessible.

```
[student@serverX ~]$ ping -c3 172.25.X.254
PING 172.25.X.254 (172.25.X.254) 56(84) bytes of data.
64 bytes from 172.25.X.254: icmp_seq=1 ttl=64 time=0.489 ms
64 bytes from 172.25.X.254: icmp_seq=2 ttl=64 time=0.510 ms
64 bytes from 172.25.X.254: icmp_seq=3 ttl=64 time=0.458 ms
```

```

--- 172.25.X.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.458/0.485/0.510/0.033 ms

```

- ❑ 5. Show all the hops between the local system and classroom.example.com.

```

[student@serverX ~]$ tracepath classroom.example.com
1: classroom.example.com 0.522ms 1H
Resume: pmtu 65535

```

- ❑ 6. Display the listening TCP sockets on the local system.

```

[student@serverX ~]$ ss -lt
State      Recv-Q Send-Q      Local Address:Port      Peer Address:Port
LISTEN     0      128          *:55630                  *:
LISTEN     0      128          *:sunrpc                 *:
LISTEN     0      128          *:ssh                    *:
LISTEN     0      100      127.0.0.1:smtp           *:
LISTEN     0      128          :::sunrpc                :::
LISTEN     0      128          :::ssh                   :::
LISTEN     0      128          :::33079                 :::
LISTEN     0      100          :::1:smtp                 :::

```

# Configuring Networking with **nmcli**

## Objectives

After completing this section, students should be able to manage network settings and devices with **nmcli** and NetworkManager.

## NetworkManager

NetworkManager is a daemon that monitors and manages network settings. In addition to the daemon, there is a GNOME Notification Area applet that provides 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 configuration used for a device which is made up of a collection of settings. Multiple connections may exist for a device, but only one may be active at a time. For example, a system may normally be connected to a network with settings provided by DHCP. Occasionally, that system needs to be connected to a lab or data center network, which only uses static networking. Instead of changing the configuration manually, each configuration can be stored as a separate connection.

## Viewing network information with **nmcli**

To display a list of all connections, use **nmcli con show**. To list only the active connections, add the **--active** option.

```
[root@demo ~]# nmcli con show
```

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	eth0
System eth0	5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03	802-3-ethernet	--
guest	f601ca8a-6647-4188-a431-dab48cc63bf4	802-11-wireless	wlp3s0

```
[root@demo ~]# nmcli con show --active
```

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	eth0
guest	f601ca8a-6647-4188-a431-dab48cc63bf4	802-11-wireless	wlp3s0

Specify a connection ID (name) to see the details of that connection. The lowercase settings represent the configuration of the connection. Setting and property names are defined in the **nm-settings(5)** man page. The uppercase settings are active data.

```
[root@demo ~]# nmcli con show "static-eth0"
```

```
...
ipv4.method:                manual
ipv4.dns:                   172.25.0.254, 8.8.8.8
ipv4.dns-search:
ipv4.addresses:              { ip = 172.25.0.11/16, gw = 172.25.0.254 }
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:                 auto
```

...

The **nmcli** command can also be used to show device status and details.

```
[root@demo ~]# nmcli dev status
DEVICE  TYPE      STATE      CONNECTION
eth0    ethernet  connected  static-eth0
wlp3s0  wifi      connected  guest
lo      loopback  unmanaged  --
[root@demo ~]# nmcli dev show eth0
GENERAL.DEVICE:           eth0
GENERAL.TYPE:             ethernet
GENERAL.HWADDR:           52:54:00:00:00:0B
GENERAL.MTU:              1500
GENERAL.STATE:            100 (connected)
GENERAL.CONNECTION:       static-eth0
GENERAL.CON-PATH:         /org/freedesktop/NetworkManager/
ActiveConnection/1
WIRED-PROPERTIES.CARRIER: on
IP4.ADDRESS[1]:           ip = 172.25.0.11/16, gw = 172.25.0.254
IP4.DNS[1]:               172.25.0.254
IP6.ADDRESS[1]:           ip = fe80::5054:ff:fe00:b/64, gw = ::
```

## Creating network connections with nmcli

When creating a new connection with **nmcli**, the order of the arguments is important. The common arguments appear first and must include the type and interface. Next, specify any type-specific arguments and finally specify the IP address, prefix, and gateway information. Multiple IP addresses may be specified for a single device. Additional settings such as a DNS server are set as modifications once the connection exists.

### Examples of creating new connections

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Define a new connection named "default" which will autoconnect as an Ethernet connection on the eth0 device using DHCP.

```
[root@demo ~]# nmcli con add con-name "default" type ethernet ifname eth0
```

2. Create a new connection named "static" and specify the IP address and gateway. Do not autoconnect.

```
[root@demo ~]# nmcli con add con-name "static" ifname eth0 autoconnect no type
ethernet ip4 192.168.0.5/24 gw4 192.168.0.254
```

3. The system will autoconnect with the DHCP connection at boot. Change to the static connection.

```
[root@demo ~]# nmcli con up "static"
```

4. Change back to the DHCP connection.

```
[root@demo ~]# nmcli con up "default"
```



## Important

If the static connection is lost, the default connection will attempt to autoconnect. To administratively disable an interface and prevent any autoconnection, use **nmcli dev disconnect DEVICENAME**.

### Type options

Type options depend on the type used. An ethernet-type connection may optionally specify a MAC address for the connection. A wifi-type connection must specify the SSID and may specify additional options. Many other types are available, including bridge, bond, team, VPN, and VLAN. To view all the options, use **nmcli con add help**.

```
[root@demo ~]# nmcli con add help
Usage: nmcli connection add { ARGUMENTS | help }

ARGUMENTS := COMMON_OPTIONS TYPE_SPECIFIC_OPTIONS IP_OPTIONS

COMMON_OPTIONS:
    type <type>
    ifname <interface name> | ""
    [con-name <connection name>]
    [autoconnect yes|no]

    [save yes|no]

TYPE_SPECIFIC_OPTIONS:
    ethernet:    [mac <MAC address>]
                 [cloned-mac <cloned MAC address>]
                 [mtu <MTU>]
    ...
```

## Modifying network interfaces with **nmcli**

An existing connection may be modified with **nmcli con mod** arguments. The arguments are sets of key/value pairs. The key includes a setting name and a property name. Use **nmcli con show "<ID>"** to see a list of current values for a connection. The **nm-settings(5)** man page documents the setting and property names and usage.

```
[root@demo ~]# nmcli con show "static"
connection.id:          static
connection.uuid:        f3e8dd32-3c9d-48f6-9066-551e5b6e612d
connection.interface-name: eth0
connection.type:         802-3-ethernet
connection.autoconnect: yes
connection.timestamp:    1394905322
connection.read-only:    no
...
```

### Examples of connection modifications

Follow along with the next steps while your instructor discusses **nmcli** syntax.

1. Turn off autoconnect.

```
[root@demo ~]# nmcli con mod "static" connection.autoconnect no
```

- Specify a DNS server.

```
[root@demo ~]# nmcli con mod "static" ipv4.dns 172.25.0.254
```

- Some configuration arguments may have values added or removed. Add a +/- symbol in front of the argument. Add a secondary DNS server.

```
[root@demo ~]# nmcli con mod "static" +ipv4.dns 8.8.8.8
```

- Replace the static IP address and gateway.

```
[root@demo ~]# nmcli con mod "static" ipv4.address "192.168.0.10/24 192.168.0.254"
```

- Add a secondary IP address without a gateway.

```
[root@demo ~]# nmcli con mod "static" +ipv4.address 10.10.10.10/16
```



### Important

The **nmcli con mod** will save the setting to the configuration files. To activate the changes, the connection needs to be activated or reactivated.

```
[root@demo ~]# nmcli con up "static"
```

## Summary of nmcli commands

Basic device and connection commands for **nmcli**:

### nmcli commands

Command	Use
nmcli dev status	List all devices.
nmcli con show	List all connections.
nmcli con up "<ID>"	Activate a connection.
nmcli con down "<ID>"	Deactivate a connection. The connection will restart if autoconnect is yes.
nmcli dev dis <DEV>	Bring down an interface and temporarily disable autoconnect.
nmcli net off	Disable all managed interfaces.
nmcli con add ...	Add a new connection.
nmcli con mod "<ID>" ...	Modify a connection.
nmcli con del "<ID>"	Delete a connection.





## Note

The **nmcli** command also has an interactive edit mode. For a graphical interface, use **nm-connection-editor**.



## References

**nmcli**(1), **nmcli-examples**(5), and **nm-settings**(5) man pages

Additional information may be available in the section on using the NetworkManager command line tool nmcli in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

|| <http://docs.redhat.com/>

# Practice: Configuring Networking with nmcli

## Guided exercise

In this lab, you will configure network settings using **nmcli**.

### Outcomes:

Convert a system from DHCP to static configuration.

### Before you begin...

Reset your serverX system.

- ❑ 1. View network settings using **nmcli**.

- ❑ 1.1. Show all connections.

```
[student@serverX ~]$ nmcli con show
```

NAME	UUID	TYPE	DEVICE
System eth0	5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03	802-3-ethernet	eth0

- ❑ 1.2. Display all configuration settings for the active connection.

```
[student@serverX ~]$ nmcli con show "System eth0"
```

```
connection.id:                System eth0
connection.uuid:              5fb06bd0-0bb0-7ffb-45f1-
d6edd65f3e03
connection.interface-name:    eth0
connection.type:              802-3-ethernet
connection.autoconnect:       yes
connection.timestamp:         1394813303
connection.read-only:         no
connection.permissions:
...
IP4.ADDRESS[1]:               ip = 172.25.X.11/24, gw =
172.25.X.254
IP4.DNS[1]:                   172.25.254.254
IP4.DOMAIN[1]:                example.com
...
```

- ❑ 1.3. Show device status.

```
[student@serverX ~]$ nmcli dev status
```

DEVICE	TYPE	STATE	CONNECTION
eth0	ethernet	connected	System eth0
lo	loopback	unmanaged	--

- ❑ 1.4. Display the settings for the eth0 device.

```
[student@serverX ~]$ nmcli dev show eth0
```

```
GENERAL.DEVICE:               eth0
GENERAL.TYPE:                 ethernet
GENERAL.HWADDR:               52:54:00:00:00:0B
GENERAL.MTU:                  1500
GENERAL.STATE:                100 (connected)
```

```

GENERAL.CONNECTION:      System eth0
GENERAL.CON-PATH:        /org/freedesktop/NetworkManager/
                           ActiveConnection/1
WIRED-PROPERTIES.CARRIER: on
IP4.ADDRESS[1]:          ip = 172.25.X.11/24, gw =
                           172.25.X.254
IP4.DNS[1]:              172.25.254.254
IP4.DOMAIN[1]:           example.com
IP6.ADDRESS[1]:          ip = fe80::5054:ff:fe00:b/64, gw
                           = ::

```

- ❑ 2. Create a static connection with the same IPv4 address, network prefix, and default gateway. Name the new connection *static-eth0*.

```

[student@serverX ~]$ sudo nmcli con add con-name "static-eth0" ifname eth0 type
ethernet ip4 172.25.X.11/24 gw4 172.25.X.254
Connection 'static-eth0' (f3e8dd32-3c9d-48f6-9066-551e5b6e612d) successfully
added.

```

- ❑ 3. Modify the new connection to add the DNS setting.

```

[student@serverX ~]$ sudo nmcli con mod "static-eth0" ipv4.dns 172.25.254.254

```

- ❑ 4. Display and activate the new connection.

- ❑ 4.1. View all connections.

```

[student@serverX ~]$ nmcli con show

```

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	--
System eth0	5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03	802-3-ethernet	eth0

- ❑ 4.2. View the active connection.

```

[student@serverX ~]$ nmcli con show --active
System eth0 5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03 802-3-ethernet eth0

```

- ❑ 4.3. Activate the new connection.

```

[student@serverX ~]$ sudo nmcli con up "static-eth0"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/3)

```

- ❑ 4.4. View the active connection.

```

[student@serverX ~]$ nmcli con show --active

```

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	eth0

- ❑ 5. Test the connectivity using the new network addresses.

- ❑ 5.1. Verify the IP address.

```
[student@serverX ~]$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff
    inet 172.25.X.11/24 brd 172.25.X.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe00:b/64 scope link
        valid_lft forever preferred_lft forever
```

- 5.2. Verify the default gateway.

```
[student@serverX ~]$ ip route
default via 172.25.X.254 dev eth0 proto static metric 1024
172.25.X.0/24 dev eth0 proto kernel scope link src 172.25.X.11
```

- 5.3. Ping the DNS address.

```
[student@serverX ~]$ ping -c3 172.25.254.254
PING 172.25.254.254 (172.25.254.254) 56(84) bytes of data.
64 bytes from 172.25.254.254: icmp_seq=1 ttl=64 time=0.419 ms
64 bytes from 172.25.254.254: icmp_seq=2 ttl=64 time=0.598 ms
64 bytes from 172.25.254.254: icmp_seq=3 ttl=64 time=0.503 ms

--- 172.25.254.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.419/0.506/0.598/0.077 ms
```

- 6. Configure the original connection so that it does not start at boot and verify that the static connection is used when the system reboots.

- 6.1. Disable the original connection from autostarting at boot.

```
[student@serverX ~]$ sudo nmcli con mod "System eth0" \
> connection.autoconnect no
```

- 6.2. Reboot the system.

```
[student@serverX ~]$ reboot
```

- 6.3. View the active connection.

```
[student@serverX ~]$ nmcli con show --active
```

NAME	UUID	TYPE	DEVICE
static-eth0	f3e8dd32-3c9d-48f6-9066-551e5b6e612d	802-3-ethernet	eth0

# Editing Network Configuration Files

## Objectives

After completing this section, students should be able to modify network settings by editing the configuration files.

## Modifying network configuration

It is also possible to configure the network by editing interface configuration files. Interface 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 configuration.

### Configuration Options for ifcfg File

Static	Dynamic	Either
<b>BOOTPROTO=none</b>	<b>BOOTPROTO=dhcp</b>	<b>DEVICE=eth0</b>
<b>IPADDR0=192.168.0.250</b>		<b>NAME="System eth0"</b>
<b>PREFIX0=24</b>		<b>ONBOOT=yes</b>
<b>GATEWAY0=192.168.0.254</b>		<b>UUID=f3e8dd32-3...</b>
<b>DEFROUTE=yes</b>		<b>USERCTL=yes</b>
<b>DNS1=192.168.0.254</b>		

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 which is 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@demo ~]# nmcli con reload
[root@demo ~]# nmcli con down "System eth0"
[root@demo ~]# nmcli con up "System eth0"
```



## References

**nmcli(1)** man page

Additional information may be available in the chapter on configuring networking in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

# Practice: Editing Network Configuration Files

## Guided exercise

In this lab, you will edit network configuration files.

### Outcomes:

An additional network address added to each system.

### Before you begin...

Reset your serverX and desktopX systems.

- ❑ 1. As the root user, edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` on serverX to add an additional address of `10.0.X.1/24`.

- ❑ 1.1. Append an entry to the file to specify the IPv4 address.

```
[root@serverX ~]# echo "IPADDR1=10.0.X.1" >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

- ❑ 1.2. Append an entry to the file to specify the network prefix.

```
[root@serverX ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

- ❑ 2. Activate the new address.

- ❑ 2.1. Reload the configuration changes.

```
[root@serverX ~]# nmcli con reload
```

- ❑ 2.2. Restart the connection with the new settings.

```
[root@serverX ~]# nmcli con up "System eth0"
```

- ❑ 3. As the root user, edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` on desktopX to add an additional address of `10.0.X.2/24` and load the new configuration.

- ❑ 3.1. Modify the file to add the IPv4 and network prefix.

```
[root@desktopX ~]# echo "IPADDR1=10.0.X.2" >> /etc/sysconfig/network-scripts/ifcfg-eth0
[root@desktopX ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-eth0
```

- ❑ 3.2. Reload the configuration changes.

```
[root@desktopX ~]# nmcli con reload
```

- ☐ 3.3. Bring up the connection with the new settings.

```
[root@desktopX ~]# nmcli con up "System eth0"
```

- ☐ 4. Test the connectivity using the new network addresses.

- ☐ 4.1. On serverX, verify the IP address.

```
[root@serverX ~]# ip addr
```

- ☐ 4.2. On serverX, ping the new address of desktopX.

```
[root@serverX ~]# ping 10.0.X.2
```

- ☐ 4.3. On desktopX, verify the IP address.

```
[root@desktopX ~]# ip addr
```

- ☐ 4.4. On desktopX, ping the new address of serverX.

```
[root@desktopX ~]# ping 10.0.X.1
```

# Configuring Host Names and Name Resolution

## Objectives

After completing this section, students should be able to configure and test system host name and name resolution.

## Changing the system host name

The **hostname** command displays or temporarily modifies the system's fully qualified host name.

```
[root@demo ~]# hostname
demo.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@demo ~]# hostnamectl set-hostname demo.example.com
[root@demo ~]# hostnamectl status
  Static hostname: desktop0.example.com
        Icon name: computer
        Chassis: n/a
  Machine ID: 9f6fb63045a845d79e5e870b914c61c9
  Boot ID: aa6c3259825e4b8c92bd0f601089ddf7
  Virtualization: kvm
  Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)
  CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:beta:server
  Kernel: Linux 3.10.0-97.el7.x86_64
  Architecture: x86_64
[root@demo ~]# cat /etc/hostname
demo.example.com
```



### Important

The static host name is stored in **/etc/hostname**. Previous versions of Red Hat Enterprise Linux stored 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. The contents of the file **/etc/hosts** are checked first.

```
[root@demo ~]# cat /etc/hosts
192.168.0.250    demo.example.com    demo    # Added by NetworkManager
127.0.0.1       localhost.localdomain localhost
::1            demo.example.com    demo    localhost6.localdomain6 localhost6
```

The **getent hosts hostname** command can be used to test host name resolution with the **/etc/hosts** file.



If an entry is not found in that file, the stub resolver looks for the information from a DNS nameserver. The **/etc/resolv.conf** file controls how this query is done:

- **nameserver**: the IP address of a nameserver to query. Up to three nameserver directives may be given to provide backups if one is down.
- **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.

```
[root@demo ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.0.254
```

NetworkManager will update the **/etc/resolv.conf** file using DNS settings in the connection configuration files. Use the **nmcli** to modify the connections.

```
[root@demo ~]# nmcli con mod ID ipv4.dns IP
[root@demo ~]# nmcli con down ID
[root@demo ~]# nmcli con up ID
[root@demo ~]# cat /etc/sysconfig/network-scripts/ifcfg-ID
...
DNS1=8.8.8.8
...
```

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 +/- symbol in front of the **ipv4.dns** argument will add or remove an individual entry.

```
[root@demo ~]# nmcli con mod ID +ipv4.dns IP
```

The **host HOSTNAME** command can be used to test DNS server connectivity.

```
[root@demo ~]# host classroom.example.com
classroom.example.com has address 172.25.0.254
[root@demo ~]# host 172.25.0.254
254.0.25.172.in-addr.arpa domain name pointer classroom.example.com.
```



## Important

If DHCP is in use, **/etc/resolv.conf** is automatically rewritten as interfaces are started, unless you specify **PEERDNS=no** in the relevant interface configuration files. The change can be made with **nmcli**.

```
[root@demo ~]# nmcli con mod "System eth0" ipv4.ignore-auto-dns yes
```



## References

**nmccli(1)**, **hostnamectl(1)**, **hosts(5)**, **getent(1)**, **host(1)**, and **resolv.conf(5)** man pages

Additional information may be available in the chapter on configuring host names in the *Red Hat Enterprise Linux Networking Guide* for Red Hat Enterprise Linux 7, which can be found at

|| <http://docs.redhat.com/>

# Practice: Configuring Host Names and Name Resolution

## Guided exercise

In this lab, you will configure the system host name and name resolution.

### Outcomes:

Customized host name and name resolution settings.

### *Before you begin...*

Reset your serverX system.

- ❑ 1. View the current host name settings.

- ❑ 1.1. Display the current host name.

```
[student@serverX ~]$ hostname
serverX.example.com
```

- ❑ 1.2. Display the host name status.

```
[student@serverX ~]$ hostnamectl status
Static hostname: n/a
Transient hostname: serverX.example.com
Icon name: computer
Chassis: n/a
Machine ID: 9f6fb63045a845d79e5e870b914c61c9
Boot ID: d4ec3a2e8d3c48749aa82738c0ea946a
Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)
CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:beta:server
Kernel: Linux 3.10.0-97.el7.x86_64
Architecture: x86_64
```

- ❑ 2. Set a static host name to match the current transient host name.

- ❑ 2.1. Change the host name and host name configuration file. Replace the X with your station number and match the output of the previous step.

```
[student@serverX ~]$ sudo hostnamectl set-hostname serverX.example.com
```

- ❑ 2.2. View the configuration file providing the host name at network start.

```
[student@serverX ~]$ cat /etc/hostname
serverX.example.com
```

- ❑ 2.3. Display the host name status.

```
[student@serverX ~]$ hostnamectl status
Static hostname: serverX.example.com
Icon name: computer
```

```

Chassis: n/a
Machine ID: 9f6fb63045a845d79e5e870b914c61c9
Boot ID: d4ec3a2e8d3c48749aa82738c0ea946a
Operating System: Red Hat Enterprise Linux Server 7.0 (Maipo)
CPE OS Name: cpe:/o:redhat:enterprise_linux:7.0:beta:server
Kernel: Linux 3.10.0-97.el7.x86_64
Architecture: x86_64

```

- 3. Temporarily change the host name.

- 3.1. Change the host name.

```
[student@serverX ~]$ sudo hostname testname
```

- 3.2. Display the current host name.

```
[student@serverX ~]$ hostname
testname
```

- 3.3. View the configuration file providing the host name at network start.

```
[student@serverX ~]$ cat /etc/hostname
serverX.example.com
```

- 3.4. Reboot the system.

```
[student@serverX ~]$ reboot
```

- 3.5. Display the current host name.

```
[student@serverX ~]$ hostname
serverX.example.com
```

- 4. Add a local nickname for the classroom server.

- 4.1. Look up the IP address of the classroom.example.com.

```
[student@serverX ~]$ host classroom.example.com
classroom.example.com has address 172.25.254.254
```

- 4.2. Modify **/etc/hosts** so that the name **class** has the IP address 172.25.254.254 and can be used to communicate with classroom.example.com.

```

[student@serverX ~]$ sudo vi /etc/hosts
[student@serverX ~]$ 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 class

```

```
172.25.254.254 content.example.com
```

- 4.3. Look up the IP address of the class.

```
[student@serverX ~]$ host class
Host class not found: 2(SERVFAIL)
[student@serverX ~]$ getent hosts class
172.25.254.254 classroom.example.com class
```

- 4.4. Ping class.

```
[student@serverX ~]$ 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
```

# Lab: Managing Red Hat Enterprise Linux Networking

## Performance checklist

In this lab, you will configure basic IPv4 networking on Red Hat Enterprise Linux systems.

### Outcomes:

The primary interface has two static IPv4 addresses configured.

### *Before you begin...*

Reset your desktopX system.

1. Create a new connection with a static network connection using the settings in the table. Be sure to replace the X with the correct number for your systems.

Parameter	Setting
Connection name	lab
IP address	172.25.X.10/24
Gateway address	172.25.X.254
DNS address	172.25.254.254

2. Configure the new connection to be autostarted. Other connections should not start automatically.
3. Modify the new connection so that it also uses the address 10.0.X.1/24.
4. Configure the **hosts** file so that 10.0.X.1 can be referenced as "private".
5. Reboot the system, then run **lab network grade** to verify settings.

## Solution

In this lab, you will configure basic IPv4 networking on Red Hat Enterprise Linux systems.

### Outcomes:

The primary interface has two static IPv4 addresses configured.

### Before you begin...

Reset your desktopX system.

1. Create a new connection with a static network connection using the settings in the table. Be sure to replace the X with the correct number for your systems.

Parameter	Setting
Connection name	lab
IP address	172.25.X.10/24
Gateway address	172.25.X.254
DNS address	172.25.254.254

```
[root@desktopX ~]# nmcli con add con-name lab ifname eth0 type ethernet ip4
172.25.X.10/24 gw4 172.25.X.254
[root@desktopX ~]# nmcli con mod "lab" ipv4.dns 172.25.254.254
```

2. Configure the new connection to be autostarted. Other connections should not start automatically.

```
[root@desktopX ~]# nmcli con mod "lab" connection.autoconnect yes
[root@desktopX ~]# nmcli con mod "System eth0" connection.autoconnect no
```

3. Modify the new connection so that it also uses the address 10.0.X.1/24.

```
[root@desktopX ~]# nmcli con mod "lab" +ipv4.addresses 10.0.X.1/24
```

Or alternately:

```
[root@desktopX ~]# echo "IPADDR1=10.0.X.1" >> /etc/sysconfig/network-scripts/ifcfg-lab
[root@desktopX ~]# echo "PREFIX1=24" >> /etc/sysconfig/network-scripts/ifcfg-lab
```

4. Configure the **hosts** file so that 10.0.X.1 can be referenced as "private".

```
[root@desktopX ~]# echo "10.0.X.1 private" >> /etc/hosts
```

5. Reboot the system, then run **lab network grade** to verify settings.

```
[root@desktopX ~]# lab network grade
```

## Summary

### Validating Network Configuration

Use basic utilities to determine current network configuration.

### Configuring Networking with **nmcli**

Manage network devices with command-line utilities.

### Editing Network Configuration Files

Modify network configuration files.

### Configuring Host Names and Name Resolution

Display and change system host name and name resolution configuration.





## CHAPTER 11

# SYSTEM LOGGING AND NTP

Overview	
<b>Goal</b>	To locate and accurately interpret relevant system log files for troubleshooting purposes.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Describe the basic syslog architecture in Red Hat Enterprise Linux 7.</li><li>• Interpret entries in relevant syslog files to troubleshoot problems or review system status.</li><li>• Find and interpret log entries in the systemd journal to troubleshoot problems or review system status.</li><li>• Configure systemd-journald to store its journal on disk rather than in memory.</li><li>• Maintain accurate time synchronization and time zone configuration to ensure correct timestamps in system logs.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• System Log Architecture (and Practice)</li><li>• Reviewing Syslog Files (and Practice)</li><li>• Reviewing systemd Journal Entries (and Practice)</li><li>• Preserving the systemd Journal (and Practice)</li><li>• Maintaining Accurate Time (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Analyzing and Storing Logs</li></ul>

# System Log Architecture

## Objectives

After completing this section, students should be able to describe the basic syslog architecture in Red Hat Enterprise Linux 7.

## System logging

Processes and the operating system kernel need to be able to record a log of events that happen. These logs can be useful for auditing the system and troubleshooting problems. By convention, the **/var/log** directory is where these logs are persistently stored.

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. In Red Hat Enterprise Linux 7, syslog messages are handled by two services, **systemd-journald** and **rsyslogd**.

The **systemd-journald** daemon provides an improved log management service that collects messages from the kernel, the early stages of the boot process, standard output and error of daemons as they start up and run, and syslog. It writes these messages to a structured journal of events that, by default, does not persist between reboots. This allows syslog messages and events which are missed by syslog to be collected in one central database. The syslog messages are also forwarded by **systemd-journald** to **rsyslogd** for further processing.

The **rsyslogd** service then sorts the syslog messages by type (or facility) and priority, and writes them to persistent files in the **/var/log** directory.

The **/var/log** directory holds various system- and service-specific log files maintained by **rsyslog**:

### Overview of system log files

Log file	Purpose
<b>/var/log/messages</b>	Most syslog messages are logged here. The exceptions are messages related to authentication and email processing, that periodically run jobs, and those which are purely debugging-related.
<b>/var/log/secure</b>	The log file for security and authentication-related messages and errors.
<b>/var/log/maillog</b>	The log file with mail server-related messages.
<b>/var/log/cron</b>	The log file related to periodically executed tasks.
<b>/var/log/boot.log</b>	Messages related to system startup are logged here.



## References

**systemd-journald.service(8)**, **rsyslogd(8)**, and **rsyslog.conf(5)** man pages

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

## Practice: System Logging Components

### Quiz

Match the following items to their counterparts in the table.

/var/log	/var/log/boot.log	/var/log/cron	/var/log/maillog
/var/log/messages	/var/log/secure		

Purpose	Log file
Most syslog messages are logged here. The exceptions are messages related to authentication, email processing, and that periodically run jobs, or those which are purely debugging-related.	
The log file for security and authentication-related messages and errors.	
The directory to which <b>rsyslog</b> is writing all the log files.	
The log file with mail server-related messages.	
The log file related to periodically executed tasks.	
Messages related to system startup are logged here.	

## Solution

Match the following items to their counterparts in the table.

Purpose	Log file
Most syslog messages are logged here. The exceptions are messages related to authentication, email processing, and that periodically run jobs, or those which are purely debugging-related.	/var/log/messages
The log file for security and authentication-related messages and errors.	/var/log/secure
The directory to which <b>rsyslog</b> is writing all the log files.	/var/log
The log file with mail server-related messages.	/var/log/maillog
The log file related to periodically executed tasks.	/var/log/cron
Messages related to system startup are logged here.	/var/log/boot.log

## Reviewing Syslog Files

### Objectives

After completing this section, students should be able to interpret entries in relevant syslog files to troubleshoot problems or review system status.

### Syslog files

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). The facilities which are available are documented by the **rsyslog.conf(5)** man page.

The eight priorities are also standardized and ranked as follows:

#### 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 **rsyslogd** service uses the facility and priority of log messages to determine how to handle them. This is configured by the file **/etc/rsyslog.conf** and by **\*.conf** files in **/etc/rsyslog.d**. Programs and administrators can change **rsyslogd** configuration in a way that will not be overwritten by updates to **rsyslog** by putting customized files with a **.conf** suffix in the **/etc/rsyslog.d** directory.

The **#### RULES ####** section of **/etc/rsyslog.conf** contains directives that define where log messages are saved. The left side of each line indicates the facility and severity of the log message the directive matches. The **rsyslog.conf** file can contain the character **\*** as a wild card in the facility and severity field, where it either stands for all facilities or all severities. The right side of each line indicates what file to save the log message in. Log messages are normally saved in files in the **/var/log** directory.



#### Note

Log files are maintained by the **rsyslog** service, and the **/var/log** directory contains a variety of log files specific to certain services. For example, the Apache Web Server or Samba write their own log files into a corresponding subdirectory of the **/var/log** directory.

A message handled by **rsyslog** can appear in multiple different log files. To prevent that, the severity field can be set to **none**, which means that none of the messages directed to this facility are added to the specified log file.

Instead of logging syslog messages to a file, they can be printed to the terminals of all logged-in users. In the default **rsyslog.conf** file, this is done for all messages that have "emerg" priority.

## Sample rules section of **rsyslog.conf**

```
##### 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 **rsyslog.conf** file is documented by the **rsyslog.conf(5)** man page and by extensive HTML documentation in **/usr/share/doc/rsyslog-\*/manual.html** contained in the *rsyslog-doc*, which is available from the Red Hat Enterprise Linux 7 software channel, but not included on the installation medium.

## Log file rotation

Logs are "rotated" by the **logrotate** utility to keep them from filling up the file system containing **/var/log/**. When a log file is rotated, it is renamed with an extension indicating the date on which it was rotated: the old **/var/log/messages** file may become **/var/log/messages-20141030** if it is rotated on October 30, 2014. 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 old log file is discarded to free disk space. A cron 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.

## Analyze a syslog entry

The system logs written by **rsyslog** start with the oldest message on top and the newest message at the end of the log file. All log entries in log files managed by **rsyslog** are recorded in a standard format. The following example will explain the anatomy of a log file message in the **/var/log/secure** log file:

```
❶ Feb 11 20:11:48 ❷ localhost ❸ sshd[1433]: ❹ Failed 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 that sent the log message.
- ❹ The actual message sent.

## Monitor a log file with tail

It is especially helpful for reproducing problems and issues to monitor one or more log files for events. The **tail -f /path/to/file** command outputs the last 10 lines of the file specified and continues to output new lines as they get written to the monitored file.

To monitor for failed login attempts on one terminal, run **ssh** as user root while a user tries to log in to the serverX machine:

```
[root@serverX ~]$ tail -f /var/log/secure
...
Feb 10 09:01:13 localhost sshd[2712]: Accepted password for root from 172.25.254.254
port 56801 ssh2
Feb 10 09:01:13 localhost sshd[2712]: pam_unix(sshd:session): session opened for user
root by (uid=0)
```

## Send a syslog message with logger

The **logger** command can send messages to the **rsyslog** service. By default, it sends the message to the facility user with severity notice (**user.notice**) unless specified otherwise with the **-p** option. It is especially useful to test changes to the **rsyslog** configuration.

To send a message to **rsyslogd** that gets recorded in the **/var/log/boot.log** log file, execute:

```
[root@serverX ~]$ logger -p local7.notice "Log entry created on serverX"
```





## References

**logger(1)**, **tail(1)**, **rsyslog.conf(5)**, and **logrotate(8)** man pages

### **rsyslog** Manual

- **/usr/share/doc/rsyslog-\*/manual.html** provided by the *rsyslog-doc* package

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

## Practice: Finding Log Entries

### Guided exercise

In this lab, you will reconfigure **rsyslog** to write specific messages to a new log file.

#### Outcomes:

The **rsyslog** service writes all messages with priority debug to the **/var/log/messages-debug** log file for temporary troubleshooting purposes.

- ❑ 1. Configure **rsyslog** on serverX to log all messages with severity debug in the newly created log file **/var/log/messages-debug** by adding the **rsyslog** configuration file **/etc/rsyslog.d/debug.conf**. Verify that a generated debug log message with the **logger** command arrives in the **/var/log/messages-debug** log file.
  - ❑ 1.1. Change the **rsyslog** configuration to log all messages with severity debug to **/var/log/messages-debug** on serverX by adding the **/etc/rsyslog.d/debug.conf** file.

```
[root@serverX ~]# echo "*.debug /var/log/messages-debug" >/etc/rsyslog.d/debug.conf
```

- ❑ 1.2. Restart the rsyslog service on serverX.

```
[root@serverX ~]# systemctl restart rsyslog
```

- ❑ 2. Generate a debug log message with the **logger** command and verify that the message gets logged to the log file **/var/log/messages-debug** with the **tail** command on serverX.
  - ❑ 2.1. Monitor the **/var/log/messages-debug** with the **tail** command on serverX.

```
[root@serverX ~]# tail -f /var/log/messages-debug
```

- ❑ 2.2. On a separate terminal window, use the **logger** command to generate a debug message on serverX.

```
[root@serverX ~]# logger -p user.debug "Debug Message Test"
```

- ❑ 2.3. Switch back to the terminal still running the **tail -f /var/log/messages-debug** command and verify the message sent with the **logger** command shows up.

```
[root@serverX ~]# tail -f /var/log/messages-debug
...
Feb 13 10:37:44 localhost root: Debug Message Test
```

# Reviewing systemd Journal Entries

## Objectives

After completing this section, students should be able to find and interpret log entries in the systemd journal to troubleshoot problems or review system status.

## Finding events with **journalctl**

The systemd journal stores logging data in a structured, indexed binary file. This data includes extra information about the log event. For syslog events, this can include the facility and priority of the original message, for example.



### Important

In Red Hat Enterprise Linux 7, the systemd journal is stored in **/run/log** by default, and its contents are cleared after a reboot. This setting can be changed by the system administrator and is discussed elsewhere in this course.

The **journalctl** command shows the full system journal, starting with the oldest log entry, when run as root user:

```
[root@serverX ~]# journalctl
Feb 13 10:01:01 server1 run-parts(/etc/cron.hourly)[8678]: starting 0yum-hourly.cron
Feb 13 10:01:01 server1 run-parts(/etc/cron.hourly)[8682]: finished 0yum-hourly.cron
Feb 13 10:10:01 server1 systemd[1]: Starting Session 725 of user root.
Feb 13 10:10:01 server1 systemd[1]: Started Session 725 of user root.
Feb 13 10:10:01 server1 CROND[8687]: (root) CMD (/usr/lib64/sa/sa1 1 1)
```

The **journalctl** command highlights in bold text messages of priority notice or warning, and messages of priority error and higher are highlighted in red.

The key to successfully using the journal for troubleshooting and auditing is to limit the journal searches to only show relevant output. In the following paragraphs, various different strategies to reduce the output of journal queries will be introduced.

By default, **journalctl -n** shows the last 10 log entries. It takes an optional parameter for how many of the last log entries should be displayed. To display the last 5 log entries, run:

```
[root@serverX ~]# journalctl -n 5
```

When troubleshooting problems, it is useful to filter the output of the journal by priority of the journal entries. The **journalctl -p** takes either the name or the number of the known priority levels and shows the given levels and all higher-level entries. The priority levels known to **journalctl** are debug, info, notice, warning, err, crit, alert, and emerg.

To filter the output of the **journalctl** command to only list any log entry of priority err or above, run:

```
[root@serverX ~]# journalctl -p err
```

Similar to the **tail -f** command, **journalctl -f** outputs the last 10 lines of the journal and continues to output new journal entries as they get written to the journal.

```
[root@serverX ~]# journalctl -f
```

When looking for specific events, it is useful to 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 parameter in the format **YYYY-MM-DD hh:mm:ss**. If the date is omitted, the command assumes the date is today, and if the time part is omitted, the whole day starting at 00:00:00 is assumed. Both options take **yesterday**, **today**, and **tomorrow** as valid parameters in addition to the date and time field.

Output all journal entries that got recorded today:

```
[root@serverX ~]# journalctl --since today
```

Output the journal entries from 10th February 2014 20:30:00 to 13th February 2014 12:00:00:

```
[root@serverX ~]# journalctl --since "2014-02-10 20:30:00" --until "2014-02-13 12:00:00"
```

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. All of the displayed extra fields 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@serverX ~]# journalctl -o verbose
Thu 2014-02-13 02:06:00.409345 EST [s=0b47ab9f995149c191a8e539e18c3f9c;
i=d28;b=1ea26e84667848af9a4a2904a76ff9a5;m=4d6878ff5a;t=4f244525daa67;
x=880bc65783036719]
  _SOURCE_REALTIME_TIMESTAMP=1392275160409345
  _PID=6833
  _COMM=sshd
  _EXE=/usr/sbin/sshd
  _SYSTEMD_CGROUP=/system.slice/sshd.service
  _SYSTEMD_UNIT=sshd.service
  _SELINUX_CONTEXT=system_u:system_r:sshd_t:s0-s0:c0.c1023
  _HOSTNAME=serverX
  _CMDLINE=sshd: root [priv]
  _SYSLOG_IDENTIFIER=sshd
  _SYSLOG_FACILITY=10
  _TRANSPORT=syslog
  _CAP_EFFECTIVE=0000001fffffffff
  _MACHINE_ID=4513ad59a3b442ffa4b7ea88343fa55f
  _BOOT_ID=1ea26e84667848af9a4a2904a76ff9a5
  _GID=0
  _UID=0
  _PRIORITY=6
```

Among the more useful options to search for lines relevant to a particular process or event are:

- **\_COMM** The name of the command
- **\_EXE** The path to the executable for the process

- **\_PID** The PID of the process
- **\_UID** The UID of the user running the process
- **\_SYSTEMD\_UNIT** The systemd unit that started the process

More than one of these can be combined. For example, the following query shows all journal entries related to processes started by the systemd unit file `sshd.service`, which also have PID 1182:

```
[root@serverX ~]# journalctl _SYSTEMD_UNIT=sshd.service _PID=1182
```



## Note

For a list of commonly used journal fields, consult the `systemd.journal-fields(7)` man page.



## References

**journalctl(1)** and `systemd.journal-fields(7)` man pages

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

## Practice: Finding Events With journalctl

### Guided exercise

In this lab, you will filter the **systemd** journal for specific criteria.

#### Outcomes:

Students will practice displaying the **systemd** journal output matching different criteria.

- ❑ 1. Output only **systemd** journal messages that originate from the **systemd** process that always runs with process id 1 on serverX.

```
[root@serverX ~]# journalctl _PID=1
```

- ❑ 2. Display all **systemd** journal messages that originate from a system service started with user id 81 on serverX.

```
[root@serverX ~]# journalctl _UID=81
```

- ❑ 3. Output the journal messages with priority **warning** and above on serverX.

```
[root@serverX ~]# journalctl -p warning
```

- ❑ 4. Create a **journalctl** query to show all log events recorded in the previous 10 minutes on serverX. The command assumes a current time of 9:15:00.

```
[root@serverX ~]# journalctl --since 9:05:00 --until 9:15:00
```

- ❑ 5. Display only the events originating from the **sshd** service with the system unit file **sshd.service** recorded since 9:00:00 this morning on serverX.

```
[root@serverX ~]# journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"
```

# Preserving the systemd Journal

## Objectives

After completing this section, students should be able to configure **systemd-journald** to store its journal on disk rather than in memory.

## Store the system journal permanently

By default, the systemd journal is kept in **/run/log/journal**, which means it is cleared when the system reboots. The journal is a new mechanism in Red Hat Enterprise Linux 7, and for most installations, a detailed journal that starts with the last boot is sufficient.

If the directory **/var/log/journal** exists, the journal will log to that directory instead. The advantage of this is the historic data will be available immediately at boot. However, even with a persistent journal, not all data will be kept forever. The journal has a built-in log rotation mechanism that will trigger monthly. In addition, by default, the journal will not be 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 in **/etc/systemd/journald.conf**, and the current limits on the size of the journal are logged when the **systemd-journald** process starts, as can be seen by the following command, which shows the top two lines of **journalctl** output:

```
[root@serverX ~]# journalctl | head -2
-- Logs begin at Wed 2014-03-05 15:13:37 CST, end at Thu 2014-03-06 21:57:54 CST. --
Mar 05 15:13:37 serverX.example.com systemd-journal[94]: Runtime journal is using 8.0M
(max 277.8M, leaving 416.7M of free 2.7G, current limit 277.8M).
```

The systemd journal can be made persistent by creating the directory **/var/log/journal** as user root:

```
[root@serverX ~]# mkdir -p /var/log/journal
```

Ensure that the **/var/log/journal** directory is owned by the root user and group systemd-journal, and has the permissions 2755.

```
[root@serverX ~]# chown root:systemd-journal /var/log/journal
[root@serverX ~]# chmod 2755 /var/log/journal
```

Either a reboot of the system or sending the special signal **USR1** as user root to the **systemd-journald** process is required.

```
[root@serverX ~]# killall -USR1 systemd-journald
```

Since the systemd journal is now persistent across reboots, **journalctl -b** can reduce the output by only showing the log messages since the last boot of the system.

```
[root@serverX ~]# 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 to how many prior system boots the output should be limited. For example, the **journalctl -b -1** limits the output to the previous boot.



## References

**mkdir(1)**, **systemd-journald(1)**, and **killall(1)** man pages

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at <http://docs.redhat.com/>



# Practice: Configure a Persistent systemd Journal

## Guided exercise

In this lab, students will make the systemd journal persistent.

### Outcomes:

The **systemd** journal is written to disk.

- ❑ 1. Configure the systemd journal to be persistent across reboots.

- ❑ 1.1. Configure the directory **/var/log/journal** on serverX.

```
[root@serverX ~]# mkdir -p /var/log/journal
[root@serverX ~]# chown root:systemd-journal /var/log/journal
[root@serverX ~]# chmod 2755 /var/log/journal
```

- ❑ 1.2. Send the **USR1** signal to the **systemd-journald** or reboot serverX.

```
[root@serverX ~]# killall -USR1 systemd-journald
```

- ❑ 2. To verify the systemd journal is persistent, look for a new directory with the systemd journal log files that have been written to **/var/log/journal**. (The exact files which appear may vary on your system, but the directory should have similar contents to the following example.)

```
[root@serverX ~]# ls /var/log/journal/4513ad59a3b442ffa4b7ea88343fa55f
system.journal      user-1000.journal
```

# Maintaining Accurate Time

## Objectives

After completing this section, students should be able to maintain accurate time synchronization and time zone configuration to ensure correct timestamps in system logs.

## Set local clocks and time zone

Correct synchronized system time is very important 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.

```
[student@serverX ~]$ timedatectl
    Local time: Thu 2014-02-13 02:16:15 EST
    Universal time: Thu 2014-02-13 07:16:15 UTC
    RTC time: Thu 2014-02-13 07:16:15
    Timezone: America/New_York (EST, -0500)
    NTP enabled: yes
    NTP synchronized: no
    RTC in local TZ: no
    DST active: no
    Last DST change: DST ended at
                     Sun 2013-11-03 01:59:59 EDT
                     Sun 2013-11-03 01:00:00 EST
    Next DST change: DST begins (the clock jumps one hour forward) at
                     Sun 2014-03-09 01:59:59 EST
                     Sun 2014-03-09 03:00:00 EDT
```

A database with known time zones is available and can be listed with:

```
[student@serverX ~]$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
...
```

Time zone names are based on the public "tz" (or "zoneinfo") time zone database maintained by IANA. 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 changes to the time zone setting of the system.

The system setting for the current time zone can be adjusted as user root:

```
[root@serverX ~]# timedatectl set-timezone America/Phoenix
[root@serverX ~]# timedatectl
    Local time: Thu 2014-02-13 00:23:54 MST
    Universal time: Thu 2014-02-13 07:23:54 UTC
    RTC time: Thu 2014-02-13 07:23:53
    Timezone: America/Phoenix (MST, -0700)
    NTP enabled: yes
    NTP synchronized: no
    RTC in local TZ: no
    DST active: n/a
```

To change the current time and date settings with the **timedatectl** command, the **set-time** option is available. The time is specified in the "YYYY-MM-DD hh:mm:ss" format, where either date or time can be omitted. To change the time to 09:00:00, run:

```
[root@serverX ~]$ timedatectl set-time 9:00:00
[root@serverX ~]$ timedatectl
    Local time: Thu 2014-02-13 09:00:27 MST
    Universal time: Thu 2014-02-13 16:00:27 UTC
    RTC time: Thu 2014-02-13 16:00:28
    Timezone: America/Phoenix (MST, -0700)
    NTP enabled: yes
    NTP synchronized: no
    RTC in local TZ: no
    DST active: n/a
```

The **set-ntp** option enables or disables NTP synchronization for automatic time adjustment. The option requires either a **true** or **false** argument to turn it on or off. To turn on NTP synchronization, run:

```
[student@desktopX ~]$ timedatectl set-ntp true
```

## 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, or if no network connectivity is available, to the calculated RTC clock drift which is recorded in the **driftfile** specified in the **/etc/chrony.conf** configuration file.

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

The quality of an NTP time source is determined by the **stratum** value reported by the time source. 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.

There are two categories of time sources that can be configured in the `/etc/chrony.conf` configuration file, **server** and **peer**. 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.

To reconfigure the **chronyd** server to synchronize with `classroom.example.com` instead of the default servers configured in the `/etc/chrony.conf`, remove the other server entries and replace them with the following configuration file entry:

```
# Use public servers from the pool.ntp.org project.
server classroom.example.com iburst
```

After pointing **chronyd** to the local time source, `classroom.example.com`, the service needs to be restarted:

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

The **chronyc** command acts as a client to the **chronyd** service. After setting up NTP synchronization, it is useful to verify the NTP server was used to synchronize the system clock. This can be achieved with the **chronyc sources** command or, for more verbose output with additional explanations about the output, **chronyc sources -v**:

```
[root@serverX ~]$ 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,
||               |  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.



## Note

Red Hat Enterprise Linux 6 and earlier use **ntpd** and **ntpq** to manage the NTP configuration. Further information may be found in the documentation for Red Hat Enterprise Linux 6.



## References

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

Additional information may be available in the *Red Hat Enterprise Linux System Administrator's Guide* for Red Hat Enterprise Linux 7, which can be found at

<http://docs.redhat.com/>

NTP Pool Project

<http://www.pool.ntp.org/>

Time Zone Database

<http://www.iana.org/time-zones>

## Practice: Adjusting System Time

### Quiz

The steps to set up the correct time zone and adjust system clocks by using **timedatectl** and **chronyd** follow. Indicate the order in which the steps should be taken.

- ☐ a. Identify the appropriate time zone with the **timedatectl list-timezones** command.
- ☐ b. Verify the clock was synchronized against the newly specified NTP source with the **chronyc sources** command.
- ☐ c. Point **chronyd** to a new time source by adjusting **/etc/chrony.conf**.
- ☐ d. Turn on NTP synchronization.
- ☐ e. Review current settings with **timedatectl**.
- ☐ f. Set the correct time zone with **timedatectl set-timezone**.
- ☐ g. Restart the **chronyd** service.

## Solution

The steps to set up the correct time zone and adjust system clocks by using **timedatectl** and **chronyd** follow. Indicate the order in which the steps should be taken.

- 2 a. Identify the appropriate time zone with the **timedatectl list-timezones** command.
- 7 b. Verify the clock was synchronized against the newly specified NTP source with the **chronyc sources** command.
- 5 c. Point **chronyd** to a new time source by adjusting **/etc/chrony.conf**.
- 4 d. Turn on NTP synchronization.
- 1 e. Review current settings with **timedatectl**.
- 3 f. Set the correct time zone with **timedatectl set-timezone**.
- 6 g. Restart the **chronyd** service.

## Lab: Analyzing and Storing Logs

### Performance checklist

In this lab, students will change the time zone and log all authentication failure log entries into a separate file.

#### Outcomes:

The time zone setting on the serverX machine is adjusted; all **systemd** journal entries recorded in a given time frame are displayed; all syslog messages with the authpriv facility and severity alert are logged into a separate log file.

#### *Before you begin...*

Reset your serverX system.

1. Your serverX machine has been relocated to Jamaica. Change the time zone on the serverX machine to Jamaica and verify the time zone has been changed properly.
2. Display all **systemd** journal entries recorded in the last 30 minutes on serverX.
3. Configure **rsyslogd** by adding a rule to the newly created configuration file **/etc/rsyslog.d/auth-errors.conf** to log all security and authentication messages with the priority alert and higher to the **/var/log/auth-errors** file as well. Test the newly added log directive with the **logger** command.



## Solution

In this lab, students will change the time zone and log all authentication failure log entries into a separate file.

### Outcomes:

The time zone setting on the serverX machine is adjusted; all **systemd** journal entries recorded in a given time frame are displayed; all syslog messages with the authpriv facility and severity alert are logged into a separate log file.

### Before you begin...

Reset your serverX system.

1. Your serverX machine has been relocated to Jamaica. Change the time zone on the serverX machine to Jamaica and verify the time zone has been changed properly.
  - 1.1. Identify the correct time zone for Jamaica on serverX.

```
[root@serverX ~]# timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
...
America/Jamaica
...
```

- 1.2. Change the time zone to Jamaica on serverX.

```
[root@serverX ~]# timedatectl set-timezone America/Jamaica
```

- 1.3. Verify the time zone has been properly set on serverX.

```
[root@serverX ~]# timedatectl
Local time: Thu 2014-02-13 11:16:59 EST
Universal time: Thu 2014-02-13 16:16:59 UTC
RTC time: Thu 2014-02-13 16:17:00
Timezone: America/Jamaica (EST, -0500)
NTP enabled: yes
NTP synchronized: no
RTC in local TZ: no
DST active: n/a
```

2. Display all **systemd** journal entries recorded in the last 30 minutes on serverX.

Assuming the current time is 9:30:00, the following command would be used

```
[root@serverX ~]# journalctl --since 9:00:00 --until 9:30:00
```

3. Configure **rsyslogd** by adding a rule to the newly created configuration file **/etc/rsyslog.d/auth-errors.conf** to log all security and authentication messages with the priority alert and higher to the **/var/log/auth-errors** file as well. Test the newly added log directive with the **logger** command.

- 3.1. Add the directive to log **authpriv.alert** syslog messages to the **/var/log/auth-errors** file in the **/etc/rsyslog.d/auth-errors.conf** configuration file.

```
[root@serverX ~]# echo "authpriv.alert /var/log/auth-errors" >/etc/rsyslog.d/auth-errors.conf
```

- 3.2. Restart the **rsyslog** service on serverX.

```
[root@serverX ~]# systemctl restart rsyslog
```

- 3.3. Monitor the newly created log file **/var/log/auth-errors** on serverX for changes in a different terminal window.

```
[root@serverX ~]# tail -f /var/log/auth-errors
```

- 3.4. Use the **logger** to create a new log entry to the **/var/log/auth-errors** on serverX.

```
[root@serverX ~]# logger -p authpriv.alert "Logging test authpriv.alert"
```

- 3.5. Verify the message sent to syslog with the **logger** command appears in the **/var/log/auth-errors** on serverX in the terminal running **tail -f /var/log/auth-errors**.

```
[root@serverX ~]# tail -f /var/log/auth-errors
Feb 13 11:21:53 server1 root: Logging test authpriv.alert
```

## Summary

### System Log Architecture

The log architecture consists of **systemd-journald** for collecting and **rsyslog** to sort and write log messages to the log files.

### Reviewing Syslog Files

The system log files are maintained by **rsyslog**.

### Reviewing systemd Journal Entries

The systemd journal provides advanced capabilities to query for events.

### Preserving the systemd Journal

Configuring **systemd-journald** to permanently store the journal on disk.

### Maintaining Accurate Time

Time synchronization is important for log file analysis.





## CHAPTER 12

# LOGICAL VOLUME MANAGEMENT

Overview	
<b>Goal</b>	To create and manage logical volumes from the command line.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Manage logical volumes.</li><li>• Extend logical volumes.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Managing Logical Volumes (and Practice)</li><li>• Extending Logical Volumes (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Managing Logical Volume Management (LVM) Storage</li></ul>

# Managing Logical Volumes

## Objectives

After completing this section, students should be able to:

- Implement LVM storage.
- Display LVM component information.

## Implementing LVM storage

LVM comes with 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 **vda** 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.

### Creating a logical volume

There are five steps needed to create a usable logical volume:

#### 1. Prepare the physical device.

Use **fdisk** or **gparted** to create a new partition for use with LVM. Always set the partition type to **Linux LVM** on LVM partitions; use **0x8e** for MBR-style 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@serverX ~]# fdisk /dev/vda
```

Use **m** for help, **p** to print the existing partition table, **n** to create a new partition, **t** to change the partition type, **w** to write the changes, and **q** to quit.

#### 2. Create a physical volume.

**pvcreate** is used to label the partition (or other physical device) for use with LVM as a physical volume. A header to store LVM configuration data is written directly to the PV. A PV is divided into physical extents (PE) of a fixed size; for example, 4MiB blocks. Label multiple devices at the same time by using space-delimited device names as arguments to **pvcreate**.

```
[root@serverX ~]# pvcreate /dev/vda2 /dev/vdb1
```

This will label devices **/dev/vda2** 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.

### 3. Create a volume group.

**vgcreate** is used to create a pool of one or more physical volumes, called a volume group. The size of the VG is determined by the total number of physical extents in the pool. A VG is responsible for hosting one or more logical volumes by allocating free PEs to a LV; therefore, it must have sufficient free PEs available at the time the LV is created.

As arguments to **vgcreate**, define a VG name and list one or more PVs to allocate to the VG.

```
[root@serverX ~]# vgcreate vg-alpha /dev/vda2 /dev/vdb1
```

This will create a VG called **vg-alpha** that is the combined size, in PE units, of the two PVs **/dev/vda2** and **/dev/vdb1**.

A VG only needs to be created when there is none in existence. 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.

### 4. Create a logical volume.

**lvcreate** creates a new logical volume from the available physical extents in a volume group. Use these arguments to **lvcreate** as a minimum: use the **-n** option to set the LV name, the **-L** option to set the LV size in bytes, and identify the VG name that the LV is to be created in.

```
[root@serverX ~]# lvcreate -n hercules -L 2G vg-alpha
```

This will create a LV called **hercules**, **2GiB** in size, in the VG **vg-alpha**. There must be sufficient free physical extents to allocate 2GiB, and if necessary, it will be rounded to a factor of the PE unit size.

There are multiple ways to specify the size: **-L** expects sizes in bytes, or larger named values, such as mebibytes (binary megabytes, 1048576 bytes) and gibibytes (binary gigabytes). The **-l** option expects sizes measured as a number of physical extents.

*Some examples:*

- **lvcreate -L 128M**: Size the logical volume to exactly 128MiB.
- **lvcreate -l 128** : Size the logical volume to exactly 128 extents in size. The total number of bytes depends on the size of the physical extent block on the underlying physical volume.



## Important

Different tools will display the logical volume name using either the traditional name, **/dev/vgname/lvname**, or the kernel device mapper name, **/dev/mapper/vgname-lvname**.

### 5. 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@serverX ~]# mkfs -t xfs /dev/vg-alpha/hercules
```

To make the file system available across reboots:

- Use **mkdir** to create a mount point directory.

```
[root@serverX ~]# mkdir /mnt/hercules
```

- Add an entry to the **/etc/fstab** file:

```
/dev/vg-alpha/hercules /mnt/hercules xfs defaults 1 2
```

- Run **mount -a** to mount all the file systems in **/etc/fstab**, including the entry just added.

```
[root@serverX ~]# mount -a
```

### Removing a logical volume

There are four steps needed to remove *all* logical volume components:

#### 1. Prepare the file system.

Move all data that must be kept to another file system, then use **umount** to unmount the file system. Do not forget to remove any **/etc/fstab** entries associated with this file system.

```
[root@serverX ~]# umount /mnt/hercules
```



## Warning

Removing a logical volume will destroy any data stored on the logical volume. Back up or move your data *BEFORE* you remove the logical volume.

#### 2. Remove the logical volume.



**lvremove** is used to remove a logical volume that is no longer needed. Use the device name as the argument.

```
[root@serverX ~]# lvremove /dev/vg-alpha/hercules
```

The LV file system must be unmounted before running this command. It will ask for confirmation before removing the LV.

The LV's physical extents will be freed and made available for assignment to existing or new LVs in the volume group.

### 3. Remove the volume group.

**vgremove** is used to remove a volume group that is no longer needed. Use the VG name as the argument.

```
[root@serverX ~]# vgremove vg-alpha
```

The VG's physical volumes will be freed and made available for assignment to existing or new VGs on the system.

### 4. Remove the physical volumes.

**pvremove** is used 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. The PV metadata is wiped from the partition (or disk). The partition is now free for reallocation or reformatting.

```
[root@serverX ~]# pvremove /dev/vda2 /dev/vdb1
```

## Reviewing LVM status information

### Physical volumes

Use **pvdisplay** to display information about physical volumes. If no arguments are specified with the command, it will list information about all PVs on the system. If the argument is a specific device name, then display information will be limited to that specific PV.

```
[root@serverX ~]# pvdisplay /dev/vda2
--- Physical volume ---
PV Name               /dev/vda2
VG Name               vg-alpha
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               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 4MiB (the *PE Size*) gives 104MiB of free space. A logical volume size will be 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. If no argument is specified for the command, then it will display information about all VGs. Using the VG name as an argument will limit the display information to that specific VG.

```
[root@serverX ~]# vgdisplay vg-alpha
--- Volume group ---

VG Name                vg-alpha ❶
System ID
Format                 lvm2
Metadata Areas         3
Metadata Sequence No   4
VG Access               read/write
VG Status               resizable
MAX LV                 0
Cur LV                 1
Open LV                 1
Max PV                 0
Cur PV                 3
Act PV                 3

VG Size                 1012.00 MiB ❷
PE Size                 4.00 MiB

Total PE                253 ❸
Alloc PE / Size         175 / 700.00 MiB
Free PE / Size           78 / 312.00 MiB ❹
VG UUID                 3snNw3-CF71-CcYG-Llk1-p6EY-rHEv-xfUSez
```

- ❶ **VG Name** is the name of this 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. Again, no argument with the command will display information about all LVs, and using the LV device name as an argument will display information about that specific device.

```
[root@serverX ~]# lvdisplay /dev/vg-alpha/hercules
--- Logical volume ---

LV Path                 /dev/vg-alpha/hercules ❶
```

LV Name	hercules	
VG Name	vg-alpha	2
LV UUID	5IyRea-W8Zw-xLHk-3h2a-IuVN-YaeZ-i3IRrN	
LV Write Access	read/write	
LV Creation host, time	server1.example.com 2014-02-19 00:26:48 -0500	
LV Status	available	
# open	1	
LV Size	700 MiB	3
Current LE	175	4
Segments	3	
Allocation	inherit	
Read ahead sectors	auto	
- current set to	8192	
Block device	252:0	

- 1 **LV Path** shows the device name of this logical volume.

Some tools may report the device name as `/dev/mapper/vgname-lvname`; both represent the same LV.

- 2 **VG Name** shows the volume group the LV is allocated from.
- 3 **LV Size** shows the total size of the LV. Use file system tools to check free space and used space for storage of data.
- 4 **Current LE** shows the number of logical extents used by this LV. A 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)`, `gparted(8)`, `partprobe(8)`, and `mkfs(8)` man pages

## Practice: Adding a Logical Volume

### Guided exercise

In this lab, you will add a physical volume, volume group, logical volume, and an XFS file system. You will persistently mount the logical volume file system.

#### Resources:

#### Machines:

serverX

#### Outcomes:

A 400MiB logical volume called **storage** in the volume group **shazam**, mounted at **/storage**. The volume group consists of two physical volumes, each 256MiB in size.

#### Before you begin...

- Reset your serverX system.
- Log into serverX.
- Open a terminal.
- Switch to root (**sudo -i**).



### Important

The following examples use device **vdb**, but your environment may have different device names. Adjust the device name as necessary in each step.

- 1. Create the Physical Resources
  - 1.1. Use **fdisk** to create two partitions of 256MiB apiece and set them to type Linux LVM.

```
[root@serverX ~]# fdisk /dev/vdb
```

**Note:** The following steps omit some output.

- 1.2. Add a new primary partition of 256MiB.

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): Enter
Using default response p
Partition number (1-4, default 1): Enter
First sector (2048-20971519, default 2048): Enter
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default
20971519): +256M
```

- 1.3. Change the partition type to *Linux LVM - 0x8e*.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'
```

- ❑ 1.4. Repeat the previous two steps to add a second primary partition of the same size in the next available partition space.
- ❑ 1.5. Write the changes to the partition table and quit.

```
Command (m for help): w
The partition table has been altered!
```

- ❑ 1.6. Use **partprobe** to register the new partitions with the kernel.

```
[root@serverX ~]# partprobe
```

- ❑ 2. Create the Physical Volumes  
Use **pvcreeate** to add the two new partitions as PVs.

```
[root@serverX ~]# pvcreeate /dev/vdb1 /dev/vdb2
Physical volume "/dev/vdb1" successfully created
Physical volume "/dev/vdb2" successfully created
```

- ❑ 3. Create the Volume Group  
Use **vgcreate** to create a new VG named **shazam** built from the two PVs.

```
[root@serverX ~]# vgcreate shazam /dev/vdb1 /dev/vdb2
Volume group "shazam" successfully created
```

- ❑ 4. Create the Logical Volume  
Use **lvcreate** to create a 400MiB LV named **storage** from the **shazam** VG.

```
[root@serverX ~]# lvcreate -n storage -L 400M shazam
Logical volume "storage" created
```

This will create a device called **/dev/shazam/storage**, currently without a file system on it.

- ❑ 5. Add a Persistent File System
  - ❑ 5.1. Use **mkfs** to place an **xfs** file system on the **storage** LV; use the LV device name.

```
[root@serverX ~]# mkfs -t xfs /dev/shazam/storage
meta-data=/dev/shazam/storage    isize=256    agcount=4, agsize=25600 blks
...
```

- ❑ 5.2. Use **mkdir** to create a mount point at **/storage**.

```
[root@serverX ~]# mkdir /storage
```

- 5.3. Use **vim** to add the following line to the bottom of **/etc/fstab** on serverX:

```
/dev/shazam/storage    /storage    xfs    defaults    1 2
```

- 5.4. Use **mount** to verify the **/etc/fstab** entry and mount the new **storage** LV device.

```
[root@serverX ~]# mount -a
```

- 6. Test and Review Your Work

- 6.1. As a final test, copy some files onto **/storage** and verify how many were copied.

```
[root@serverX ~]# cp -a /etc/*.conf /storage
[root@serverX ~]# ls /storage | wc -l
47
```

We will check that we still have the same number of files in the next practice exercise.

- 6.2. **fdisk -l /dev/vdb** will show you the partitions that exist on **/dev/vdb**.

```
[root@serverX ~]# fdisk -l /dev/vdb
```

Check the **/dev/vdb1** and **/dev/vdb2** entries, and notice the **Id** and **System** columns showing **8e** and **Linux LVM**, respectively.

- 6.3. **pvdisplay** will show you information about each of the physical volumes. Optionally, include the device name to limit details to a specific PV.

```
[root@serverX ~]# pvdisplay /dev/vdb2
--- Physical volume ---
PV Name               /dev/vdb2
VG Name               shazam
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               N64t6x-URdJ-fVU3-FQ67-zU6g-So7w-hvXMmC
```

This shows that our PV is allocated to VG *shazam*, is 256MiB in size (although 4MiB is not usable), and our physical extent size (**PE Size**) is 4MiB (the smallest allocatable LV size).

There are 63 PEs, of which 26 PEs are free for allocation to LVs in the future and 37 PEs are currently allocated to LVs. These translate to MiB values as follows:

- Total 252MiB (63 PEs x 4MiB); remember, 4MiB are unusable.

- Free 104MiB (26 PEs x 4MiB)
  - Allocated 148MiB (37 PEs x 4MiB)
- ❑ 6.4. **vgdisplay *vgname*** will show you information about the volume group named ***vgname***.

```
[root@serverX ~]# vgdisplay shazam
```

Check 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**.
- ❑ 6.5. **lvdisplay /dev/*vgname*/*lvname*** will show you information about the logical volume named ***lvname***.

```
[root@serverX ~]# lvdisplay /dev/shazam/storage
```

Notice the **LV Path**, **LV Name**, **VG Name**, **LV Status**, **LV Size**, and **Current LE** (logical extents, which map to physical extents).

- ❑ 6.6. **mount** will show all the devices that are mounted and any mount options. It should include **/dev/shazam/storage**.



### Note

**Reminder:** Many tools will report the device mapper name instead, **/dev/mapper/shazam-storage**; it is the same logical volume.

```
[root@serverX ~]# mount
```

You should see (probably on the last line) **/dev/mapper/shazam-storage** mounted on **/storage** and the associated mount information.

- ❑ 6.7. **df -h** will show human-readable disk free space. Optionally, include the mount point to limit details to that file system.

```
[root@serverX ~]# df -h /storage
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/shazam-storage 397M   21M  377M   6% /storage
```

Allowing for file system metadata, these values are what we would expect.

# Extending Logical Volumes

## Objectives

After completing this section, students should be able to:

- Extend and reduce a volume group.
- Extend a LV with an XFS file system.
- Extend an LV with an ext4 file system.

## Extending and reducing a volume group

A volume group can have more disk space added to it by adding additional physical volumes. This is called extending the volume group. The new physical extents provided by the additional physical volumes can then be assigned to logical volumes.

Unused physical volumes can be removed from a volume group. This is called reducing the volume group. A tool called *pvmove* can be used 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. This can be done while the logical volumes in the volume group are in use.



### 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.

### Extending a volume group

There are potentially four steps needed to extend a volume group:

#### 1. Prepare the physical device.

As with creating a new volume group, a new partition must be created and prepared for use as an LVM physical volume.

Use **fdisk** or **gparted** to create a new partition for use with LVM. Always set the partition type to **Linux LVM** on LVM partitions; use **0x8e** for MBR-style 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 extend the volume group.

```
[root@serverX ~]# fdisk /dev/vdb
```

Use **m** for help, **p** to print the existing partition table, **n** to create a new partition, **t** to change the partition type, **w** to write the changes, and **q** to quit.



## 2. Create the physical volume.

**pvcreate** is used to label the partition (or other physical device) for use with LVM as a physical volume. A header to store LVM configuration data is written directly to the PV. A PV is divided into physical extents of a fixed size; for example, 4MiB blocks. Use the device name as the argument to **pvcreate**.

```
[root@serverX ~]# pvcreate /dev/vdb2
```

This will label device **/dev/vdb2** as a PV, ready for allocation into the volume group.

A PV only needs to be created if there are no PVs free to extend the VG.

## 3. Extend the volume group.

**vgextend** is used to add the new physical volume to the volume group. Use the VG name and PV device name as arguments to **vgextend**.

```
[root@serverX ~]# vgextend vg-alpha /dev/vdb2
```

This will extend the **vg-alpha** VG by the size of the **/dev/vdb2** PV.

## 4. Verify the new space is available.

Use **vgdisplay** to confirm the additional physical extents are available. Check the **Free PE / Size** in the output. It should not be zero.

```
[root@serverX ~]# vgdisplay vg-alpha
--- Volume group ---
VG Name                vg-alpha
...
Free  PE / Size        178 / 712.00 MiB
...
```

## Reducing a volume group

There are only two steps needed to reduce a volume group:

### 1. Move the physical extents.

**pvmove** is used to relocate any physical extents used on the physical volume to other PVs in the VG. This is only possible if there are enough free extents in the VG and if all of those come from other PVs. Use the PV device name for which the PEs will be moved as the argument to the command.

```
[root@serverX ~]# pvmove /dev/vdb2
```

This will move the PEs from **/dev/vdb2** to other PVs with free PEs in the same VG.



## Warning

Before using **pvmove**, it is recommended to 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.

### 2. Reduce the volume group.

**vgreduce** is used to remove the physical volume from the volume group. Use the VG name and PV device name as arguments to the command.

```
[root@serverX ~]# vgreduce vg-alpha /dev/vdb2
```

The **/dev/vdb2** PV is now removed from the **vg-alpha** VG and can be added to another VG. Alternatively, **pvremove** can be used to stop using the device as a PV permanently.

## Extend 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

There are three steps needed to extend a logical volume:

#### 1. Verify the volume group has space available.

**vgdisplay** is used to verify that there are sufficient physical extents available for use.

```
[root@serverX ~]# vgdisplay vg-alpha
--- Volume group ---
VG Name                vg-alpha
...
Free  PE / Size        178 / 712.00 MiB
...
```

Check the **Free PE / Size** in the output. It should report a value equal to or more than the additional space required. If there is insufficient space available, then extend the volume group by at least the required space. See "Extending and Reducing a Volume Group."

#### 2. Extend the logical volume.

**lvextend** extends the logical volume to a new size. Add the LV device name as the last argument to the command.

```
[root@serverX ~]# lvextend -L +300M /dev/vg-alpha/hercules
```

This will increase the size of logical volume **hercules** by 300MiB. Notice the "+" in front of the size, which means add this value to the existing size; otherwise, the value defines the final, exact size of the LV.

Like **lvcreate**, there are multiple ways to specify the size: **-l** generally expects physical extent values, while **-L** expects sizes in bytes or larger named values, such as mebibytes and gibibytes.

Some examples:

- **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* 128MiB.
- **lvextend -L +128M**: Add 128MiB to the current size of the logical volume.
- **lvextend -l +50%FREE**: Add 50 percent of the current free space in the VG to the LV.

### 3. Extend the file system.

**xfs\_growfs /mountpoint** expands the file system to occupy the extended LV.

**xfs\_growfs** requires the file system be mounted while it is being run; it can continue to be used during the resize operation.

```
[root@serverX ~]# xfs_growfs /mnt/hercules
```



#### Note

A common mistake is to run **lvextend**, but forget to run **xfs\_growfs**. An alternative to running the two steps consecutively is to include **-r** as an 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.

- It is a good idea to verify the new size of the mounted file system:

```
df -h /mountpoint.
```

## Extend a logical volume and ext4 file system

### Extending a logical volume

The steps for extending an **ext4**-based logical volume are essentially the same as for an **xfs**-based LV, except the step that resizes the file system. Review "Extend a Logical Volume and XFS File System" for more details.

#### 1. Verify the volume group has space available.

**vgdisplay vgname** is used to verify that there are sufficient physical extents available for use.

#### 2. Extend the logical volume.

**lvextend -l +extents /dev/vgname/lvname** extends the logical volume **/dev/vgname/lvname** by the **extents** value.

### 3. **Extend the file system.**

**resize2fs** **/dev/vgname/lvname** expands the file system to occupy the new extended LV. Just like **xfs\_growfs**, the file system can be mounted and in use while it is being run. Optionally, include the **-p** option to see the progress of the resize operation.

```
[root@serverX ~]# resize2fs /dev/vg-alpha/hercules
```



#### 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.



#### References

**lvm(8)**, **pvcreate(8)**, **pvmove(8)**, **vgdisplay(8)**, **vgextend(8)**, **vgreduce(8)**, **vgdisplay(8)**, **vgextend(8)**, **vgreduce(8)**, **lvextend(8)**, **fdisk(8)**, **gparted(8)**, **partprobe(8)**, **xfs\_growfs(8)**, and **resize2fs(8)** man pages

# Practice: Extending a Logical Volume

## Guided exercise

In this lab, you will extend the logical volume added in the previous practice exercise.

### Resources:

#### Machines:

serverX

### Outcomes:

A resized logical volume, 700MiB total, called **storage** in the volume group **shazam**, mounted at **/storage**. Resizing done while the file system is still mounted and in use. The volume group extended to include an additional physical volume of 512MiB, giving a total VG size of 1GiB.

### Before you begin...

Complete Practice: Adding a Logical Volume



## Important

The following examples use device **vdb**. Your environment may have different device names. Adjust the device name as necessary in each step.

- ❑ 1. Check for Space in the Volume Group  
Use **vgdisplay** to check if the VG has sufficient free space to extend the LV to a total size of 700MiB.

```
[root@serverX ~]# vgdisplay shazam
--- Volume group ---
 VG Name                shazam
 System ID
 Format                 lvm2
 ..
 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
```

There is only 104MiB available (26 PEs x 4MiB extents) and we need at least 300MiB to have 700MiB in total. We need to extend the VG.

For later comparison, use **df** to check current disk free space:

```
[root@serverX ~]# df -h /storage
Filesystem                Size      Used Avail Use% Mounted on
/dev/mapper/shazam-storage 397M      21M   377M   6% /storage
```

- ❑ 2. Create the Physical Resources

Use **fdisk** to create an additional partition of 512MiB and set it to type Linux LVM.

- 2.1. `[root@serverX ~]# fdisk /dev/vdb`

**Note:** The following steps omit some output.

- 2.2. Add a new primary partition of 512MiB.

```
Command (m for help): n
Partition type:
   p   primary (2 primary, 0 extended, 2 free)
   e   extended
Select (default p): Enter
Using default response p
Partition number (3,4, default 3): Enter
First sector (1050624-20971519, default 1050624): Enter
Using default value 1050624
Last sector, +sectors or +size{K,M,G} (1050624-20971519, default
20971519): +512M
Partition 3 of type Linux and of size 512 MiB is set
```

- 2.3. Change the partition type to *Linux LVM - 0x8e*.

```
Command (m for help): t
Partition number (1-3, default 3): Enter
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'
```

- 2.4. Write the changes to the partition table and quit.

```
Command (m for help): w
The partition table has been altered!
```

- 2.5. Use **partprobe** to register the new partitions with the kernel.

```
[root@serverX ~]# partprobe
```

- 3. Create the Physical Volume  
Use **pvcreeate** to add the new partition as a PV.

```
[root@serverX ~]# pvcreeate /dev/vdb3
Physical volume "/dev/vdb3" successfully created
```

- 4. Extend the Volume Group

- 4.1. Use **vgextend** to extend the VG named **shazam**, using the new **/dev/vdb3** PV.

```
[root@serverX ~]# vgextend shazam /dev/vdb3
Volume group "shazam" successfully extended
```

- 4.2. Use **vgdisplay** to check the **shazam** VG free space again. There should be plenty of free space now.

```
[root@serverX ~]# vgdisplay shazam
--- Volume group ---
VG Name                shazam
System ID
Format                 lvm2
...
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
```

There is now 612MiB available (153 PEs x 4MiB extents); perfect.

- 5. Extend the Logical Volume  
Use **lvextend** to extend the existing LV to 700MiB.

```
[root@serverX ~]# lvextend -L 700M /dev/shazam/storage
Extending logical volume storage to 700.00 MiB
Logical volume storage successfully resized
```



### Note

In our example, we specified the exact size to make the final LV, but we could also have used:

- **-L +300M** to add the new space using size in MiB.
- **-l 175** to specify the total number of extents (175 PEs x 4MiB).
- **-l +75** to add the additional extents needed.

- 6. Resize the File System  
Use **xfs\_growfs** to extend the XFS file system to the remainder of the free space on the LV.

```
[root@serverX ~]# xfs_growfs /storage
meta-data=/dev/mapper/shazamstorage isize=256    agcount=4, agsize=25600 blks
...
```

- 7. Verify Content Availability and New File System Size  
Use **df** and **ls | wc** to review the new file system size and verify the existing files are still present.

```
[root@serverX ~]# df -h /storage
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/shazam-storage 684M   21M  663M   6% /storage
[root@serverX ~]# ls /storage | wc -l
47
```

The files are still there and the file system is about the size we expect.



# Lab: Managing Logical Volume Management (LVM) Storage

## Performance checklist

In this lab, you will resize an existing logical volume, adding LVM resources as necessary, and then add a new logical volume with a persistently mounted XFS file system on it.

**Resources:****Machines:****serverX****Outcomes:**

- Logical volume **loans** resized to 768MiB and persistently mounted at **/finance/loans**.
- A new 128MiB logical volume called **risk** with an XFS file system, persistently mounted at **/finance/risk**.

**Before you begin...**

- Reset your serverX system.
- Log into and set up your server system.

```
[student@serverX ~]$ lab lvm setup
```

- Open a terminal.
- Switch to **root** using **sudo -i**.

Your company's finance department has a logical volume called **loans** that is starting to run out of disk space, and you have been asked to extend it to 768MiB in size.

You have also been asked to create a new file system to host documents for the risk management team, part of the finance department; a 128MiB logical volume called **risk** has been agreed upon and it should be mounted on **/finance/risk**. Your company standard file system is XFS.

There is a volume group called **finance** used to hold the department's logical volumes, but unfortunately it has insufficient space to extend the existing logical volume and add the new one, so you have been allocated a further 512MiB from the current hard disk. The partition needs to be created.

When you are done, reboot your **serverX** machine, then run the command **lab lvm grade** from your **serverX** machine to verify your work.

1. Create a 512MiB partition on **/dev/vdb**, initialize it as a physical volume, and extend the **finance** volume group with it.
2. Extend the **loans** logical volume to 768MiB, including the file system.
3. In the existing volume group, create a new logical volume called **risk** with a size of 128MiB. Add an XFS file system and mount it persistently on **/finance/risk**.

4. When you are done, reboot your **serverX** machine, then run the command **lab lvm grade** from your **serverX** machine to verify your work.

## Solution

In this lab, you will resize an existing logical volume, adding LVM resources as necessary, and then add a new logical volume with a persistently mounted XFS file system on it.

### Resources:

#### Machines:

serverX

### Outcomes:

- Logical volume **loans** resized to 768MiB and persistently mounted at **/finance/loans**.
- A new 128MiB logical volume called **risk** with an XFS file system, persistently mounted at **/finance/risk**.

### Before you begin...

- Reset your serverX system.
- Log into and set up your server system.

```
[student@serverX ~]$ lab lvm setup
```

- Open a terminal.
- Switch to **root** using **sudo -i**.

Your company's finance department has a logical volume called **loans** that is starting to run out of disk space, and you have been asked to extend it to 768MiB in size.

You have also been asked to create a new file system to host documents for the risk management team, part of the finance department; a 128MiB logical volume called **risk** has been agreed upon and it should be mounted on **/finance/risk**. Your company standard file system is XFS.

There is a volume group called **finance** used to hold the department's logical volumes, but unfortunately it has insufficient space to extend the existing logical volume and add the new one, so you have been allocated a further 512MiB from the current hard disk. The partition needs to be created.

When you are done, reboot your **serverX** machine, then run the command **lab lvm grade** from your **serverX** machine to verify your work.

1. Create a 512MiB partition on **/dev/vdb**, initialize it as a physical volume, and extend the **finance** volume group with it.
  - 1.1. Use **fdisk** to create the 512MiB partition and set it to type Linux LVM.

```
[root@serverX ~]# fdisk /dev/vdb
```

**Note:** The following steps omit some output.

- 1.2. Add a new primary partition of 512MiB.

```
Command (m for help): n
```

```

Partition type:
  p   primary (1 primary, 0 extended, 3 free)
  e   extended
Select (default p): Enter
Partition number (2-4, default 2): Enter
First sector (1050624-20971519, default 1050624): Enter
Last sector, +sectors or +size{K,M,G} (1050624-20971519, default
20971519): +512M

```

- 1.3. Change the partition type to *Linux LVM - 0x8e*.

```

Command (m for help): t
Partition number (1,2, default 2): Enter
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'

```

- 1.4. Write the changes to the partition table and quit.

```

Command (m for help): w
The partition table has been altered!

```

- 1.5. Use **partprobe** to register the new partition with the kernel.

```
[root@serverX ~]# partprobe
```

- 1.6. Use **pvccreate** to initialize the partition as a PV.

```
[root@serverX ~]# pvccreate /dev/vdb2
Physical volume "/dev/vdb2" successfully created

```

- 1.7. Use **vgextend** to extend the VG named **finance**, using the new **/dev/vdb2** PV.

```
[root@serverX ~]# vgextend finance /dev/vdb2
Volume group "finance" successfully extended

```

2. Extend the **loans** logical volume to 768MiB, including the file system.

- 2.1. Use **lvextend** to extend the **loans** LV to 768MiB.

```
[root@serverX ~]# lvextend -L 768M /dev/finance/loans
Extending logical volume loans to 768.00 MiB
Logical volume loans successfully resized

```



### Note

Alternatively, you could have used **-L +512M** to resize the LV.

- 2.2. Use **xfs\_growfs** to extend the XFS file system to the remainder of the free space on the LV.

```
[root@serverX ~]# xfs_growfs /finance/loans
meta-data=/dev/mapper/finance-loans isize=256    agcount=4, agsize=16384 blks
...
```



### Note

This example shows the **xfs\_growfs** step to extend the file system. An alternative would have been to add the **"-r"** option to the **lvextend** command.

3. In the existing volume group, create a new logical volume called **risk** with a size of 128MiB. Add an XFS file system and mount it persistently on **/finance/risk**.

- 3.1. Use **lvcreate** to create a 128MiB LV named **risk** from the **finance** VG.

```
[root@serverX ~]# lvcreate -n risk -L 128M finance
Logical volume "risk" created
```

- 3.2. Use **mkfs** to place an **xfs** file system on the **risk** LV; use the LV device name.

```
[root@serverX ~]# mkfs -t xfs /dev/finance/risk
meta-data=/dev/finance/risk    isize=256    agcount=4, agsize=8192 blks
...
```

- 3.3. Use **mkdir** to create a mount point at **/finance/risk**.

```
[root@serverX ~]# mkdir /finance/risk
```

- 3.4. Use **vim** to add the following line to the bottom of **/etc/fstab** on serverX:

```
/dev/finance/risk /finance/risk xfs defaults 1 2
```

- 3.5. Use **mount** to verify the **/etc/fstab** entry and mount the new **risk** LV device.

```
[root@serverX ~]# mount -a
```

4. When you are done, reboot your **serverX** machine, then run the command **lab lvm grade** from your **serverX** machine to verify your work.

- 4.1. 

```
[root@serverX ~]$ systemctl reboot
```

- 4.2. 

```
[student@serverX ~]$ lab lvm grade
```

## Summary

### Managing Logical Volumes

- **pvccreate**, **pvremove**, and **pvdiskdisplay** create, remove, and list physical volumes (PV).
- **vgcreate**, **vgremove**, and **vgdisplay** create, remove, and list volume groups (VG).
- **lvcreate**, **lvremove**, and **lvdisplay** create, remove, and list logical volumes (LV).
- Adding logical volumes is done in the order PV, VG, and LV.
- Removing logical volumes is done in the order LV, VG, and PV.

### Extending Logical Volumes

- Extend a volume group (VG) using **pvccreate** and **vgextend**; use **vgdisplay** to check the results.
- Reduce a VG using **pvmove** and **vgreduce**.
- Extend a logical volume (LV) using **lvextend**.
- Use **xfs\_growfs** to resize **xfs** file systems.
- Use **resize2fs** to resize **ext4** file systems.



## CHAPTER 13

# SCHEDULED PROCESSES

Overview	
<b>Goal</b>	Schedule tasks to automatically execute in the future.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Schedule recurring jobs with cron.</li><li>• Manage temporary files.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Scheduling Recurring Jobs with cron (and Practice)</li><li>• Managing Temporary Files (and Practice)</li></ul>

## Scheduling System cron Jobs

### Objectives

After completing this section, students should be able to:

- Schedule recurring system tasks.

### System cron jobs

Apart from *user* **cron** jobs, there are also *system* **cron** jobs.

System cron jobs are not defined using the **crontab** command, but are instead configured in a set of configuration files. The main difference in these configuration files is an extra field, located between the **Day-of-Week** field and the **Command** field, specifying under which user a job should be run.

The **/etc/crontab** 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, wed, thu, fri, sat
# | | | | |
# * * * * * user-name command to be executed
```

System **cron** jobs are defined in two locations: **/etc/crontab** and **/etc/cron.d/\***. Packages that install **cron** jobs should do so by placing a file in **/etc/cron.d/**, but administrators can also use this location to more easily group related jobs into a single file, or to push jobs using a configuration management system.

There are also predefined jobs that run every hour, day, week, and month. These jobs will execute all scripts placed in **/etc/cron.hourly/**, **/etc/cron.daily/**, **/etc/cron.weekly/**, and **/etc/cron.monthly** respectively. Please note that these directories contain *executable scripts*, and not **cron** configuration files.



### Important

Make sure to make any scripts you place in these directories executable. If a script is not made executable (e.g., with **chmod +x**), it will not be run.

The **/etc/cron.hourly/\*** scripts are executed using the **run-parts** command, from a job defined in **/etc/cron.d/0hourly**. The daily, weekly, and monthly jobs are also executed using the **run-parts** command, but from a different configuration file: **/etc/anacrontab**.

In the past, **/etc/anacrontab** was handled by a separate daemon (**anacron**), but in Red Hat Enterprise Linux 7, the file is parsed by the regular **crond** daemon. The purpose of this file is to make sure that important jobs will always be run, and not skipped accidentally because the system was turned off or hibernating when the job should have been executed.



The syntax of **/etc/anacrontab** is different from the other **cron** configuration files. It contains exactly four fields per line:

- **Period in days**

Once per how many days this job should be run.

- **Delay in minutes**

The amount of time the **cron** daemon should wait before starting this job.

- **Job identifier**

This is the name of the file in **/var/spool/anacron/** that will be used to check if this job has run. When **cron** starts a job from **/etc/anacrontab**, it will update the timestamp on this file. The same timestamp is used to check when a job has last run in the past.

- **Command**

The command to be executed

**/etc/anacrontab** also contains environment variable declarations using the syntax **NAME=value**. Of special interest is **START\_HOURS\_RANGE**: Jobs will **not** be started outside of this range.



## References

**crond(8)**, **crontab(1)**, and **crontab(5)**, **anacron(8)**, and **anacrontab(5)** man pages

## Practice: Scheduling System cron Jobs

### Guided exercise

In this lab, you will work with recurring system jobs.

Resources	
<b>Files:</b>	<ul style="list-style-type: none"> <li>• <code>/etc/crontab</code></li> <li>• <code>/etc/cron.d/*</code></li> <li>• <code>/etc/cron.{hourly,daily,weekly,monthly}/*</code></li> </ul>
<b>Machines:</b>	<code>desktopX</code>

#### Outcomes

A daily job to count the number of active users, and an updated **cron** job to gather system performance data.

- ❑ 1. Log into your **desktopX** system as **student**, then elevate your privileges to **root**.

- ❑ 1.1.

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

- ❑ 2. Create a new daily **cron** job that logs a message to the system log with the number of currently active users (`w -h | wc -l`). You can use the **logger** command to send messages to the system log.

- ❑ 2.1. Open a new file in `/etc/cron.daily` in an editor, e.g., `/etc/cron.daily/usercount`.

```
[root@desktopX ~]# vim /etc/cron.daily/usercount
```

- ❑ 2.2. Write the script that logs the number of active users to the system log.

Insert the following in your editor:

```
#!/bin/bash
USERCOUNT=$(w -h | wc -l)
logger "There are currently ${USERCOUNT} active users"
```

- ❑ 2.3. Make the script executable:

```
[root@desktopX ~]# chmod +x /etc/cron.daily/usercount
```

- ❑ 3. The **sysstat** package, when installed, has a cron job that runs every 10 minutes, collecting data using a command called **sa1**. Make sure this package is installed, then change this job to run every five minutes.

- ❑ 3.1. Make sure the **sysstat** package is installed.

```
[root@desktopX ~]# yum -y install sysstat
```

- ❑ 3.2. Find out in which file the *sysstat* package has configured the **cron** jobs. Cron jobs are generally configured in files marked as a configuration file for the package manager.

```
[root@desktopX ~]# rpm -qc sysstat
```

**/etc/cron.d/sysstat** looks promising.

- ❑ 3.3. Open **/etc/cron.d/sysstat** in an editor.

```
[root@desktopX ~]# vim /etc/cron.d/sysstat
```

- ❑ 3.4. Change **\*/10** on the **sa1** line to **\*/5**.
- ❑ 3.5. Monitor the files in **/var/log/sa** to see when their sizes and timestamps change.

```
[root@desktopX ~]# watch ls -l /var/log/sa
```

- ❑ 3.6. Save your changes and exit.

# Managing Temporary Files

## Objectives

After completing this section, students should be able to manage temporary files using **systemd-tmpfiles**.

## Managing temporary files with systemd-tmpfiles

A modern system requires a large number of temporary files and directories. Not just the highly user-visible ones such as **/tmp** that get used and abused by regular users, but also more task-specific ones 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, since 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.

In the past, system administrators relied on RPM packages and SystemV init-scripts to create these directories, and a tool called **tmpwatch** to remove old, unused files from configured directories.

In Red Hat Enterprise Linux 7 **systemd** provides a more structured, and configurable, method to manage temporary directories and files: **systemd-tmpfiles**.

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 will be removed, and any files and directories marked for creation (or permission fixes) will be created with the correct permissions if necessary.

### Regular cleaning

To make sure that long-running systems do not fill up their disks with stale data, there is also *systemd timer unit* that calls **systemd-tmpfiles --clean** on a regular interval.

**systemd** timer units are a special type of **systemd** service that have a **[Timer]** block indicating how often the service with the same name should be started.

On a Red Hat Enterprise Linux 7 system, the configuration for the **systemd-tmpfiles-clean.timer** unit looks like this:

```
[Timer]
OnBootSec=15min
OnUnitActiveSec=1d
```

This indicates that the service with the same name (**systemd-tmpfiles-clean.service**) will be started 15 minutes after **systemd** has started, and then once every 24 hours afterwards.

The command **systemd-tmpfiles --clean** parses the same configuration files as the **systemd-tmpfiles --create**, 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.



## Important

The man page **tmpfiles.d(5)** claims that files "older than" the age in the date field of the configuration file are removed. This is not exactly true.

Files on a Linux file system following the POSIX standard have three timestamps: **atime**, the last time the file was accessed, **mtime**, the last time the file's contents were modified, and **ctime**, the last time the file's status was changed (by **chown**, **chmod**, and so on). Most Linux file systems do not have a creation time stamp. This is common among Unix-like file systems.

Files will be considered unused if *all three* timestamps are older than the **systemd-tmpfiles** age configuration. If *any* of the three timestamps are newer than the age configuration, the file will not be removed due to age by **systemd-tmpfiles**.

The **stat** command can be run on a file to see the values of all three of its time stamps. The **ls -l** command normally displays **mtime**.

### systemd-tmpfiles configuration files

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.

Some examples with explanations:

```
d /run/systemd/seats 0755 root root -
```

When creating files and directories, create the directory **/run/systemd/seats** 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 directory **/home/student** 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 live 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 should not be edited by system administrators. The files under **/run/tmpfiles.d/** are themselves volatile files, normally used by daemons to manage their own runtime temporary files, and 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/** will be 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/** will be used.

Given these precedence rules, an administrator 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 out of 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

# Practice: Managing Temporary Files

## Guided exercise

In this lab, you will configure your system to purge files older than 5 days from **/tmp**. You will also add a new temporary directory called **/run/gallifrey** to be automatically created, with files which have been unused for more than 30 seconds being automatically purged.

Resources	
<b>Files:</b>	<ul style="list-style-type: none"> <li>• <b>/etc/tmpfiles.d/</b></li> <li>• <b>/usr/lib/tmpfiles.d/tmp.conf</b></li> </ul>
<b>Machines:</b>	<b>serverX</b>

### Outcomes:

A new temporary directory called **/run/gallifrey**, set up for automatic purging, and a modified purging configuration for **/tmp**.

### Before you begin...

Reset your **serverX** system.

In production, you have run into a number of issues:

- **/tmp** is running out of disk space. It seems that allowing files to be unused for 10 days before they are deleted is too long for your site. You have determined that deleting files after five days of disuse is acceptable.
- Your time-travel research daemon **gallifrey** needs a separate temporary directory called **/run/gallifrey**. Files in this directory should be purged automatically after they have been unused for 30 seconds. Only **root** should have read and write access to **/run/gallifrey**.

- ❑ 1. **/tmp** is under **systemd-tmpfiles** control. To override the upstream settings, copy **/usr/lib/tmpfiles.d/tmp.conf** to **/etc/tmpfiles.d/**.

❑ 1.1. 

```
[student@serverX ~]$ sudo cp /usr/lib/tmpfiles.d/tmp.conf /etc/tmpfiles.d/
```

- ❑ 2. Find the line in **/etc/tmpfiles.d/tmp.conf** that controls the purging interval for **/tmp**, and change the interval from **10d** to **5d**.

- ❑ 2.1. Open **/etc/tmpfiles.d/tmp.conf** in an editor and make the change, or:

```
[student@serverX ~]$ sudo sed -i '/^d .tmp /s/10d/5d/' /etc/tmpfiles.d/tmp.conf
```

- ❑ 3. Test if **systemd-tmpfiles --clean** accepts the new configuration.

```
[student@serverX ~]$ sudo systemd-tmpfiles --clean tmp.conf
```

- ❑ 4. Create a new configuration file **/etc/tmpfiles.d/gallifrey.conf** with the following content:

```
# Set up /run/gallifrey, owned by root with 0700 permissions
# Files not used for 30 seconds will be automatically deleted
d /run/gallifrey 0700 root root 30s
```

- ❑ 5. Test your new configuration for creating **/run/gallifrey**.

❑ 5.1. 

```
[student@serverX ~]$ sudo systemd-tmpfiles --create gallifrey.conf
```

❑ 5.2. 

```
[student@serverX ~]$ ls -ld /run/gallifrey
drwx-----. 2 root root Feb 19 10:29 /run/gallifrey
```

- ❑ 6. Test the purging of your **/run/gallifrey** directory.

- ❑ 6.1. Create a new file under **/run/gallifrey**.

```
[student@serverX ~]$ sudo touch /run/gallifrey/companion
```

- ❑ 6.2. Wait for at least 30 seconds.

```
[student@serverX ~]$ sleep 30s
```

- ❑ 6.3. Have **systemd-tmpfiles** clean the **/run/gallifrey** directory.

```
[student@serverX ~]$ sudo systemd-tmpfiles --clean gallifrey.conf
```

- ❑ 6.4. Inspect the contents of **/run/gallifrey**.

```
[student@serverX ~]$ sudo ls -l /run/gallifrey
```



## Summary

### Scheduling System cron Jobs

- System crontabs have an extra column: **Username**.
- System crontab files in **/etc/crontab** and **/etc/cron.d/\***.
- Scripts controlled by **/etc/anacrontab** in **/etc/cron.{hourly,daily,weekly,monthly}/**.

### Managing Temporary Files

- **systemd-tmpfiles** is used to manage temporary files and volatile storage.
- Called during boot from **systemd-tmpfiles-setup.service**.
- Called at regular intervals from **systemd-tmpfiles-clean.timer**.
- Configured from **/usr/lib/tmpfiles.d/\*.conf** and **/etc/tmpfiles.d/\*.conf**.
- Files in **/etc/tmpfiles.d/** take precedence over similarly named files in **/usr/lib/tmpfiles.d/**.





## CHAPTER 14

# MOUNTING NETWORK FILE SYSTEMS

Overview	
<b>Goal</b>	To use <code>autofs</code> and the command line to mount and unmount network storage with NFS and SMB.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Mount, access, and unmount network storage with NFS.</li><li>• Automount and access network storage with NFS.</li><li>• Mount, automount, and unmount SMB file systems.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Mounting Network Storage with NFS (and Practice)</li><li>• Automounting Network Storage with NFS (and Practice)</li><li>• Accessing Network Storage with SMB (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Accessing Network Storage with Network File System (NFS)</li><li>• Accessing Network Storage with SMB</li></ul>

# Mounting Network Storage with NFS

## Objectives

After completing this section, students should be able to manually mount, access, and unmount an NFS share.

## Manually 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 under active extension which supports native Linux permissions and file system features.

Red Hat Enterprise Linux supports NFSv4 (version 4 of the protocol) by default, and falls back automatically to NFSv3 and NFSv2 if that is not available. NFSv4 uses the TCP protocol to communicate with the server, while older versions of NFS may use either TCP or UDP.

NFS servers *export* shares (directories) and NFS clients mount an exported share to a local mount point (directory). The local mount point must exist. NFS shares can be mounted a number of ways:

- manually mounting an NFS share using the **mount** command.
- automatically mounting an NFS share at boot time using **/etc/fstab**.
- mounting an NFS share on demand through a process known as *automounting*.

### Securing file access on NFS shares

NFS servers secure access to files using a number of methods: **none**, **sys**, **krb5**, **krb5i**, and **krb5p**. The NFS server can choose to offer a single method or multiple methods for each exported share. NFS clients must connect to the exported share using one of the methods mandated for that share, specified as a mount option **sec=method**.

### Security methods

- **none**: anonymous access to the files, writes to the server (if allowed) will be allocated UID and GID of **nfsnobody**.
- **sys**: file access based on standard Linux file permissions for UID and GID values. If not specified, this is the default.
- **krb5**: Clients must prove identity using Kerberos and then standard Linux file permissions apply.
- **krb5i**: adds a cryptographically strong guarantee that the data in each request has not been tampered with.
- **krb5p**: adds encryption to all requests between the client and the server, preventing data exposure on the network. This will have a performance impact.



## Important

Kerberos options will require, as a minimum, a **/etc/krb5.keytab** and additional authentication configuration that is not covered in this section (joining the Kerberos Realm). The **/etc/krb5.keytab** will normally be provided by the authentication or security administrator. Request a **keytab** that includes either a *host principal*, *nfs principal*, or (ideally) both.

NFS uses the **nfs-secure** service to help negotiate and manage communication with the server when connecting to Kerberos-secured shares. It must be running to use the secured NFS shares; **start** and **enable** it to ensure it is always available.

```
[student@desktopX ~]$ sudo systemctl enable nfs-secure
ln -s '/usr/lib/systemd/system/nfs-secure.service' ...
[student@desktopX ~]$ sudo systemctl start nfs-secure
```



## Note

The **nfs-secure** service is part of the **nfs-utils** package, which should be installed by default. If it is not installed, use:

```
[student@desktopX ~]$ sudo yum -y install nfs-utils
```

## Mount an NFS share

There are three basic steps to mounting an NFS share:

1. **Identify:** The administrator for the NFS server can provide export details, including security requirements. Alternatively:

NFSv4 shares can be identified by mounting the root folder of the NFS server and exploring the exported directories. Do this as **root**. Access to shares that are using Kerberos security will be denied, but the share (directory) name will be visible. Other share directories will be browsable.

```
[student@desktopX ~]$ sudo mkdir /mountpoint
[student@desktopX ~]$ sudo mount serverX:/ /mountpoint
[student@desktopX ~]$ sudo ls /mountpoint
```

NFSv2 and NFSv3 shares can be discovered using **showmount**.

```
[student@desktopX ~]$ showmount -e serverX
```

2. **Mount point:** Use **mkdir** to create a mount point in a suitable location.

```
[student@desktopX ~]$ mkdir -p /mountpoint
```

3. **Mount:** There are two choices here: manually or incorporated in the **/etc/fstab** file. Switch to *root* or use **sudo** for either operation.

- *Manual:* Use the **mount** command.

```
[student@desktopX ~]$ sudo mount -t nfs -o sync serverX:/share /mountpoint
```

The **-t nfs** option is the file system type for NFS shares (not strictly required, shown for completeness). The **-o sync** option tells **mount** to immediately synchronize write operations with the NFS server (the default is asynchronous). The default security method (**sec=sys**) will be used to try mounting the NFS share, using standard Linux file permissions.

- */etc/fstab:* Use **vim** to edit the **/etc/fstab** file and add the mount entry to the bottom of the file. The NFS share will be mounted at each system boot.

```
[student@desktopX ~]$ sudo vim /etc/fstab
...
serverX:/share /mountpoint nfs sync 0 0
```

Use **umount**, using **root** privileges, to manually unmount the share.

```
[student@desktopX ~]$ sudo umount /mountpoint
```



## References

**mount(8)**, **umount(8)**, **fstab(5)**, and **mount.nfs(8)** man pages

# Practice: Mounting and Unmounting NFS

## Guided exercise

In this lab, you will manually mount a Kerberos-secured NFS share, access it, and optionally unmount it. Create a persistent share mount in `/etc/fstab`, mount it, and access it. `serverX` is the NFSv4 host.

Resources:	
<b>Files:</b>	<code>nfs_ldapuserX.txt</code> and <code>nfs_student.txt</code>
<b>Machines:</b>	<code>desktopX</code> and <code>serverX</code>

### Outcomes:

- User `ldapuserX` will be able to successfully log in and access the persistently mounted NFS share `public` at `/mnt/public`.
- The NFS share `manual` can be mounted by users on an ad hoc basis at `/mnt/manual`.

### Before you begin...

- Reset the `serverX` system.
- Log into and set up your server system.

```
[student@serverX ~]$ lab nfsmount setup
```

- Reset the `desktopX` system.
- Log into and set up your desktop system.

```
[student@desktopX ~]$ lab nfsmount setup
```

- Open a terminal.



### Important

The `serverX` setup is used for both practice exercises in this chapter. It only needs to be run once.

S.H.I.E.L.D. (Storage Hardware Incorporating Every Last Document) uses a central server, `serverX`, to host a number of document share directories. Access to most directories is via LDAP-based users, authenticating using Kerberos; however, a number of shares are using standard Linux file access security. Users need to be able to log in and mount the `manual` NFS share, and should have the `public` NFS share available constantly.

Here are the key details you will need:

- Username: `ldapuserX`

- Password: **kerberos**
- serverX is sharing two directories under **/shares: manual** and **public**.
- desktopX mount point: **/mnt/public** and **/mnt/manual**
- The **public** NFS share requires **krb5p** authentication to access; **manual** is using **sys** security.
- The **krb5.keytab** is available from <http://classroom.example.com/pub/keytabs/desktopX.keytab>.
- Each share should have read and write access.

- ❑ 1. Download and install the **krb5.keytab** file to enable Kerberos access and security.

```
[student@desktopX ~]$ sudo wget -O /etc/krb5.keytab http://classroom.example.com/pub/keytabs/desktopX.keytab
```

- ❑ 2. Enable and start the **nfs-secure** service.

```
[student@desktopX ~]$ sudo systemctl enable nfs-secure
ln -s '/usr/lib/systemd/system/nfs-secure.service' ...
[student@desktopX ~]$ sudo systemctl start nfs-secure
```

- ❑ 3. Use **mkdir** to create both mount points: **/mnt/public** and **/mnt/manual**.

```
[student@desktopX ~]$ sudo mkdir -p /mnt/{public,manual}
```

- ❑ 4. Create the persistent mount. This mount will only be accessible to authenticated users.

- ❑ 4.1. Use **vim** to edit the **/etc/fstab** file.

```
[student@desktopX ~]$ sudo vim /etc/fstab
```

Add this line to the end of the file:

```
serverX:/shares/public /mnt/public nfs sec=krb5p,sync 0 0
```

- ❑ 4.2. Use **mount** to mount the share and begin using it.

```
[student@desktopX ~]$ sudo mount -a
```

- ❑ 5. Use **mount** to manually mount **/shares/manual** on **/mnt/manual**. Since you already have a kerberized NFSv4 mount from the same server you will need to specify the **sec=sys** option.

```
[student@desktopX ~]$ sudo mount -o sync,sec=sys serverX:/shares/manual /mnt/manual
```





## Note

When you are finished using the network storage, you can use the **umount** command to manually unmount the NFS shares.

```
[student@desktopX ~]$ sudo umount /mnt/manual
```

- ❑ 6. Use **ssh** to switch to **ldapuserX** on **localhost** and confirm the mounts, and read/write access.

- ❑ 6.1. Use **ssh** to log in as **ldapuserX**.

```
[student@desktopX ~]$ ssh ldapuserX@localhost
```

If you see something similar to the following, type **yes** to accept and continue.

```
The authenticity of host 'localhost (::1)' can't be established.  
ECDSA key fingerprint is d9:cc:73:82:3b:8a:74:e4:11:2f:f3:2b:03:a4:46:4d.  
Are you sure you want to continue connecting (yes/no)? yes
```

Enter the password: **kerberos**.

```
ldapuserX@localhost's password: kerberos
```

- ❑ 6.2. Verify you can switch to both share directories and confirm you have read/write access.

Use **cd** to switch directories.

```
[ldapuserX@desktopX ~]$ cd /mnt/manual
```

Use **echo** and **cat** to verify read and write access.

```
[ldapuserX@desktopX manual]$ echo hello > test.txt  
[ldapuserX@desktopX manual]$ cat test.txt  
hello
```

Repeat this step to test **/mnt/public**.

Use **exit** or **Ctrl+D** to log out of **ldapuserX**.

- ❑ 6.3. Repeat the previous step as **student** on both directories. You should be able to change directory and list **/mnt/manual**, but get **Permission denied** on **/mnt/public** because **student** cannot authenticate using Kerberos.

Instead of **test.txt**, you will want to use something like **test2.txt**, since **student** is not allowed to write to files owned by **ldapuserX**.

# Automounting Network Storage with NFS

## Objectives

After completing this section, students 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 can automatically mount 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/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 entirely on the client side; no server-side configuration required.
- The automounter uses the same mount options used by the **mount** command, including security options.
- Support for both direct and indirect mount point mapping, providing flexibility in mount point locations.
- Indirect mount points are created and removed by **autofs**, alleviating the need to manually manage them.
- NFS is the default file system for the automounter, but it can be used to automount a range of different file systems.
- **autofs** is a service that is managed like other system services.

### Create an automount

Configuring an automount is a multistep process:

1. Install the **autofs** package.

```
[student@desktopX ~]$ sudo yum -y 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.

Use **vim** to create and edit the master-map file:

```
[student@desktopX ~]$ sudo vim /etc/auto.master.d/demo.autofs
```

The name of the master-map file is not important, but it is normally something meaningful. The only requirement is it must have an extension of **.autofs**. The master-map file can hold multiple mapping entries, or use multiple files to separate configuration data.

Add the master-map entry, in this case, for indirectly mapped mounts:

```
/shares /etc/auto.demo
```

This entry would use the **/shares** directory as the base of future indirect automounts. The **/etc/auto.demo** file contains the mount details; use an absolute filename. The **auto.demo** file needs to be created before starting the **autofs** service.

To use directly mapped mount points, add an entry to the same file (or in a separate file):

```
/- /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**.

3. Create the mapping file(s). The mapping file identifies the mount point, mount options, and source location to mount.

Use **vim** to create and edit the mapping file:

```
[student@desktopX ~]$ sudo vim /etc/auto.demo
```

The file name is not important, but by convention is located in **/etc** and called **auto.name**, where *name* is something meaningful to the included contents.

```
work -rw, sync serverX:/shares/work
```

The format of an entry is *mount point*, *mount options*, and *source location*. This example is showing a basic indirect mapping entry. Direct maps and indirect maps using wildcards will be covered later in this section.

- Known as the "key" in the man pages, the *mount point* will be created and removed automatically by the **autofs** service. In this case, the fully qualified mount point will be **/shares/work**—see the master-map file. The **/shares** directory and the **work** directory will be created and removed as needed by the **autofs** service.

In this example, the local mount point mirrors the server's directory structure. The local mount point can be named anything. There is no requirement to align the names of the local mount point and the server directory structure.

- *Mount options* start with a **"-"** (dash) and are comma-separated with no white space. The mount options available are the same as those available to the equivalent manual mount

command. In this example, the automounter will try and mount the share using read/write access, security will be based on standard Linux file permissions (the default: `sec=sys`), and the server will be synchronized immediately during write operations.

There are a couple of useful automounter specific options: **-fstype=** and **-strict**. Use **fstype** to specify the file system if it is not NFS 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, `serverX:/shares/work`. This directory will need to have been *exported* on `serverX` with support for read/write access and standard Linux file permissions for the mount to be successful.

If the file system to be mounted begins with a `/` (slash), such as local device entries or SMB shares, then a `:` (colon) needs to be prefixed; for example, an SMB share would be `://serverX/share`.

#### 4. Start and enable the automount service.

Use **systemctl** to both start and enable the **autofs** service.

```
[student@desktopX ~]$ sudo systemctl enable autofs
ln -s '/usr/lib/systemd/system/autofs.service' ...
[student@desktopX ~]$ sudo systemctl start autofs
```

#### The mapping file—direct maps

As the name implies, direct maps are used to map an NFS share to an existing mount point. The automounter will not attempt to create the mount point automatically; it must exist prior to the **autofs** service being started.

Continuing from the previous example, the content for the `/etc/auto.direct` file might look like this:

```
/mnt/docs -rw,sync serverX:/shares/docs
```

The mount point (or key) is always an absolute path, starting with `/` (slash). The rest of the mapping file uses the same structure.

Only the right-most directory is put under automounter control. Thus, the directory structure above the mount point (`/mnt` in this example) is not obscured by **autofs**.

#### The mapping file—indirect wildcard maps

When an NFS server is exporting multiple subdirectories within a directory, then the automounter can be configured to access any one of those subdirectories using a single mapping entry. As an example, this can be really useful for automounting user *home* directories from an NFS server.

Continuing the previous example, if `serverX:/shares` is exporting two or more subdirectories and they are able to be accessed using the same mount options, then the content for the `/etc/auto.demo` file might look like this:

```
* -rw,sync serverX:/shares/&
```

The mount point (or key) is an "\*" (asterisk), and the subdirectory on the source location is an "&" (ampersand). Everything else in the entry is the same.

When a user attempts to access `/shares/work`, the key `*` (which is `work` in this example) will replace the ampersand in the source location and `serverX:/shares/work` will be mounted. As with the indirect example, the `work` directory will be created and removed automatically by the `autofs` service.



## References

`autofs(5)`, `automount(8)`, `auto.master(5)`, and `mount.nfs(8)` man pages

# Practice: Automounting NFS

## Guided exercise

In this lab, you will install a package to support automount. Create a direct-map automount and an indirect-map automount using wildcards. `serverX` is the NFSv4 host.

Resources:	
<b>Files:</b>	<code>nfs_ldapuserX.txt</code>
<b>Machines:</b>	<code>desktopX</code> and <code>serverX</code>

### Outcomes:

User `ldapuserX` will be able to successfully log in and use the three automounted directories.

### Before you begin...

- Reset the `desktopX` system.
- Log into and set up your desktop system.

```
[student@desktopX ~]$ lab nfsmount setup
```

- Open a terminal.



### Important

The `serverX` setup performed at the beginning of "*Mounting and Unmounting NFS*" is used for this practice exercise as well. If you have not yet performed the server setup, then run it now. It only needs to be run once for both practice exercises.

S.H.I.E.L.D. (Storage Hardware Incorporating Every Last Document) uses a central server, `serverX`, to host a number of document share directories. Access to these directories is via LDAP-based users, authenticating using Kerberos with encryption. Users need to be able to log in and have the share directories automount with read and write access, ready for use.

Here are the key details you will need:

- Username: `ldapuserX`
- Password: `kerberos`
- `serverX` is sharing three directories under `/shares`: `docs`, `work`, and `public`.
- File access is secured using Kerberos with encryption: `krb5p`.
- `desktopX` mount point: `/shares` for `docs` and `work` and a direct map of `public` to `/mnt/public`.
- The `krb5.keytab` is available from `http://classroom.example.com/pub/keytabs/desktopX.keytab`.
- Each share should have read and write access.

When done with the work, reboot the **desktopX** machine, then run the command **lab nfsmount grade** from the **desktopX** machine to verify the work.

- ❑ 1. Download and install the **krb5.keytab** file to enable Kerberos access and security.

```
[student@desktopX ~]$ sudo wget -O /etc/krb5.keytab http://classroom.example.com/pub/keytabs/desktopX.keytab
```

- ❑ 2. Enable and start the **nfs-secure** service.

```
[student@desktopX ~]$ sudo systemctl enable nfs-secure
ln -s '/usr/lib/systemd/system/nfs-secure.service' ...
[student@desktopX ~]$ sudo systemctl start nfs-secure
```

- ❑ 3. Use **yum** to install **autofs**, needed for automounting directories.

```
[student@desktopX ~]$ sudo yum -y install autofs
Loaded plugins: langpacks
Resolving Dependencies
...
Complete!
```

- ❑ 4. Create the automount configuration files for the *direct-map* automount.

- ❑ 4.1. Use **vim** to create and edit the **/etc/auto.master.d/direct.autofs** file.

```
[student@desktopX ~]$ sudo vim /etc/auto.master.d/direct.autofs
```

**Note:** The file extension must be **.autofs**.

Add the line as follows:

```
/- /etc/auto.direct
```

- ❑ 4.2. Use **vim** to create and edit the **auto.direct** map file.

```
[student@desktopX ~]$ sudo vim /etc/auto.direct
```

Add the line as follows:

```
/mnt/public -rw,sync,sec=krb5p serverX:/shares/public
```

**Note:** The file names above are not important; they were chosen to be meaningful.

- ❑ 5. Create the automount configuration files for the *indirect-map* automounts.

- ❑ 5.1. Use **vim** to create and edit the **/etc/auto.master.d/shares.autofs** file.

```
[student@desktopX ~]$ sudo vim /etc/auto.master.d/shares.autofs
```



**Note:** The file extension must be **.autofs**.

Add the line as follows:

```
/shares /etc/auto.shares
```

- ☐ 5.2. Use **vim** to create and edit the **auto.shares** map file.

```
[student@desktopX ~]$ sudo vim /etc/auto.shares
```

Add the line as follows:

```
* -rw, sync, sec=krb5p serverX:/shares/&
```

**Note:** The file names above are not important; they were chosen to be meaningful.

- ☐ 6. Use **mkdir** to create the **/mnt/public** mount point for the *direct-map* automount.

```
[student@desktopX ~]$ sudo mkdir -p /mnt/public
```

- ☐ 7. Enable and start the automount service.

```
[student@desktopX ~]$ sudo systemctl enable autofs
ln -s '/usr/lib/systemd/system/autofs.service' ...
[student@desktopX ~]$ sudo systemctl start autofs
```

- ☐ 8. Use **ssh** to switch to **ldapuserX** on **localhost** and confirm the mounts, and read/write access.

- ☐ 8.1. Use **ssh** to log in as **ldapuserX**.

```
[student@desktopX ~]$ ssh ldapuserX@localhost
```

If you see something similar to the following, type **yes** to accept and continue.

```
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is d9:cc:73:82:3b:8a:74:e4:11:2f:f3:2b:03:a4:46:4d.
Are you sure you want to continue connecting (yes/no)? yes
```

Enter the password: **kerberos**.

```
ldapuserX@localhost's password: kerberos
```

- ☐ 8.2. Verify you can switch to the automounted share directories and confirm you have read/write access.

Use **cd** to switch directories.

```
[ldapuserX@desktopX ~]$ cd /shares/docs
```

Use **echo** and **cat** to verify read and write access.

```
[ldapuserX@desktopX docs]$ echo hello > test.txt  
[ldapuserX@desktopX docs]$ cat test.txt  
hello
```

Repeat this step to test **/shares/work** and **/mnt/public**.

Use **exit** or **Ctrl+D** to log out of **ldapuserX**.

- 9. Reboot the **desktopX** machine, then run the command **lab nfsmount grade** from the **desktopX** machine to verify the work.

□ 9.1.

```
[student@desktopX ~]$ sudo systemctl reboot
```

□ 9.2.

```
[student@desktopX ~]$ lab nfsmount grade
```

# Accessing Network Storage with SMB

## Objectives

After completing this section, students should be able to:

- Mount and unmount SMB file systems using the command line.
- Automount SMB file systems.

## Manually mounting and unmounting SMB file systems

Many organizations need to provide network storage and print services for a range of desktop operating systems. Red Hat Enterprise Linux uses the Samba server to provide services that Microsoft Windows clients can use. Samba implements the Server Message Block (SMB) protocol, and Common Internet File System (CIFS) is a dialect of SMB. Often the two names are used interchangeably.

### Connecting to SMB/CIFS shares

Red Hat desktops and servers can connect to shares offered via *any* server that use the SMB protocol.

### Three Basic Steps for Accessing an SMB Share

1. *Identify* the remote share to access.
2. *Determine a mount point* where the share should be mounted and create the mount point's empty directory.
3. *Mount* the network file system with an appropriate command or configuration change.

Before starting, there is one package that must be installed in order to mount SMB shares: **cifs-utils**. Both the **mount** command and **autofs** automounter rely on this package for mounting CIFS file systems.

A second package, **samba-client**, has some useful command-line utilities—for example, **smbclient**—and is often worth installing as well.

### Mount SMB Share

1. **Identify:** The administrator for the SMB server host can provide share details, such as *username* and *password*, *share* names, etc. An alternative is to use a client that can browse the shares, such as **smbclient**.

```
[student@desktopX ~]$ smbclient -L //serverX
```

The **-L** option asks the **smbclient** to list the shares available on serverX.

2. **Mount point:** Use **mkdir** to create a mount point in a suitable location.

```
[student@desktopX ~]$ mkdir -p /mountpoint
```

3. **Mount:** There are two choices here: manually or incorporated in the **/etc/fstab** file. Switch to *root* or use **sudo** for either operation.

- *Manual:* Use the **mount** command.

```
[student@desktopX ~]$ sudo mount -t cifs -o guest //serverX/share /mountpoint
```

The **-t cifs** option is the file system type for SMB shares and the **-o guest** option tells **mount** to try and authenticate as a *guest* account without a password.

- */etc/fstab:* Use **vim** to edit the **/etc/fstab** file and add the mount entry to the bottom of the file. The SMB share will be mounted at each system boot.

```
[student@desktopX ~]$ sudo vim /etc/fstab
...
//serverX/share /mountpoint cifs guest 0 0
```

Use **umount**, using *root* privileges, to manually unmount the share.

```
[student@desktopX ~]$ sudo umount /mountpoint
```

### Authentication to SMB shares

SMB shares can be flagged as non-browsable, meaning clients such as **smbclient** will not display them. The SMB shares are still accessible by explicitly specifying the SMB share name during the mount operation.

SMB servers typically restrict access to specific users, or groups of users. Accessing protected shares will require appropriate credentials be presented to the SMB server. There are a range of authentication methods that an SMB server can choose to implement, too many to cover here.

A common choice for authentication is **username** and **password** pairs. These can either be added to the **mount** command (or **/etc/fstab** entry) or stored in a **credentials** file that is referenced during the mount operation. The **mount** command will prompt for a password if it is not provided, but it must be provided if using **/etc/fstab**. Guest access can be explicitly requested with the **guest** option.

*Some examples:*

```
[student@desktopX ~]$ sudo mount -t cifs -o guest //serverX/docs /public/docs
```

Mount the SMB share **//serverX/docs** at **/public/docs** and attempt to authenticate as *guest*.

```
[student@desktopX ~]$ sudo mount -t cifs -o username=watson //serverX/cases /bakerst/cases
```

Mount the SMB share **//serverX/cases** at **/bakerst/cases** and attempt to authenticate as *watson*. The **mount** command will prompt for the password in this example.

The **credentials** file offers better security because the password is stored in a more secure file, whereas the **/etc/fstab** file is easily examined.

```
[student@desktopX ~]$ sudo mount -t cifs -o credentials=/secure/sherlock //serverX/sherlock /home/sherlock/work
```

Mount the SMB share `//serverX/sherlock` at `/home/sherlock/work` and attempt to authenticate using the credentials stored in `/secure/sherlock`.

The format for the **credentials** file is:

```
username=username
password=password
domain=domain
```

It should be placed somewhere secure with only *root* access (for example, **chmod** 600).

During file operations, the SMB server will check file access against the credentials used to mount the share. The client will check file access against the UID/GID of the files sent from the server. This means that the client will need to have the same UID/GID, and if necessary, supplementary group membership as the files on the SMB server.

There are a number of mount options that handle local access checking and alternate authentication methods, such as `multiuser` (and `cifscreds`) and Kerberos-based options. They will not be covered here; for more information, refer to the **man** pages and articles available at `access.redhat.com`.

## Mounting SMB file systems with the automounter

Using the **mount** command requires *root* privileges to connect to the SMB shares. Alternatively, entries can be added to `/etc/fstab`, but then the connections to the SMB servers would be active all the time.

The automounter, or **autofs**, service can be configured to mount SMB shares "on demand" when a process attempts to access a file on the SMB share. The automounter will then unmount the share once it is no longer in use, after a certain period of inactivity.

The process for setting up an automount on a SMB share using **autofs** is essentially the same as other automounts:

- Add an **auto.master.d** configuration file that identifies the base directory for shares and the associated mapping file.
- Create or edit the mapping file to include the mount details for the SMB share.
- Enable and start the **autofs** service.



### Note

If it is not already installed, install the **autofs** package. Like **mount**, the automounter is also dependent on the **cifs-utils** package for mounting SMB shares.

### The mapping file

The file system type needs to be specified with the **-fstype=cifs** option and then a comma-separated list of mount options, the same mount options used by the **mount** command. The server URI address needs to be prefixed with a colon `:"`.

*An example:*

The following creates an automount at **/bakerst/cases** for SMB share **//serverX/cases**, and authenticates against the credentials file **/secure/sherlock**.

- **/etc/auto.master.d/bakerst.autofs** content:

```
/bakerst    /etc/auto.bakerst
```

- **/etc/auto.bakerst** content:

```
cases    -fstype=cifs,credentials=/secure/sherlock    ://serverX/cases
```

- **/secure/sherlock** content (owned by **root**, perms **600**):

```
username=sherlock  
password=violin221B  
domain=BAKERST
```

- **autofs** enable and start:

```
[student@desktopX ~]$ sudo systemctl enable autofs  
[student@desktopX ~]$ sudo systemctl start autofs
```



## References

**mount(8)**, **umount(8)**, **fstab(5)**, **mount.cifs(8)**, **smbclient(1)**, **autofs(5)**, **automount(8)**, and **auto.master(5)** man pages

# Practice: Mounting a SMB File System

## Guided exercise

In this lab, you will create a mount entry in **/etc/fstab** and mount it.

Resources:	
<b>Files:</b>	<b>samba.txt</b> in the server directory, for testing.
<b>Machines:</b>	<b>desktopX</b> and <b>serverX</b>

### Outcomes:

- **cifs-utils** package installed.
- The serverX student home folder mounted at **/home/student/work**.
- The **/etc/fstab** file includes the mount entry.

### Before you begin...

- Reset your serverX system.
- Log into and set up your server system.

```
[student@serverX ~]$ lab samba setup
```

- Reset your desktopX system.
- Log into desktopX and open a terminal.

You have a home directory on serverX that is used to store work-related documents. The directory is shared via Samba to support all of the company desktop operating systems.

The serverX administrator has confirmed that the share name is **student** and that the **uid/gid** are the same as your desktopX instance; the share password is *student*.

- ❑ 1. Install the Package  
Use **yum** to install **cifs-utils**.

```
[student@desktopX ~]$ sudo yum -y install cifs-utils
Loaded plugins: langpacks
Resolving Dependencies
...
Complete!
```

This package provides support for mounting CIFS file systems and is used by the **mount** command.

- ❑ 2. Create the Mount Point  
Use **mkdir** to create the **work** directory mount point.

```
[student@desktopX ~]$ mkdir ~/work
```

❑ 3. Create the Credentials File

- ❑ 3.1. Use **mkdir** to create the **secure** directory.

```
[student@desktopX ~]$ sudo mkdir /secure
```

- ❑ 3.2. Use **vim** to create the credentials file **student.smb** and populate it.

```
[student@desktopX ~]$ sudo vim /secure/student.smb
```

Add the following lines:

```
username=student
password=student
domain=MYGROUP
```

- ❑ 3.3. Use **chmod** to protect the **secure** directory and the **student.smb** credentials file.

```
[student@desktopX ~]$ sudo chmod 770 /secure
[student@desktopX ~]$ sudo chmod 600 /secure/student.smb
```

❑ 4. Update **/etc/fstab** and Mount

- ❑ 4.1. Use **vim** to add the mount settings to the end of **/etc/fstab**.

```
[student@desktopX ~]$ sudo vim /etc/fstab
...
//serverX/student /home/student/work cifs credentials=/secure/
student.smb 0 0
```

- ❑ 4.2. Use **mount** to verify the settings and mount the file system.

```
[student@desktopX ~]$ sudo mount -a
```

This command should report no errors. If it does, check your settings in **/etc/fstab**.

❑ 5. Check Your Access

- ❑ 5.1. Use **cat** to output the **samba.txt** file.

```
[student@desktopX ~]$ cat ~/work/samba.txt
Success
```

- ❑ 5.2. Use **echo** to write to the **work** mount point.

```
[student@desktopX ~]$ echo testing > ~/work/test.txt
```



# Lab: Accessing Network Storage with Network File System (NFS)

## Performance checklist

In this lab, you will install a package to support automount. Create an automount for `ldapuserX`'s "home" directory from `classroom.example.com`, an NFSv4 host.

### Resources:

<b>Machines:</b>	<b>desktopX</b> and <b>classroom.example.com</b>
------------------	--

### Outcomes:

User `ldapuserX` will be able to successfully log in and use the *home* directory mounted at `/home/guests/ldapuserX`.

### Before you begin...

- Reset the desktopX system.
- Log into and set up your desktop system.

```
[student@desktopX ~]$ lab nfs setup
```

- Open a terminal.

Umbrella Corp uses a central server, **classroom**, to host the *home* directories of their LDAP-based users. Users need to be able to log in and have their *home* directories automount with read and write access, ready for use.

Here are the key details you will need:

- Username: **ldapuserX**
- Password: **kerberos**
- **classroom.example.com** is sharing `/home/guests`.
- desktopX mount point: `/home/guests/ldapuserX`
- The *home* directory should have read and write access.

When done with the work, reboot the **desktopX** machine, then run the command **lab nfs grade** from the **desktopX** machine to verify the work.

1. Install any packages needed to automount the *home* directory.
2. Add an **auto.master.d** configuration file that identifies the base directory and associated map file (use any name desired for the configuration file, but it must end with **.autofs**), and create the associated map file (use any name desired for the map file).
3. Enable and start the automount service.

4. Use **ssh** to switch to **ldapuserX** on **localhost** and confirm the mount, and read/write access.
5. Reboot the **desktopX** machine, then run the command **lab nfs grade** from the **desktopX** machine to verify the work.

## Solution

In this lab, you will install a package to support automount. Create an automount for `ldapuserX`'s "home" directory from `classroom.example.com`, an NFSv4 host.

### Resources:

#### Machines:

**desktopX** and **classroom.example.com**

### Outcomes:

User **ldapuserX** will be able to successfully log in and use the *home* directory mounted at **/home/guests/ldapuserX**.

### Before you begin...

- Reset the **desktopX** system.
- Log into and set up your desktop system.

```
[student@desktopX ~]$ lab nfs setup
```

- Open a terminal.

Umbrella Corp uses a central server, **classroom**, to host the *home* directories of their LDAP-based users. Users need to be able to log in and have their *home* directories automount with read and write access, ready for use.

Here are the key details you will need:

- Username: **ldapuserX**
- Password: **kerberos**
- **classroom.example.com** is sharing **/home/guests**.
- **desktopX** mount point: **/home/guests/ldapuserX**
- The *home* directory should have read and write access.

When done with the work, reboot the **desktopX** machine, then run the command **lab nfs grade** from the **desktopX** machine to verify the work.

1. Install any packages needed to automount the *home* directory.

Use **yum** to install **autofs**.

```
[student@desktopX ~]$ sudo yum -y install autofs
Loaded plugins: langpacks
Resolving Dependencies
...
Complete!
```

2. Add an **auto.master.d** configuration file that identifies the base directory and associated map file (use any name desired for the configuration file, but it must end with **.autofs**), and create the associated map file (use any name desired for the map file).

- 2.1. Use **vim** to create and edit the **/etc/auto.master.d/home.autofs** file.

```
[student@desktopX ~]$ sudo vim /etc/auto.master.d/home.autofs
```

Add the line as follows:

```
/home/guests /etc/auto.home
```



### Note

This solution is using **home.autofs** as the master map file and **auto.home** as the map file, but the file names are not important.

- 2.2. Use **vim** to create and edit the **auto.home** map file.

```
[student@desktopX ~]$ sudo vim /etc/auto.home
```

Add the line as follows:

```
* -rw,sync classroom.example.com:/home/guests/&
```

3. Enable and start the automount service.

```
[student@desktopX ~]$ sudo systemctl enable autofs  
ln -s '/usr/lib/systemd/system/autofs.service' ...  
[student@desktopX ~]$ sudo systemctl start autofs
```

4. Use **ssh** to switch to **ldapuserX** on **localhost** and confirm the mount, and read/write access.

- 4.1. Use **ssh** to log in as **ldapuserX**.

```
[student@desktopX ~]$ ssh ldapuserX@localhost
```

If you see something similar to the following, type **yes** to accept and continue.

```
The authenticity of host 'localhost (::1)' can't be established.  
ECDSA key fingerprint is d9:cc:73:82:3b:8a:74:e4:11:2f:f3:2b:03:a4:46:4d.  
Are you sure you want to continue connecting (yes/no)? yes
```

Enter the password: **kerberos**.

```
ldapuserX@localhost's password: kerberos
```

- 4.2. Verify the current directory and read/write access.

Use **pwd** to verify the current directory.

```
[ldapuserX@desktopX ~]$ pwd
/home/guests/ldapuserX
```

Use **echo** and **cat** to verify read and write access.

```
[ldapuserX@desktopX ~]$ echo hello > test.txt
[ldapuserX@desktopX ~]$ cat test.txt
hello
```

Use **exit** or **Ctrl+D** to log out of **ldapuserX**.

5. Reboot the **desktopX** machine, then run the command **lab nfs grade** from the **desktopX** machine to verify the work.

5.1. 

```
[student@desktopX ~]$ sudo systemctl reboot
```

5.2. 

```
[student@desktopX ~]$ lab nfs grade
```

## Lab: Accessing Network Storage with SMB

### Performance checklist

In this lab, you will install packages to support automounting CIFS shares and create three automounts.

Resources:	
<b>Files:</b>	<b>samba.txt</b> in each share directory, for testing.
<b>Machines:</b>	<b>desktopX</b> and <b>serverX</b>

#### Outcomes:

- Installation of at least two packages to support automounting Samba shares.
- Automount **/shares/work** with authenticated, **RW** access to your home directory on serverX.
- Automount **/shares/docs** with **RO** guest access to the **public** share.
- Automount **/shares/cases** with authenticated, **RW** access to restricted team share **bakerst**.
- Available persistently after a *reboot*.

#### *Before you begin...*

If you haven't already done so at start of the previous exercise:

- Reset your serverX system.
- Log into and set up your server system.

```
[student@serverX ~]$ lab samba setup
```

Always perform this step:

- Reset your desktopX system.
- Log into desktopX and open a terminal.

Your company runs a Samba service on serverX to provide document sharing for both Red Hat Enterprise Linux and Microsoft Windows clients. The server contains a directory for each user to store their personal documents, a publicly available read-only directory for common documents, and a number of team directories to host collaborative documents.

You may need to perform some basic user and group administration on desktopX to ensure **student** can access files on all of the shares.

Here are the key details from serverX that you will need:

- Username: **student**
- Password: **student**

- Group membership: **bakerst**, **GID=10221**
- Domain: **MYGROUP**
- Home shares are enabled and writeable.  
desktopX mount point: **/shares/work**
- There is a share called **public** that only requires guest privileges to access.  
desktopX mount point: **/shares/docs**
- Your team has a private, writeable share called **bakerst** that is only accessible to members of the **bakerst** group.  
desktopX mount point: **/shares/cases**

When you are done, reboot your **desktopX** machine, then run the command **lab samba grade** from your **desktopX** machine to verify your work.

1. Install the two packages needed to automount a CIFS file system.
2. Add an **auto.master.d** configuration file that identifies the base directory and associated map file (use any name you like for the configuration file, but it must end with **.autofs**), and create the associated map file (use any name you like for the map file), ensuring proper authentication on each mount. As needed, you can create other configuration files to support the automount mapping configuration.
3. Ensure that username **student** has the correct UID and GIDs to access each of the shares (*Hint: **bakerst***). If necessary, add any new groups that are needed, modify student's group membership, or both.

**Note:** If you add a new group to student's supplementary groups, then you will either need to exit the shell and start a new shell, or use **newgrp groupname** to switch to the newly added group. This is necessary because the environment Bash starts with does not get updated with student's new details.

4. Enable and start the automount service.
5. Check that you can access each share and write to those shares you have write privileges on, **work** and **cases**.

There is a file called **samba.txt** that contains the message "Success" in each of the share locations. Use **cat samba.txt**.

Use **echo testing > my.txt** to test if you can write to a directory.

6. When you are done, reboot your **desktopX** machine, then run the command **lab samba grade** from your **desktopX** machine to verify your work.

## Solution

In this lab, you will install packages to support automounting CIFS shares and create three automounts.

Resources:	
<b>Files:</b>	<b>samba.txt</b> in each share directory, for testing.
<b>Machines:</b>	<b>desktopX</b> and <b>serverX</b>

### Outcomes:

- Installation of at least two packages to support automounting Samba shares.
- Automount **/shares/work** with authenticated, **RW** access to your home directory on serverX.
- Automount **/shares/docs** with **RO** guest access to the **public** share.
- Automount **/shares/cases** with authenticated, **RW** access to restricted team share **bakerst**.
- Available persistently after a *reboot*.

### Before you begin...

If you haven't already done so at start of the previous exercise:

- Reset your serverX system.
- Log into and set up your server system.

```
[student@serverX ~]$ lab samba setup
```

Always perform this step:

- Reset your desktopX system.
- Log into desktopX and open a terminal.

Your company runs a Samba service on serverX to provide document sharing for both Red Hat Enterprise Linux and Microsoft Windows clients. The server contains a directory for each user to store their personal documents, a publicly available read-only directory for common documents, and a number of team directories to host collaborative documents.

You may need to perform some basic user and group administration on desktopX to ensure **student** can access files on all of the shares.

Here are the key details from serverX that you will need:

- Username: **student**
- Password: **student**
- Group membership: **bakerst**, **GID=10221**
- Domain: **MYGROUP**



- Home shares are enabled and writeable.

desktopX mount point: **/shares/work**

- There is a share called **public** that only requires guest privileges to access.

desktopX mount point: **/shares/docs**

- Your team has a private, writeable share called **bakerst** that is only accessible to members of the **bakerst** group.

desktopX mount point: **/shares/cases**

When you are done, reboot your **desktopX** machine, then run the command **lab samba grade** from your **desktopX** machine to verify your work.

1. Install the two packages needed to automount a CIFS file system.

```
[student@desktopX ~]$ sudo yum -y install cifs-utils autofs
Loaded plugins: langpacks
Resolving Dependencies
...
Complete!
```

2. Add an **auto.master.d** configuration file that identifies the base directory and associated map file (use any name you like for the configuration file, but it must end with **.autofs**), and create the associated map file (use any name you like for the map file), ensuring proper authentication on each mount. As needed, you can create other configuration files to support the automount mapping configuration.

- 2.1. Use **vim** to create and edit the **/etc/auto.master.d/shares.autofs** file.

```
[student@desktopX ~]$ sudo vim /etc/auto.master.d/shares.autofs
```

Add the following line:

```
/shares /etc/auto.shares
```



## Note

This solution is using **shares.autofs** as the master map file and **auto.shares** as the map file, but the file names are not important.

- 2.2. Use **vim** to create the **auto.shares** map file.

```
[student@desktopX ~]$ sudo vim /etc/auto.shares
```

Add the following lines:

```
work -fstype=cifs,credentials=/etc/me.cred ://serverX/student
docs -fstype=cifs,guest ://serverX/public
```

```
cases -fstype=cifs,credentials=/etc/me.cred ://serverX/bakerst
```



## Note

An alternative to the credentials file (and the steps shown here to create and edit it) would be to substitute the **credentials=/etc/me.cred** entry in the **auto.shares** file with two entries, **username=student,password=student**, but that would be less secure.

- 2.3. Use **vim** to create the credentials file.

```
[student@desktopX ~]$ sudo vim /etc/me.cred
```

Add the following lines:

```
username=student
password=student
domain=MYGROUP
```

- 2.4. Use **chmod** to secure the credentials file.

```
[student@desktopX ~]$ sudo chmod 600 /etc/me.cred
```



## Note

This step is not essential for this lab, but shown for completeness.

3. Ensure that username **student** has the correct UID and GIDs to access each of the shares (*Hint: bakerst*). If necessary, add any new groups that are needed, modify student's group membership, or both.

**Note:** If you add a new group to student's supplementary groups, then you will either need to exit the shell and start a new shell, or use **newgrp groupname** to switch to the newly added group. This is necessary because the environment Bash starts with does not get updated with student's new details.

- 3.1. Use the **groups** command to check the student's group membership.

```
[student@desktopX ~]$ groups
student
```

The **student** account does not belong to the **bakerst** group (GID **10221**) and will need to be added.

- 3.2. Check if the **bakerst** group exists on desktopX. Use **grep** to check the **/etc/group** file.

```
[student@desktopX ~]$ grep -e bakerst -e 10221 /etc/group
```

The **bakerst** group does not exist either; it will need to be added first.

- 3.3. Use **groupadd** to add the **bakerst** group with GID **10221**.

```
[student@desktopX ~]$ sudo groupadd -g 10221 bakerst
```

- 3.4. Use **usermod** to add the **bakerst** group to **student** as a supplementary group.

```
[student@desktopX ~]$ sudo usermod -aG bakerst student
```



### Note

This approach is not typically the best solution to align UID and GID values, as there are mount options that can handle this. However, it is a suitable solution for this lab, and you get to practice some user and group administration skills.

- 3.5. Use **newgrp** to switch to **bakerst**.

```
[student@desktopX ~]$ newgrp bakerst
```

4. Enable and start the automount service.

```
[student@desktopX ~]$ sudo systemctl enable autofs  
ln -s '/usr/lib/systemd/system/autofs.service' ...  
[student@desktopX ~]$ sudo systemctl start autofs
```

5. Check that you can access each share and write to those shares you have write privileges on, **work** and **cases**.

There is a file called **samba.txt** that contains the message "Success" in each of the share locations. Use **cat samba.txt**.

Use **echo testing > my.txt** to test if you can write to a directory.

- 5.1. Check you can read and write in **work**:

```
[student@desktopX ~]$ cd /shares/work  
[student@desktopX work]$ cat samba.txt  
Success  
[student@desktopX work]$ echo testing > my.txt
```

- 5.2. Check you can read, but not write, in **docs**:

```
[student@desktopX work]$ cd ../docs  
[student@desktopX docs]$ cat samba.txt  
Success
```

```
[student@desktopX docs]$ echo testing > my.txt
bash: my.txt: Permission denied
```

5.3. Check you can read and write in **cases**:

```
[student@desktopX docs]$ cd ../cases
[student@desktopX cases]$ cat samba.txt
Success
[student@desktopX cases]$ echo testing > my.txt
```

6. When you are done, reboot your **desktopX** machine, then run the command **lab samba grade** from your **desktopX** machine to verify your work.

6.1. 

```
[student@desktopX ~]$ sudo systemctl reboot
```

6.2. 

```
[student@desktopX ~]$ lab samba grade
```

## Summary

### Mounting Network Storage with NFS

- Identify the NFS share details; NFSv4 mount the NFS server root folder.
- Create a mount point directory.
- **mount** or update **/etc/fstab** to mount the NFS share.
- **umount** to unmount a NFS share.

### Automounting Network Storage with NFS

- Install the necessary package: **autofs**.
- Create a master map file in **/etc/auto.master.d/file.autofs**.
- Create a map file for accessing the NFS share: **/etc/auto.name**.
  - Direct maps.
  - Indirect maps.
  - Indirect maps using wildcards.
- Start and enable the **autofs** service using **systemctl**.

### Accessing Network Storage with SMB

- Identify the share details; for example, **smbclient -L //server**.
- Create a mount point directory.
- **mount** or update **/etc/fstab** to mount the SMB share.
- **umount** to unmount a share.
- Use **autofs** service for automounting, using the same mount options as **mount**.
  - **enable** the service so it starts at boot.
  - **start** the service.
- Use **-fstype=cifs** and prefix the URI with a ":",





## CHAPTER 15

# FIREWALL CONFIGURATION

Overview	
<b>Goal</b>	To configure a basic firewall.
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Configure a basic firewall using <code>firewalld</code>, <code>firewall-config</code>, and <code>firewall-cmd</code>.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Limiting Network Communication (and Practice)</li></ul>
<b>Lab</b>	<ul style="list-style-type: none"><li>• Limiting Network Communication with <code>firewalld</code></li></ul>

# Limiting Network Communication

## Objectives

After completing this section, students should be able to configure a basic firewall.

## Netfilter and firewalld concepts

The Linux kernel includes a powerful network filtering subsystem, **netfilter**. The **netfilter** subsystem allows kernel modules to inspect every packet traversing the system. This means any incoming, outgoing, or forwarded network packet can be inspected, modified, dropped, or rejected in a programmatic way, before reaching components in user space. This is the main building block for building a firewall on a Red Hat Enterprise Linux 7 machine.

### Interacting with netfilter

Although it is theoretically possible for system administrators to write their own kernel modules to interact with **netfilter**, this is typically not done. Instead, other programs are used to interact with **netfilter**. One of the most common and well-known of these programs is **iptables**. In previous Red Hat Enterprise Linux releases, **iptables** was the main method of interacting with the kernel **netfilter** subsystem.

The **iptables** command is a low-level tool, and it can be challenging to correctly manage firewalls with that tool. In addition, it only adjusts IPv4 firewall rules. Other utilities, such as **ip6tables** for IPv6 and **ebtables** for software bridges, need to be used for more complete firewall coverage.

### Introducing firewalld

In Red Hat Enterprise Linux 7 a new method of interacting with **netfilter** has been introduced: **firewalld**. **firewalld** is a system daemon that can configure and monitor, the system firewall rules. Applications can talk to **firewalld** to request ports to be opened using the DBus messaging system, a feature which can be disabled or locked down). It both covers IPv4, IPv6, and potentially **ebtables** settings. The **firewalld** daemon is installed from the **firewalld** package. This package is part of a **base** install, but not part of a **minimal** install.

**firewalld** simplifies firewall management 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 then diverted into the firewall rules for the appropriate zone. Each zone can have its own list of ports and services to be opened 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 can be 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.



Every packet that comes into the system will first be checked for its *source address*. If that source address is tied to a specific zone, the rules for that zone will be parsed. If the source address is not tied to a zone, the zone for the *incoming* network interface will be used.

If the network interface is not associated with a zone for some reason, the *default* zone will be used. The default zone is not a separate zone itself; it is one of the other zones. The **public** zone is used by default, but this can be changed by a system administrator.

Most zones will allow traffic through the firewall which matches a list of particular ports and protocols ("631/udp") or pre-defined services ("ssh"). If the traffic does not match a permitted port/protocol or service, it will generally be rejected. (The **trusted** zone, which permits all traffic by default, is one exception to this.)

#### Pre-defined zones

**firewalld** ships with a number of pre-defined zones, suitable for various purposes. The default zone is set to *public* and interfaces are assigned to *public* if no changes are made. The **lo** interface is treated as if it were in the *trusted* zone. The following table details the configuration of these zones on installation, but the system administrator may then customize these zones to have different settings. By default, all zones permit any incoming traffic which is part of a communication initiated by the system, and all outgoing traffic.

#### Default configuration of firewalld zones

Zone name	Default configuration
<b>trusted</b>	Allow all incoming traffic.
<b>home</b>	Reject incoming traffic unless related to outgoing traffic or matching the <b>ssh</b> , <b>mdns</b> , <b>ipp-client</b> , <b>samba-client</b> , or <b>dhcpv6-client</b> pre-defined services.
<b>internal</b>	Reject incoming traffic unless related to outgoing traffic or matching the <b>ssh</b> , <b>mdns</b> , <b>ipp-client</b> , <b>samba-client</b> , or <b>dhcpv6-client</b> pre-defined services (same as the <b>home</b> zone to start with).
<b>work</b>	Reject incoming traffic unless related to outgoing traffic or matching the <b>ssh</b> , <b>ipp-client</b> , or <b>dhcpv6-client</b> pre-defined services.
<b>public</b>	Reject incoming traffic unless related to outgoing traffic or matching the <b>ssh</b> or <b>dhcpv6-client</b> pre-defined services. <i>The default zone for newly-added network interfaces.</i>
<b>external</b>	Reject incoming traffic unless related to outgoing traffic or matching the <b>ssh</b> pre-defined service. Outgoing IPv4 traffic forwarded through this zone is <i>masqueraded</i> to look like it originated from the IPv4 address of the outgoing network interface.
<b>dmz</b>	Reject incoming traffic unless related to outgoing traffic or matching the <b>ssh</b> pre-defined service.
<b>block</b>	Reject all incoming traffic unless related to outgoing traffic.
<b>drop</b>	Drop all incoming traffic unless related to outgoing traffic (do not even respond with ICMP errors).

For a list of all available pre-defined zones and their intended uses, consult the **firewalld.zones(5)** manual page.

### Pre-defined services

**firewalld** also ships with a number of pre-defined services. These service definitions can be used to easily permit traffic for particular network services to pass through the firewall. The following table details the configuration of the pre-defined services used in the default configuration of the firewall zones.

#### Selected pre-defined firewalld services

Service name	Configuration
<b>ssh</b>	Local SSH server. Traffic to 22/tcp
<b>dhcpv6-client</b>	Local DHCPv6 client. Traffic to 546/udp on the fe80::/64 IPv6 network
<b>ipp-client</b>	Local IPP printing. Traffic to 631/udp.
<b>samba-client</b>	Local Windows file and print sharing client. Traffic to 137/udp and 138/udp.
<b>mdns</b>	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 other pre-defined services exist. The **firewall-cmd --get-services** command will list them. The configuration files that define the ones included in the *firewalld* package can be found in the **/usr/lib/firewalld/services** directory, in a format defined by **firewalld.zone(5)**. We will not discuss these files further in this chapter.

For the purposes of this chapter, the easiest options for a system administrator new to **firewalld** is to either use pre-defined services or to explicitly specify the port/protocol they wish to permit. The **firewall-config** graphical tool can also be used to review pre-defined services and to define additional services.

## Configure firewall settings

There are three main ways for system administrators to interact with **firewalld**:

- By directly editing configuration files in **/etc/firewalld/** (not discussed in this chapter)
- By using the graphical **firewall-config** tool
- By using **firewall-cmd** from the command line

### Configure firewall settings with firewall-config

**firewall-config** is a graphical tool that can be used to alter and inspect both the running, in-memory configuration for **firewalld**, as well as the persistent, on-disk configuration. The **firewall-config** tool can be installed from the *firewall-config* package.

Once installed, **firewall-config** can be launched from the command line as **firewall-config**, or from the Applications menu under **Applications > Sundry > Firewall**. If **firewall-config** is started by an unprivileged user, it will prompt for the **root** password to continue.

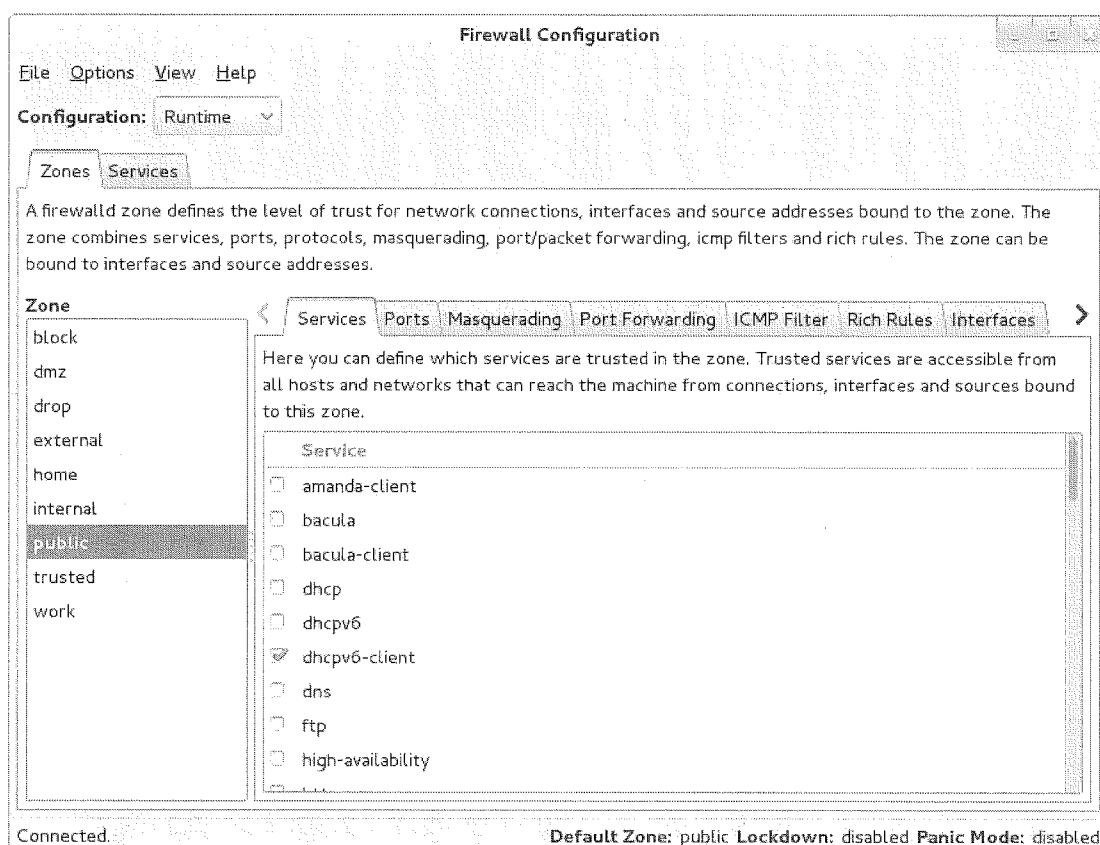


Figure 15.1: The main firewall configuration screen

On the main screen of **firewall-config**, a system administrator can select between modifying the current, in-memory configuration, or the persistent, on-disk configuration that will be used after a restart/reload of **firewalld**. This is achieved with the **Configuration** dropdown menu. In most cases, system administrators will want to adjust the persistent (**Permanent**) configuration, and then use the **Options > Reload Firewalld** menu entry to activate their changes.

To modify a zone, select the zone in the **Zone** menu on the left. Network interfaces and source IP addresses/ranges can be assigned in the **Interfaces** and **Sources** tabs on the right, respectively.

Ports can be opened by either putting a checkmark in front of them in the **Services** tab, or by adding a new port in the **Ports** tab for that zone.

If a specific set of ports has to be opened in multiple zones, a system administrator can also define a service for those ports. This can be done in the **Services** tab at the top of the window.

The *default* zone for otherwise unspecified connections can be changed under **Options > Change Default Zone**.



## Important

Any changes made in the **Permanent** configuration will not become active until the next time that the **firewalld** service unit is restarted or reloaded. Likewise, any changes made in the **Runtime** configuration will not survive a reload or restart of the **firewalld** service.

### Configure firewall settings with **firewall-cmd**

For those system administrators who prefer to work on the command line or who can not use a graphical environment for any reason, there is also a command-line client to interact with **firewalld**, **firewall-cmd**.

**firewall-cmd** is installed as part of the main **firewalld** package. **firewall-cmd** can perform the same actions that **firewall-config** can.

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. Many of the commands listed take the **--zone=<ZONE>** option to determine which zone they affect.

<b>firewall-cmd</b> commands	Explanation
<b>--get-default-zone</b>	Query the current default zone.
<b>--set-default-zone=&lt;ZONE&gt;</b>	Set the default zone. This changes both the runtime and the permanent configuration.
<b>--get-zones</b>	List all available zones.
<b>--get-active-zones</b>	List all zones currently in use (have an interface or source tied to them), along with their interface and source information.
<b>--add-source=&lt;CIDR&gt; [--zone=&lt;ZONE&gt;]</b>	Route all traffic coming from the IP address or network/netmask <b>&lt;CIDR&gt;</b> to the specified zone. If no <b>--zone=</b> option is provided, the default zone will be used.
<b>--remove-source=&lt;CIDR&gt; [--zone=&lt;ZONE&gt;]</b>	Remove the rule routing all traffic coming from the IP address or network/netmask <b>&lt;CIDR&gt;</b> from the specified zone. If no <b>--zone=</b> option is provided, the default zone will be used.
<b>--add-interface=&lt;INTERFACE&gt; [--zone=&lt;ZONE&gt;]</b>	Route all traffic coming from <b>&lt;INTERFACE&gt;</b> to the specified zone. If no <b>--zone=</b> option is provided, the default zone will be used.
<b>--change-interface=&lt;INTERFACE&gt; [--zone=&lt;ZONE&gt;]</b>	Associate the interface with <b>&lt;ZONE&gt;</b> instead of its current zone. If no

firewall-cmd commands	Explanation
	--zone= option is provided, the default zone will be used.
--list-all [--zone=<ZONE>]	List all configured interfaces, sources, services, and ports for <ZONE>. If no --zone= option is provided, the default zone will be used.
--list-all-zones	Retrieve all information for all zones. (Interfaces, sources, ports, services, etc.)
--add-service=<SERVICE>	Allow traffic to <SERVICE>. If no --zone= option is provided, the default zone will be used.
--add-port=<PORT/PROTOCOL>	Allow traffic to the <PORT/PROTOCOL> port(s). If no --zone= option is provided, the default zone will be used.
--remove-service=<SERVICE>	Remove <SERVICE> from the allowed list for the zone. If no --zone= option is provided, the default zone will be used.
--remove-port=<PORT/PROTOCOL>	Remove the <PORT/PROTOCOL> port(s) from the allowed list for the zone. If no --zone= option is provided, the default zone will be used.
--reload	Drop the runtime configuration and apply the persistent configuration.

#### firewall-cmd example

The following examples show the default zone being set to **dmz**, all traffic coming from the **192.168.0.0/24** network being assigned to the **internal** zone, and the network ports for **mysql** being opened on the **internal** zone.

```
[root@serverX ~]# firewall-cmd --set-default-zone=dmz
[root@serverX ~]# firewall-cmd --permanent --zone=internal --add-source=192.168.0.0/24
[root@serverX ~]# firewall-cmd --permanent --zone=internal --add-service=mysql
[root@serverX ~]# firewall-cmd --reload
```



## Note

For situations where the basic syntax of **firewalld** is not enough, system administrators can also add *rich-rules*, a more expressive syntax, to write more complex rules. If even the rich-rules syntax is not enough, system administrators can also use *Direct Configuration* rules, basically raw **iptables** syntax that will be mixed in with the **firewalld** rules.

These advanced modes are beyond the scope of this chapter.



## References

**firewall-cmd(1)**, **firewall-config(1)**, **firewalld(1)**, **firewalld.zone(5)** and **firewalld.zones(5)** man pages

# Practice: Limiting Network Communication

## Guided exercise

In this lab, you will configure a basic firewall.

Resources	
<b>Machines:</b>	<b>serverX</b> and <b>desktopX</b>

### Outcomes:

After completion of this exercise, your **serverX** machine should have a running web server, listening on both the cleartext port **80/TCP** and the SSL encapsulated port **443/TCP**. The firewall configuration on **serverX** should only allow connections to the SSL encapsulated port.

The firewall should allow access to **sshd** from all hosts.

### Before you begin...

- Reset your **serverX** system.

- ❑ 1. On your **serverX** system, make sure that both the **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.

❑ 1.1. 

```
[student@serverX ~]$ sudo yum -y install httpd mod_ssl
```

- ❑ 2. On your **serverX** system, create a new file called **/var/www/html/index.html**, with the following contents:

```
I am alive
```

❑ 2.1. 

```
[student@serverX ~]$ sudo bash -c "echo 'I am alive' > /var/www/html/index.html"
```

- ❑ 3. Start and enable the **httpd** service on your **serverX** system.

❑ 3.1. 

```
[student@serverX ~]$ sudo systemctl start httpd
```

❑ 3.2. 

```
[student@serverX ~]$ sudo systemctl enable httpd
```

- ❑ 4. On your **serverX** system, make sure that both the **iptables** and **ip6tables** services are *masked*, and that the **firewalld** service is enabled and running.

❑ 4.1. 

```
[student@serverX ~]$ sudo systemctl mask iptables
[student@serverX ~]$ sudo systemctl mask ip6tables
[student@serverX ~]$ sudo systemctl status firewalld
```

- ❑ 5. On your **serverX** system, start the **firewall-config** application. When prompted for the **root** password, enter **redhat**.

- ☐ 5.1. `[student@serverX ~]$ firewall-config`

or

Select **Applications > Sundry > Firewall** from the system menu.

- ☐ 6. From the **Configuration** dropdown menu, select **Permanent** to switch to editing the permanent configuration.
- ☐ 7. Add the **https** service to the list of services allowed in the **public** zone.
- ☐ 7.1. In the **Zone** list, select **public**. Since this zone is also the default zone, it is highlighted in bold.
- ☐ 7.2. In the **Services** tab, add a checkmark in front of the **https** service.
- ☐ 8. Activate your firewall configuration by selecting **Options > Reload Firewall** from the menu.
- ☐ 9. Verify your work by attempting to view your web server contents from **desktopX**.
- ☐ 9.1. This command should fail:

```
[student@desktopX ~]$ curl -k http://serverX.example.com
```

- ☐ 9.2. This command should succeed:

```
[student@desktopX ~]$ curl -k https://serverX.example.com
```



### 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.



# Lab: Limiting Network Communication

## Performance checklist

In this lab, you will configure a firewall on your **serverX** system to block all access to services other than **ssh** and a web server running on port **8080/TCP**.

Resources	
<b>Machines:</b>	<b>serverX</b> and <b>desktopX</b>

### Outcomes:

A firewall configured on **serverX** blocking access to services other than **ssh** and **8080/TCP**.

### Before you begin...

- Reset your **serverX** system.
- Log into and set up your **serverX** system.

```
[student@serverX ~]$ lab firewall setup
```

- Reset your **desktopX** system.

Your company has decided to run a new web app. This application listens on ports **80/TCP** and **8080/TCP**. Due to security considerations, only port **8080/TCP** should be reachable from the outside world. It is understood that **ssh** (port **22/TCP**) should also be available. All changes you make should persist across a reboot.

When you are done with your work, reboot your **serverX** machine, then run the command **lab firewall grade** from your **desktopX** machine to verify your work.

1. Configure your system so that the **iptables** and **ip6tables** services will not be accidentally started by an administrator.
2. Check if the **firewalld** service is running. If not, start it.
3. Verify that the default firewall zone is set to **public**.
4. Verify that there are no unwanted ports open in the permanent configuration for the **public** zone.
5. Add port **8080/TCP** to the permanent configuration for the **public** zone. Verify your configuration.
6. Reboot your **serverX** machine. (For a quick test, you can also use **sudo firewall-cmd --reload**.)
7. From your **desktopX** machine, run **lab firewall grade** to verify your work.

## Solution

In this lab, you will configure a firewall on your **serverX** system to block all access to services other than **ssh** and a web server running on port **8080/TCP**.

Resources	
<b>Machines:</b>	<b>serverX</b> and <b>desktopX</b>

### Outcomes:

A firewall configured on **serverX** blocking access to services other than **ssh** and **8080/TCP**.

### Before you begin...

- Reset your **serverX** system.
- Log into and set up your **serverX** system.

```
[student@serverX ~]$ lab firewall setup
```

- Reset your **desktopX** system.

Your company has decided to run a new web app. This application listens on ports **80/TCP** and **8080/TCP**. Due to security considerations, only port **8080/TCP** should be reachable from the outside world. It is understood that **ssh** (port **22/TCP**) should also be available. All changes you make should persist across a reboot.

When you are done with your work, reboot your **serverX** machine, then run the command **lab firewall grade** from your **desktopX** machine to verify your work.

1. Configure your system so that the **iptables** and **ip6tables** services will not be accidentally started by an administrator.

```
1.1. [student@serverX ~]$ sudo systemctl mask iptables
[student@serverX ~]$ sudo systemctl mask ip6tables
```

2. Check if the **firewalld** service is running. If not, start it.

```
2.1. [student@serverX ~]$ sudo systemctl status firewalld
```

- 2.2. If the previous step indicated that **firewalld** was not enabled and/or running:

```
[student@serverX ~]$ sudo systemctl enable firewalld
[student@serverX ~]$ sudo systemctl start firewalld
```

3. Verify that the default firewall zone is set to **public**.

```
3.1. [student@serverX ~]$ sudo firewall-cmd --get-default-zone
public
```

- 3.2. If the previous step returned another zone:

```
[student@serverX ~]$ sudo firewall-cmd --set-default-zone public
```

4. Verify that there are no unwanted ports open in the permanent configuration for the **public** zone.

4.1.

```
[student@serverX ~]$ sudo firewall-cmd --permanent --zone=public --list-all
public (default)
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

5. Add port **8080/TCP** to the permanent configuration for the **public** zone. Verify your configuration.

5.1.

```
[student@serverX ~]$ sudo firewall-cmd --permanent --zone=public --add-port
8080/tcp
```

5.2.

```
[student@serverX ~]$ sudo firewall-cmd --permanent --zone=public --list-all
public (default)
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports: 8080/tcp
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

6. Reboot your **serverX** machine. (For a quick test, you can also use **sudo firewall-cmd --reload**.)
7. From your **desktopX** machine, run **lab firewall grade** to verify your work.

7.1.

```
[student@desktopX ~]$ lab firewall grade
```

## Summary

### Limiting Network Communication

- The Linux kernel has a subsystem called **netfilter** to filter network traffic.
- **firewalld** is the user-space component that manages firewall rules.
- **firewalld** splits traffic into zones based on source address and the network interface it arrives on, with each zone having its own firewall rules.
- **firewall-config** and **firewall-cmd** can be used to control the firewall rules.



## CHAPTER 16

# VIRTUALIZATION AND KICKSTART

Overview	
<b>Goal</b>	To automate the installation of Red Hat Enterprise Linux on virtual machines with Kernel-based Virtual Machine (KVM) and <code>libvirt</code> .
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Explain Kickstart concepts and architecture.</li><li>• Create a Kickstart configuration file.</li><li>• Install a Red Hat Enterprise Linux system as a host for running virtual machines.</li></ul>
<b>Sections</b>	<ul style="list-style-type: none"><li>• Defining the Anaconda Kickstart System (and Practice)</li><li>• Deploying a New Virtual System with Kickstart (and Practice)</li><li>• Managing a Local Virtualization Host (and Practice)</li></ul>

# Defining the Anaconda Kickstart System

## Objectives

After completing this section, students should be able to identify key configuration elements found inside a Kickstart configuration file.

## Introduction to Kickstart installations

A system administrator can automate the installation of Red Hat Enterprise Linux using a feature called *Kickstart*. Anaconda, the Red Hat installer, needs to be told how to install a system: partition disks, configure network interfaces, select which packages to install, etc. This is an interactive process by default. A Kickstart installation uses a text file to provide all of the answers to these questions, so no interaction is required.



### Comparison

Kickstart in Red Hat Enterprise Linux is similar to Jumpstart for Oracle Solaris, or to an unattended installation for Microsoft Windows.

Kickstart configuration files begin with a list of commands that define how the target machine is to be installed. Lines that start with `#` characters are comments that are ignored by the installer. Additional sections begin with a line that starts with a `%` character and end with a line with the `%end` directive.

The `%packages` section specifies the software to be installed on the target system. Individual packages are specified by name (without versions). Package groups can be specified by name or ID, and start with an `@` character. Environment groups (groups of package groups) can be specified with the `@^` followed immediately by the name or ID of the environment group. Groups have mandatory, default, and optional components. Normally, mandatory and default components will be installed by Kickstart. Package or group names that are preceded with a `-` character are excluded from installation unless they are mandatory or installed due to RPM dependencies from other packages.

Two additional sections are the `%pre` and `%post` scripts. `%post` scripts are more common. They configure the system after all of the software has been installed. The `%pre` script is executed before any disk partitioning is done.

The configuration commands must be specified first. The `%pre`, `%post`, and `%packages` can occur in any order after the configuration commands.

## Kickstart configuration file commands

### Installation commands

- `url`: Specifies the location for the installation media.

Example:

```
url --url="ftp://installserver.example.com/pub/RHEL7/dvd"
```

- **repo**: This option tells Anaconda where to find the packages for installation. This option must point to a valid **yum** repository.

Example:

```
repo --name="Custom Packages" --baseurl="ftp://repo.example.com/custom"
```

- **text**: Forces text mode install.
- **vnc**: Allows the graphical installation to be viewed remotely via VNC.

Example:

```
vnc --password=redhat
```

- **askmethod**: Do not automatically use the CD-ROM as the source of packages when installation media is detected in the CD-ROM drive.

#### Partitioning commands

- **clearpart**: Clears the specified partitions before installation.

Example:

```
clearpart --all --drives=sda,sdb --initlabel
```

- **part**: Specifies the size, format, and name of a partition.

Example:

```
part /home --fstype=ext4 --label=homes --size=4096 --maxsize=8192 --grow
```

- **ignoredisk**: Ignores the specified disks when installing.

Example:

```
ignoredisk --drives=sdz
```

- **bootloader**: Defines where to install the bootloader.

Example:

```
bootloader --location=mbr --boot-drive=sda
```

- **volgroup, logvol**: Creates LVM volume groups and logical volumes.

Example:

```
part pv.01 --size=8192
volgroup myvg pv.01
logvol / --vgname=myvg --fstype=xfs --size=2048 --name=rootvol --grow
```

```
logvol /var --vgname=myvg --fstype=xfs --size=4096 --name=varvol
```

- **zerombr:** Disks whose formatting is unrecognized are initialized.

#### Network commands

- **network:** Configures network information for target system and activates network devices in installer environment.

Example:

```
network --device=eth0 --bootproto=dhcp
```

- **firewall:** This option defines how the firewall will be configured on the target system.

Example:

```
firewall --enabled --service=ssh,cups
```

#### Configuration commands

- **lang:** This required command sets the language to use during installation and the default language of the installed system.

Example:

```
lang en_US.UTF-8
```

- **keyboard:** This required command sets the system keyboard type.

Example:

```
keyboard --vckeymap=us --xlayouts='us','us'
```

- **timezone:** Defines timezone, NTP servers, and whether the hardware clock uses UTC.

Example:

```
timezone --utc --ntpservers=time.example.com Europe/Amsterdam
```

- **auth:** This required command sets up the authentication options for the system.

Example:

```
auth --useshadow --enablemd5 --passalgo=sha512
```

- **rootpw:** Defines the initial **root** password.

Example:

```
rootpw --plaintext redhat
```



or

```
rootpw --iscrypted $6$KUnFfrTz08jv.PiH$YlBb0tXBkwzoMuRfb0.SpbQ...XDR1UuchoMG1
```

- **selinux:** Sets the state of SELinux on the installed system.

Example:

```
selinux --enforcing
```

- **services:** Modifies the default set of services that will run under the default runlevel.

Example:

```
services --disabled=network, iptables, ip6tables --enabled=NetworkManager, firewallld
```

- **group, user:** Create a local group or user on the system.

Example:

```
group --name=admins --gid=10001
user --name=jdoe --gecos="John Doe" --groups=admins --password=changeme --plaintext
```

#### Miscellaneous commands

- **logging:** This command defines how Anaconda will log during the installation.

Example:

```
logging --host=loghost.example.com --level=info
```

- **firstboot:** Determines whether firstboot starts the first time the system is booted.

Example:

```
firstboot --disabled
```

- **reboot, poweroff, halt:** Specify what should happen after the installation finishes.



#### Note

The **ksverdiff** utility from the *pykickstart* package is useful for identifying changes in Kickstart file syntax between two versions of Red Hat Enterprise Linux or Fedora.

For example, **ksverdiff -f RHEL6 -t RHEL7** will identify changes in syntax from RHEL 6 to RHEL 7. Available versions are listed in the top of the file **/usr/lib/python2.7/site-packages/pykickstart/version.py**.

## Example Kickstart file:

```
#version=RHEL7
# System authorization information
auth --useshadow --enablemd5
# Use network installation
url --url="http://classroom.example.com/content/rhel7.0/x86_64/dvd/"
# Firewall configuration
firewall --enabled --service=ssh
firstboot --disable
ignoredisk --only-use=vda
# Keyboard layouts
keyboard --vckeymap=us --xlayouts='us','us'
# System language
lang en_US.UTF-8
# Installation logging level
logging --level=info
# Network information
network --bootproto=dhcp
# Root password
rootpw --iscrypted $6$/h/Mumvarr2dKrv1$Krv7h9.QoV0s....foMXsGXP1KllaiJ/w7EWiL1
# SELinux configuration
selinux --enforcing
# System services
services --disabled="kdump,rhsmcertd" --enabled="network,sshd,rsyslog,chronyd"
# System timezone
timezone --utc America/Los_Angeles
# System bootloader configuration
bootloader --location=mbr --boot-drive=vda
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all --initlabel
# Disk partitioning information
part / --fstype="xfs" --ondisk=vda --size=10000

%packages
@core
chrony
cloud-init
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
-NetworkManager
-plymouth

%end

%post --erroronfail

# For cloud images, 'eth0' _is_ the predictable device name, since
# we don't want to be tied to specific virtual (!) hardware
rm -f /etc/udev/rules.d/70*
ln -s /dev/null /etc/udev/rules.d/80-net-name-slot.rules

# simple eth0 config, again not hard-coded to the build hardware
cat > /etc/sysconfig/network-scripts/ifcfg-eth0 << EOF
DEVICE="eth0"
BOOTPROTO="dhcp"
```

```
ONBOOT="yes"
TYPE="Ethernet"
USERCTL="yes"
PEERDNS="yes"
IPV6INIT="no"
EOF

%end
```



## Note

In a Kickstart file, missing required values cause the installer to interactively prompt for an answer or to abort the installation entirely.



## References

**ksverdiff(1)** man page

The file **/usr/share/doc/pykickstart-\*/kickstart-docs.txt** provided by the *pykickstart* package contains useful and detailed information on the syntax of Kickstart files.

Additional information may be available in the *Red Hat Enterprise Linux Installation Guide* for RHEL 7 located at:

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

# Practice: Kickstart File Syntax and Modification

## Quiz

Match the following Kickstart commands with their descriptions in the table.

<b>%packages</b>	<b>%post</b>	<b>auth</b>	<b>clearpart</b>	<b>network</b>
<b>part</b>	<b>rootpw</b>	<b>services</b>	<b>timezone</b>	<b>url</b>

Description	Command
Section of a Kickstart configuration file that specifies what software is installed on the new system.	
Required Kickstart command that configures how users access the system.	
Location of the software used by Kickstart to install a system.	
Scripting in a Kickstart configuration file that is executed after the software is installed on a system.	
Kickstart command that specifies which partitions should be cleared before installation.	
Modifies which services will start by default at system boot.	

Description	Command
Defines the default authentication credentials for the superuser.	
Kickstart command that specifies the size, format, and name of a disk partition.	
Kickstart command used to specify NTP servers.	
Determines the network configuration for the installation and the target system.	

## Solution

Match the following Kickstart commands with their descriptions in the table.

Description	Command
Section of a Kickstart configuration file that specifies what software is installed on the new system.	<b>%packages</b>
Required Kickstart command that configures how users access the system.	<b>auth</b>
Location of the software used by Kickstart to install a system.	<b>url</b>
Scripting in a Kickstart configuration file that is executed after the software is installed on a system.	<b>%post</b>
Kickstart command that specifies which partitions should be cleared before installation.	<b>clearpart</b>
Modifies which services will start by default at system boot.	<b>services</b>
Defines the default authentication credentials for the superuser.	<b>rootpw</b>
Kickstart command that specifies the size, format, and name of a disk partition.	<b>part</b>
Kickstart command used to specify NTP servers.	<b>timezone</b>

Description	Command
Determines the network configuration for the installation and the target system.	<b>network</b>

# Deploying a New Virtual System with Kickstart

## Objectives

After completing this section, students should be able to:

- Create a Kickstart configuration file with the **system-config-kickstart** utility.
- Modify an existing Kickstart configuration file with a text editor and check its syntax with **ksvalidator**.
- Publish a Kickstart configuration file to the installer.
- Perform a network Kickstart installation.

## Kickstart installation steps

An ordered process is required to automate the successful installation of Red Hat Enterprise Linux.

Three steps must be taken to perform a Kickstart installation:

1. Create a Kickstart configuration file.
2. Publish the Kickstart configuration file to the installer.
3. Boot Anaconda and point it to the Kickstart configuration file.

## Creating a Kickstart configuration file

There are two ways to create a Kickstart configuration file:

- Use the **system-config-kickstart** utility.
- Use a text editor.

The **system-config-kickstart** utility presents a number of graphical dialog boxes, takes inputs from the user, then creates a text file with Kickstart directives that correspond to the user's choices. Each dialog box corresponds to a category of questions asked by the Red Hat installer, Anaconda. Optionally, an existing configuration file can be passed as an argument and **system-config-kickstart** will use it to populate values for configuration options. **system-config-kickstart** is provided by the *system-config-kickstart* package.



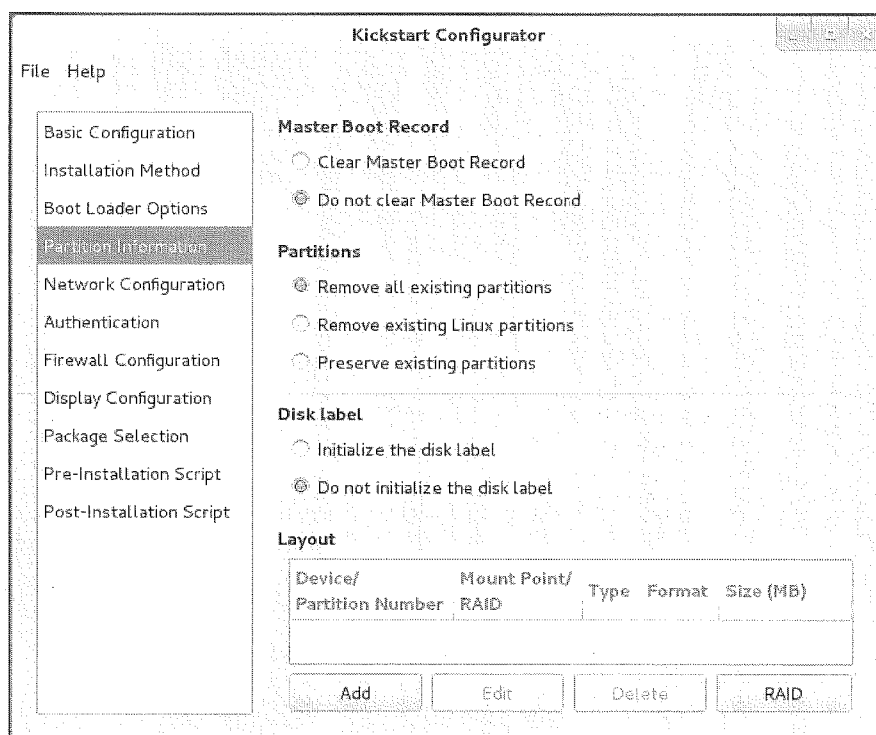


Figure 16.1: Configuring storage with `system-config-kickstart`

Creating a Kickstart configuration file from scratch with a text editor is rare. The Anaconda installer creates a file called `/root/anaconda-ks.cfg` that contains the Kickstart directives that can be used to generate the freshly installed system. This file makes a good starting point when creating a Kickstart configuration file with a text editor.

The following are some reasons for creating a Kickstart file manually instead of using **system-config-kickstart**:

1. The GUI and/or **system-config-kickstart** is unavailable.
2. Advanced disk partition configuration instructions are needed. **system-config-kickstart** does not support LVM and software RAID.
3. Individual packages need to be included or omitted (not just groups).
4. More advanced scripting is needed in the `%pre` and `%post` sections.

**ksvalidator** is a utility that checks for syntax errors in a Kickstart configuration file. It will ensure keywords and options are properly used, but it will not validate URL paths, individual packages, or groups, nor any part of `%post` or `%pre` scripts. For instance, if the **firewall --disabled** directive is misspelled, **ksvalidator** could produce one of the following errors:

```
[student@desktopX]$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:

Unknown command: frewall

[student@desktopX]$ ksvalidator /tmp/anaconda-ks.cfg
The following problem occurred on line 10 of the kickstart file:
```

```
no such option: --dsabled
```

The *pykickstart* RPM provides **ksvalidator**.

## Publish the Kickstart configuration file to Anaconda

Make the Kickstart configuration file available to the installer:

- Network servers: FTP, HTTP, NFS
- DHCP/TFTP server
- USB disk or CD-ROM
- Local hard disk

The installer must be able to access the Kickstart file to begin an automated installation. Although there are several methods to make the Kickstart configuration file available, the most common is through a network server such as an FTP server, a web server, or an NFS server. Network servers facilitate Kickstart file maintenance because changes only need to be made once and take effect immediately.

Providing Kickstart files on USB or CD-ROM is another convenient way to publish configuration files. The Kickstart configuration file is embedded on the boot media used to start the installation. When changes are made, new installation media must be generated.

It is possible to provide the Kickstart file on a local disk. This allows a quick way to rebuild a development server.

## Boot Anaconda and point it to the Kickstart configuration file

Once a Kickstart method is chosen, the installer must be told where the Kickstart file is located. This is done by passing a **ks=LOCATION** argument to the installation kernel. The following are some sample specifications:

- `ks=http://server/dir/file`
- `ks=ftp://server/dir/file`
- `ks=nfs:server:/dir/file`
- `ks=hd:device:/dir/file`
- `ks=cdrom:/dir/file`

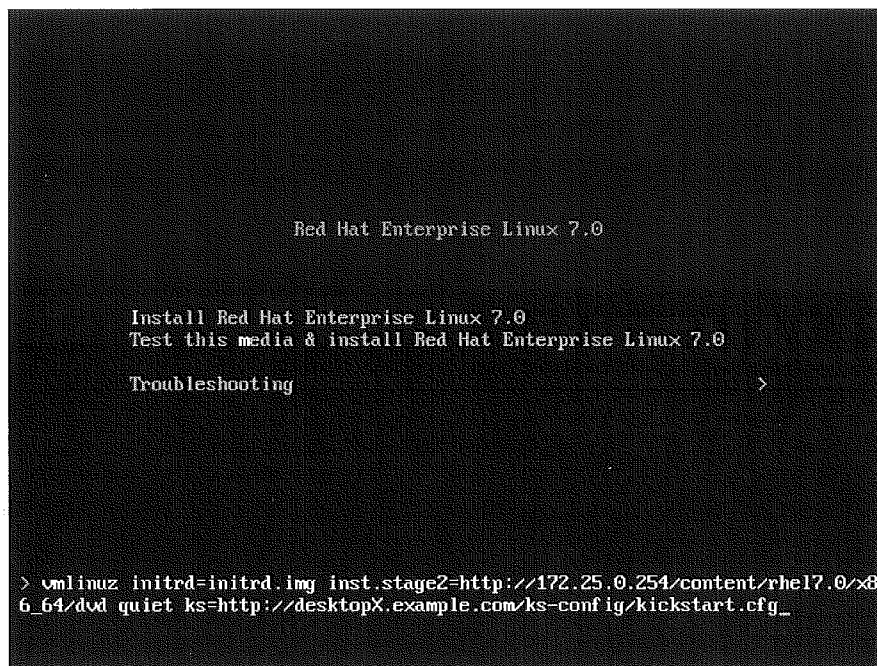


Figure 16.2: Specifying the Kickstart file location during PXE boot

For virtual machine installations using the Virtual Machine Manager or **virt-manager**, the Kickstart URL can be specified in a box under **URL Options**. When installing physical machines, boot using installation media and press the **Tab** key to interrupt the boot process. Enter one of the **ks=** entries above as a parameter to the installation kernel.



## References

**ksvalidator**(1) and **system-config-kickstart**(8) man pages

## Practice: Installing a System Using Kickstart

### Guided exercise

In this lab, you will create a Kickstart configuration file, confirm it is syntactically correct, and publish it for use.

Resources	
<b>Files:</b>	/root/anaconda-ks.cfg
<b>Machines:</b>	desktopX and serverX

#### Outcomes

You will have a Kickstart configuration file based on the **anaconda-ks.cfg** file on **desktopX**. It will install packages from **classroom.example.com**, use DHCP for networking, partition storage and install packages according to specifications, and perform minor customization of the newly installed system. Additionally, you will go through the process of using your Kickstart configuration file to reinstall **serverX**.

#### Before you begin...

- Reset your **desktopX** system.
- Log into and set up your **desktopX** system.

```
[student@desktopX ~]$ lab kickstart setup
```

1. Copy **/root/anaconda-ks.cfg** on **desktopX** to a file called **kickstart.cfg** that **student** can edit.

```
[student@desktopX ~]$ sudo cat /root/anaconda-ks.cfg > kickstart.cfg
```

2. Make the following changes to **kickstart.cfg**.

- 2.1. Change the **url** command to specify the HTTP installation source media used in the classroom:

```
url --url="http://classroom.example.com/content/rhel7.0/x86_64/dvd/"
```

- 2.2. Configure the network to use DHCP. There should only be a single **network** directive that looks like the following:

```
network --bootproto=dhcp
```

- 2.3. Modify the disk configuration to only have the following three directives:

```
# Clear the Master Boot Record
zerombr
# Partition clearing information
clearpart --all --initlabel
# Disk partitioning information
```

```
part / --fstype="xfs" --ondisk=vda --size=5120
```

Be sure the size is adjusted to 5120.

- ❑ 2.4. Comment the **reboot** directive:

```
#reboot
```

- ❑ 2.5. Change the packages that are installed to include **httpd**, but not **cloud-init**. Simplify the package specification to look exactly like the following:

```
@core
chrony
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
httpd
-plymouth
```

- ❑ 2.6. Delete all of the content in the **%post** section except for the following lines:

```
%post --erroronfail
# make sure firstboot doesn't start
echo "RUN_FIRSTBOOT=NO" > /etc/sysconfig/firstboot
# append /etc/issue with a custom message
echo "Kickstarted for class on $(date)" >> /etc/issue
%end
```

- ❑ 2.7. Set the **root** password to **redhat**. Change the line that starts with **rootpw** to:

```
rootpw --plaintext redhat
```

- ❑ 3. Use the **ksvalidator** command to check the Kickstart file for syntax errors.

```
[student@desktopX ~]$ ksvalidator kickstart.cfg
```

- ❑ 4. Copy **kickstart.cfg** to the **/var/www/html/ks-config** directory.

```
[student@desktopX ~]$ sudo cp ~student/kickstart.cfg /var/www/html/ks-config
```

- ❑ 5. Run the **lab kickstart** grading script on **desktopX** to confirm the specified changes have been made and the kickstart file is available via HTTP.

```
[root@desktopX ~]# lab kickstart grade
Kickstart file available via HTTP ..... PASS
Confirming installation media ..... PASS
Checking installed disk size ..... PASS
Confirming network configuration ..... PASS
```

```
Checking software package selection ... PASS
```

- ❑ 6. PXE-boot the **serverX** virtual machine and initiate a Kickstart installation.

- ❑ 6.1. Boot the **serverX** virtual machine. Quickly, during the boot sequence, it will display the following message:

```
Press F12 for boot menu.
```

Press **F12** to get to the boot menu.

- ❑ 6.2. You should see a menu similar to the following:

```
Select boot device:
1. Virtio disk PCI:0:4
2. Virtio disk PCI:0:5
3. Legacy option rom
4. iPXE (PCI 00:03.0)
```

Select the number that selects the iPXE device.

- ❑ 6.3. Use the arrow keys to highlight the line that reads, **Install Red Hat Enterprise Linux 7.0**. Press the **Tab** key to see the full configuration options. Add the **ks=http://desktopX.example.com/ks-config/kickstart.cfg** directive to the end of the line, then press **Enter**.

```
.../dvd quiet ks=http://desktopX.example.com/ks-config/kickstart.cfg
```

It takes a couple minutes for the installation to begin. If it aborts or prompts for input, make corrections to your Kickstart configuration file, publish it, then try again.

- ❑ 7. Watch the installation and wait for it to complete.

- ❑ 7.1. Once the installation gets started, you should see a graphical screen appear as Anaconda formats the hard drive, then installs packages.
  - ❑ 7.2. The installer will pause once the installation is finished. Click the **Reboot** button to continue. **serverX** should display the following when it finishes booting:

```
Red Hat Enterprise Linux Server 7.0 (Maipo)
Kernel 3.10.0-84.el7.x86_64 on an x86_64

Kickstarted for class on Fri Feb 28 20:08:22 EST 2014
serverX login:
```

- ❑ 8. Log into **serverX** as **root**, download, and run the grading script.

```
[root@serverX ~]# curl http://classroom.example.com/pub/materials/lab-kickstart -o lab-kickstart
[root@serverX ~]# chmod 755 lab-kickstart
[root@serverX ~]# ./lab-kickstart grade
```

```
Confirming installation media ..... PASS
Checking installed disk size ..... PASS
Confirming network configuration ..... PASS
Checking software package selection ... PASS
Checking effects of kickstart %post ... PASS
```

## Managing a Local Virtualization Host

### Objectives

After completing this section, students should be able to:

- Describe and contrast Red Hat virtualization platforms.
- Install Red Hat Enterprise Linux as a virtualization host system.

### System virtualization and Red Hat Enterprise Linux

KVM (Kernel-based Virtual Machine) is a full virtualization solution built into the standard Red Hat Enterprise Linux kernel. It can run multiple, unmodified Windows and Linux guest operating systems. The KVM hypervisor in Red Hat Enterprise Linux is managed with the *libvirt* API and utilities, such as **virt-manager** and **virsh**. Since Red Hat Enterprise Linux is the foundation of Red Hat Enterprise Virtualization and the Red Hat OpenStack platform, KVM is a consistent component across products of the Red Hat cloud infrastructure.

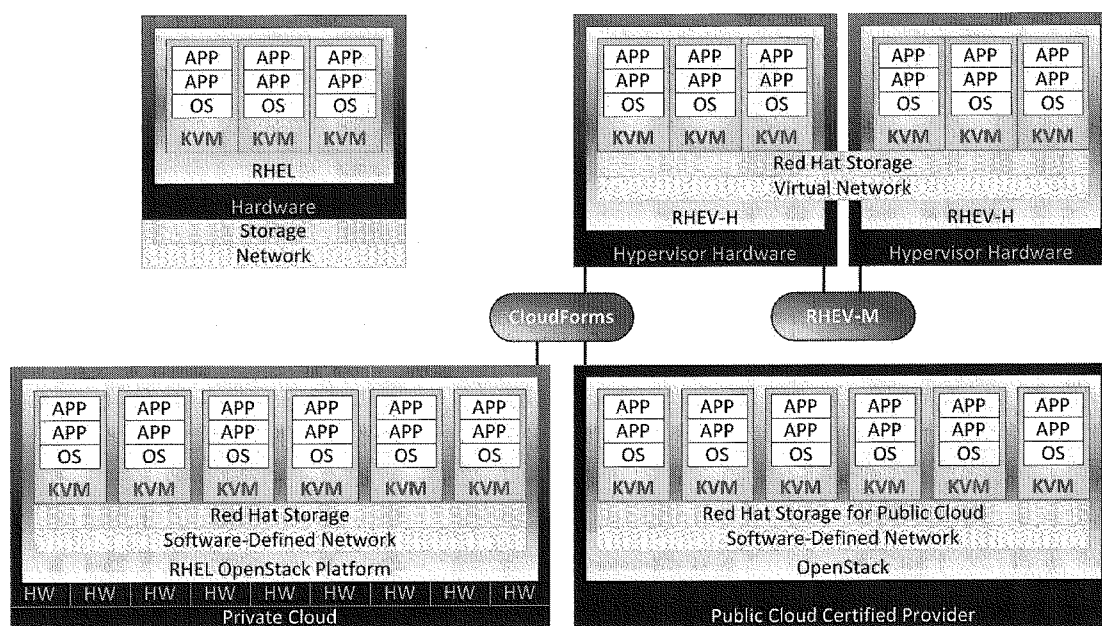


Figure 16.3: KVM throughout the Red Hat cloud infrastructure

KVM provides the virtual machine (VM) technology across all Red Hat products, from standalone physical instances of Red Hat Enterprise Linux up to the cloud OpenStack platform. Starting at the upper-left corner of the previous figure:

- **Physical (legacy) systems**—Red Hat Enterprise Linux installations on legacy hardware provide KVM virtualization, up to the physical limitations of single systems, and are managed by libvirt utilities such as **virt-manager**. Red Hat Enterprise Linux instances may also be directly hosted at Red Hat Certified Cloud Providers through Red Hat Cloud Access.

Red Hat Enterprise Linux is typically configured as a *thick host*, a system that supports VMs while also providing other local and network services, applications, and management functions.



- **Red Hat Enterprise Virtualization (RHEV)**—supports KVM instances across multiple Red Hat Enterprise Virtualization hypervisor (RHEV-H) systems, providing KVM migration, redundancy, and high availability managed by RHEV Manager (RHEV-M).

Red Hat Enterprise Virtualization Hypervisor is a *thin host*, an expertly minimized and tuned version of Red Hat Enterprise Linux dedicated to the singular purpose of provisioning and supporting guest VMs.

- **RHEL OpenStack platform**—Red Hat private cloud architecture using integrated and tuned OpenStack on a Red Hat Enterprise Linux foundation with KVM, managed by either the Red Hat OpenStack Dashboard (Horizon component) or by Red Hat CloudForms.
- **OpenStack in public cloud**—OpenStack public cloud architecture implemented at Red Hat Certified Cloud Providers, and managed by either the OpenStack Horizon component or by Red Hat CloudForms.
- **Hybrid Cloud**—Red Hat CloudForms cloud management utilities manage and migrate KVM instances across Red Hat RHEV and OpenStack architectures, and transition KVM instances with third-party OpenStack and VMware platforms.

KVM instance configurations are compatible across Red Hat products. Installation requirements, parameters, and procedures are equivalent on supported platforms.

#### Configure a Red Hat Enterprise Linux Physical System as a Virtualization Host

Red Hat Enterprise Linux can be configured as a virtualization host, appropriate for development, testing, training, or when needing to work in multiple operating systems at the same time. Red Hat Enterprise Linux hosts provide the ability to install additional software on the host platform as needed, such as monitoring utilities and agents, network services, specialized storage, and/or other development tools that may not be appropriate to install on dedicated Red Hat Enterprise Virtualization hypervisors.

Red Hat Enterprise Linux installations also provide easier access to tuning and resource management tools (e.g., **tuned** and **cgroups**). By comparison, RHEV-H hypervisors are highly secured and self-tuned, restricting system administrator-initiated customization by design. When greater administrative control is required and the performance compromise is acceptable, Red Hat Enterprise Linux is a flexible standalone KVM platform. KVM instances built on RHEL can be migrated or transitioned to more appropriate KVM platforms as enterprise needs increase.

Preparing a Red Hat Enterprise Linux system to become a virtualization host requires checking for minimal system requirements and installing a choice of virtualization host packages.

Recommended system requirements:

- One processor core or hyperthread for the maximum number of virtualized CPUs in a guest virtual machine and one for the host.
- 2GB of RAM, plus additional RAM for virtual machines.
- 6GB disk space for the host, plus the required disk space for each virtual machine. Most guest operating systems require at least 6GB of disk space, but actual storage space requirements depend on each guest's image format.

The KVM hypervisor requires either an Intel processor with the Intel VT-x and Intel 64 extensions for x86-based systems, or an AMD processor with the AMD-V and the AMD64 extensions. To verify that the host system hardware supports the correct extensions, view **/proc/cpuinfo**.

```
[root@serverX ~]# grep --color -E "vmx|svm" /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc
arch_perfmon pebs bts rep_good aperfmperf pni dtes64 monitor ds_cpl vmx smx est
tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dts tpr_shadow vnmi flexpriority
```

The No eXecute (NX) feature, called eXecute Disable (XD) by Intel and Enhanced Virus Protection by AMD, is not necessary for building a host on Red Hat Enterprise Linux, but is required for a Red Hat Enterprise Virtualization hypervisor (RHEV-H).

```
[root@serverX ~]# grep --color -E "nx" /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_tsc
arch_perfmon pebs bts rep_good aperfmperf pni dtes64 monitor ds_cpl vmx smx est
tm2 ssse3 cx16 xtpr pdcm sse4_1 xsave lahf_lm dts tpr_shadow vnmi flexpriority
```

Building a RHEL virtualization host requires the **qemu-kvm** and **qemu-img** packages at minimum, to provide the user-level KVM emulator and disk image manager.

```
[root@serverX ~]# yum install qemu-kvm qemu-img
```

Additional virtualization management packages are also recommended:

- **python-virtinst**—Provides the `virt-install` command for creating virtual machines.
- **libvirt**—Provides the host and server libraries for interacting with hypervisors and host systems.
- **libvirt-python**—Contains a module to permit Python applications to use the libvirt API.
- **virt-manager**—Provides the Virtual Machine Manager graphical tool for administering VMs, using the `libvirt-client` library as the management API.
- **libvirt-client**—Provides the client APIs and libraries for accessing libvirt servers, including the `virsh` command-line tool to manage and control VMs.

```
[root@serverX ~]# yum install virt-manager libvirt libvirt-python python-virtinst
libvirt-client
```

The updated **anaconda** graphical installation program for Red Hat Enterprise Linux 7 provides better support for installing RHEL to meet specific purposes. An **anaconda** install no longer provides the ability to select individual RPM packages—only base environments and add-ons appropriate for the selected base—thus eliminating guesswork and resulting in leaner configurations. System administrators may still install any other desired RPM packages after an installation is complete by using standard RPM installation tools (e.g., **yum** or GNOME PackageKit).

To build a virtualization host during a graphical install of Red Hat Enterprise Linux, choose the base environment **Virtualization Host** in the left pane of the **anaconda Software Selection** screen. Select the **Virtualization Platform** Add-On checkbox in the right pane to include the management tools and utilities, as shown in the following figure.

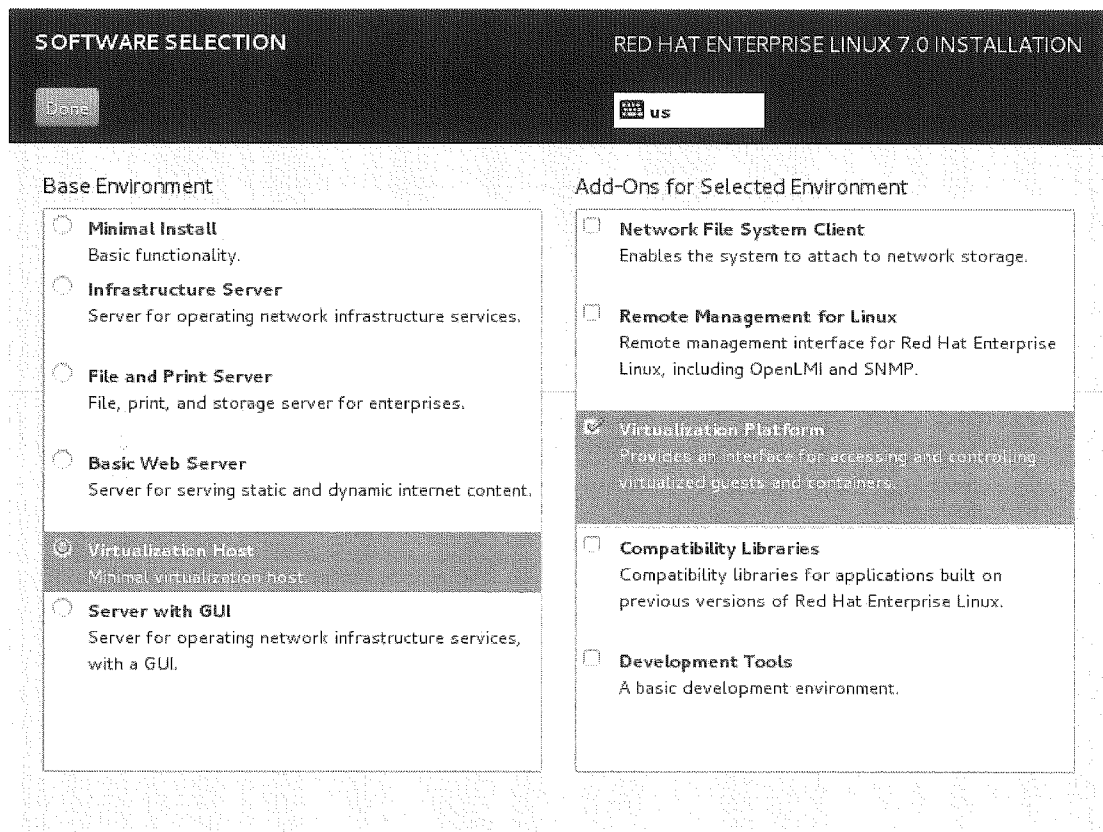


Figure 16.4: Creating a virtualization host during a graphical installation

## Managing virtual machines

The libvirt package is a hypervisor-independent virtualization API to securely manage virtual machines by providing the ability to provision, create, modify, monitor, control, migrate, and stop virtual machines on a single host. The libvirt package provides APIs to enumerate, monitor, and use the resources available on the managed host, including CPUs, memory, storage, and networking. Management tools utilizing libvirt can access host systems remotely using secure protocols.

Red Hat Enterprise Linux uses libvirt-based tools as the default for virtualization management. Support is included for the RHEL 5 Xen hypervisor, and for KVM on RHEL 5, 6, and 7. These management tools utilize libvirt:

- **virsh**—The virsh command-line tool is an alternative to the graphical virt-manager application. Unprivileged users can use virsh in read-only mode, or with root access for full administrative functionality. The virsh command is ideal for scripting virtualization administration.
- **virt-manager**—virt-manager is a graphical desktop tool allowing access to guest consoles, and is used to perform virtual machine creation, migration, configuration, and administrative tasks. Both local and remote hypervisors can be managed through a single interface.
- **RHEV-M**—Red Hat Enterprise Virtualization Manager provides a central management platform for physical and virtual resources, allowing virtual machines to be started, stopped, built, and migrated between hosts. RHEV-M also manages the storage and network components of a data center and provides secure, remote graphical guest console access.

Start the Virtual Machine Manager from the menu at **Applications > System Tools > Virtual Machine Manager**, or by running the **virt-manager** command from the shell. Use this interface to power on or power off virtual machines, assign memory and CPU resources, monitor performance, and connect to the console of the virtual machines.

The **virsh** command-line tool provides the same functionality as **virt-manager**. Use **virsh** as an interactive shell to perform subcommands such as edit, list, start, stop, and destroy. The following examples show the **virsh** commands run as standalone commands from the shell:

```
[root@FoundationX ~]# virsh list
Id Name                               State
-----
 1 desktop                            running
 2 server                             running

[root@FoundationX ~]# virsh destroy server
[root@FoundationX ~]# virsh list --all
Id Name                               State
-----
 1 desktop                            running
 - server                             shut off

[root@FoundationX ~]# virsh start server
[root@FoundationX ~]# virsh list
Id Name                               State
-----
 1 desktop                            running
 2 server                             running
```

**virsh** has subcommands for additional management tasks:

- **connect**—Connect to a local or remote KVM host using **qemu:///host** syntax.
- **nodeinfo**—Returns basic information about the host, including CPUs and memory.
- **autostart**—Configures a KVM domain to start when the host boots.
- **console**—Connect to the virtual *serial* console of a guest.
- **create**—Create a domain from an XML configuration file and start it.
- **define**—Create a domain from an XML configuration file, but do not start it.
- **undefine**—Undefine a domain. If the domain is inactive, the domain configuration is removed.
- **edit**—Edit the XML configuration file for a domain, which will affect the next boot of the guest.
- **reboot**—Reboot the domain, as if the **reboot** command had been run from inside the guest.
- **shutdown**—Gracefully shuts down the domain, as if the **shutdown** command had been run from inside the guest.
- **screenshot**—Takes a screenshot of a current domain console and stores it in a file.



## References

Additional information may be available in the introduction and chapter on system requirements in the *Red Hat Enterprise Linux Virtualization Deployment and Administration Guide* for Red Hat Enterprise Linux 7, which can be found at <http://docs.redhat.com/>

Red Hat Enterprise Virtualization Administration Guide

- Section 1. Basics

Red Hat Enterprise Linux OpenStack Platform 4 Getting Started Guide

- Section 1. Introduction

**virsh(1)**, **virt-manager(1)** man pages

# Practice: Managing a Local Virtualization Host

## Quiz

Match the following items to their counterparts in the table.

create	define	destroy	reboot	shutdown	start
undefine					

Purpose	virsh subcommand
Boot an existing configured virtual machine	
Immediately stop a virtual machine, similar to unplugging it	
Delete the configuration for a virtual machine permanently	
Use an XML configuration to create and boot a virtual machine	
Use an XML configuration to create a virtual machine	
Gracefully stop and restart a virtual machine	
Gracefully stop a virtual machine	

## Solution

Match the following items to their counterparts in the table.

Purpose	virsh subcommand
Boot an existing configured virtual machine	start
Immediately stop a virtual machine, similar to unplugging it	destroy
Delete the configuration for a virtual machine permanently	undefine
Use an XML configuration to create and boot a virtual machine	create
Use an XML configuration to create a virtual machine	define
Gracefully stop and restart a virtual machine	reboot
Gracefully stop a virtual machine	shutdown

## Summary

### Defining the Anaconda Kickstart System

- Kickstart automates Red Hat Enterprise Linux installation using a text file.
- Kickstart configuration files start with commands, followed by the **%packages** section.
- Optional **%post** and **%pre** sections can contain scripting that customizes installations.

### Deploying a New Virtual System with Kickstart

- The **system-config-kickstart** utility can be used to create a Kickstart configuration file.
- Another way to create a Kickstart configuration file is to use a text editor and the **ksvalidator** command to check for syntax errors.
- The **ks=ksfile-location** option to the Anaconda kernel specifies where to find the Kickstart configuration file.

### Managing a Local Virtualization Host

Prepare and build virtualization infrastructure using Red Hat Enterprise Linux.