

**RED HAT®
TRAINING**



Comprehensive, hands-on training that solves real world problems

Red Hat System Administration II

Student Workbook (ROLE)

For use by CStanley.SLFS Copyright © 2017 Red Hat, Inc.

RED HAT SYSTEM ADMINISTRATION II

Red Hat Enterprise Linux 7 RH134

Red Hat System Administration II

Edition 3 20170803

Authors: Wander Boessenkool, Bruce Wolfe, Scott McBrien, George Hacker,
Chen Chang
Editor: Steven Bonneville

Copyright © 2015 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2015 Red Hat, Inc.

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

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

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

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

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

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

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

All other trademarks are the property of their respective owners.

Contributors: Rob Locke, Bowe Strickland, Forrest Taylor, Rudolf Kastl

Reviewers: Michael Phillips, Lars Bohnsack, Michael Bashford, Clint Tinsley

Maintainers: Anuj Verma, Mary Tomson, Michael Jarrett

Document Conventions	xi
Notes and Warnings	xi
Introduction	xiii
Red Hat System Administration II	xiii
Orientation to the Classroom Lab Environment	xiv
Internationalization	xvi
1. Automating Installation with Kickstart	1
Defining the Anaconda Kickstart System	2
Practice: Kickstart File Syntax and Modification	8
Deploying a New Virtual System with Kickstart	12
Practice: Installing a System Using Kickstart	16
Chapter Test: Automating Installation with Kickstart	19
2. Using Regular Expressions with grep	23
Regular Expressions Fundamentals	24
Practice: Match the Regular Expression	27
Matching Text with grep	29
Practice: Using grep with Logs	32
Lab: Using Regular Expressions with grep	34
3. Creating and Editing Text Files with vim	41
The vim Text Editor	42
Practice: vim Modes	44
Basic vim Workflow	46
Practice: Basic vim Workflow	50
Editing with vim	51
Practice: Edit a File with vim	55
Lab: Edit a System File with vim	57
4. Scheduling Future Linux Tasks	61
Scheduling One-Time Tasks with at	62
Practice: Scheduling One-Time Tasks with at	64
Scheduling Recurring Jobs with cron	66
Practice: Scheduling Recurring Jobs with cron	69
Scheduling System cron Jobs	70
Practice: Scheduling System cron Jobs	72
Managing Temporary Files	74
Practice: Managing Temporary Files	77
Chapter Test: Scheduling Future Linux Tasks	79
5. Managing Priority of Linux Processes	83
Process Priority and "nice" Concepts	84
Practice: Process Priority and "nice" Concepts	86
Using nice and renice to Influence Process Priority	88
Practice: Discovering Process Priorities	90
Lab: Managing Priority of Linux Processes	93
6. Controlling Access to Files with Access Control Lists (ACLs)	97
POSIX Access Control Lists (ACLs)	98
Practice: Interpret ACLs	103
Securing Files with ACLs	106
Practice: Using ACLs to Grant and Limit Access	110
Lab: Controlling Access to Files with Access Control Lists (ACLs)	114

7. Managing SELinux Security	123
Enabling and Monitoring Security Enhanced Linux (SELinux)	124
Practice: SELinux Concepts	128
Changing SELinux Modes	130
Practice: Changing SELinux Modes	132
Changing SELinux Contexts	133
Practice: Changing SELinux Contexts	136
Changing SELinux Booleans	138
Practice: Changing SELinux Booleans	140
Troubleshooting SELinux	142
Practice: Troubleshooting SELinux	145
Lab: Managing SELinux Security	148
8. Connecting to Network-defined Users and Groups	153
Using Identity Management Services	154
Practice: Connecting to a Central LDAP and Kerberos Server	161
Lab: Connecting to Network-defined Users and Groups	163
9. Adding Disks, Partitions, and File Systems to a Linux System	169
Adding Partitions, File Systems, and Persistent Mounts	170
Practice: Adding Partitions, File Systems, and Persistent Mounts	181
Managing Swap Space	184
Practice: Adding and Enabling Swap Space	188
Lab: Adding Disks, Partitions, and File Systems to a Linux System	192
10. Managing Logical Volume Management (LVM) Storage	199
Logical Volume Management Concepts	200
Practice: Logical Volume Management Concepts	202
Managing Logical Volumes	204
Practice: Adding a Logical Volume	210
Extending Logical Volumes	214
Practice: Extending a Logical Volume	219
Lab: Managing Logical Volume Management (LVM) Storage	222
11. Accessing Network Storage with Network File System (NFS)	229
Mounting Network Storage with NFS	230
Practice: Mounting and Unmounting NFS	233
Automounting Network Storage with NFS	236
Practice: Automounting NFS	240
Lab: Accessing Network Storage with Network File System (NFS)	244
12. Accessing Network Storage with SMB	251
Accessing Network Storage with SMB	252
Practice: Mounting a SMB File System	256
Lab: Accessing Network Storage with SMB	258
13. Controlling and Troubleshooting the Red Hat Enterprise Linux Boot Process	267
The Red Hat Enterprise Linux Boot Process	268
Practice: Selecting a Boot Target	272
Repairing Common Boot Issues	274
Practice: Resetting a Lost root Password	278
Repairing File System Issues at Boot	280
Practice: Repairing Boot Problems	281
Repairing Boot Loader Issues	283

Practice: Repairing a Boot Loader Problem	285
Chapter Test: Controlling and Troubleshooting the Red Hat Enterprise Linux Boot Process	287
14. Limiting Network Communication with firewalld	291
Limiting Network Communication	292
Practice: Limiting Network Communication	299
Lab: Limiting Network Communication	301
15. Comprehensive Review of System Administration II	307
Red Hat System Administration II Comprehensive Review	308
Lab: Comprehensive Review of System Administration II	311

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.



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

Red Hat System Administration II

This course is specifically designed for students who have completed Red Hat System Administration I (RH124). Red Hat System Administration II (RH134) focuses on the key tasks needed to become a full time Linux Administrator and to validate those skills via the Red Hat Certified System Administrator exam. This course goes deeper into enterprise Linux administration including filesystems and partitioning, logical volumes, SELinux, firewalling, and troubleshooting.

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

- This course is singularly designed for students who have completed Red Hat System Administration I (RH124). The organization of topics is such that it is not appropriate for student to use RH134 as a curriculum entry point. Students who have not taken a previous Red Hat course are encouraged to take either System Administration I if they are new to Linux or the RHCSA Fast Track course (RH200) if they are experienced with enterprise Linux administration.

Prerequisites

- Having sat the Red Hat System Administration I (RH124) course, or equivalent knowledge.

Orientation to the Classroom Lab 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 host names `desktopX.example.com` and `serverX.example.com`, where the number *X* in the computers' host names 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 Red Hat Online Learning classroom, students will be assigned remote computers which will be accessed through a web application hosted at `rol.redhat.com`. Students should log into this machine using the user credentials they provided when registering for the class.

The systems used by each student use separate IPv4 subnets. For a specific student, their IPv4 network is `172.25.X.0/24`, where the number *X* matches the number in the host name of their **desktop** and **server** systems.

Classroom Machines

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

Controlling your stations

The top of the console describes the state of your machine.

Machine States

State	Description
none	Your machine has not yet been started. When started, your machine will boot into a newly initialized state (the desk will have been reset).
starting	Your machine is in the process of booting.
running	Your machine is running and available (or, when booting, soon will be.)
stopping	Your machine is in the process of shutting down.
stopped	Your machine is completely shut down. Upon starting, your machine will boot into the same state as when it was shut down (the disk will have been preserved).
impaired	A network connection to your machine cannot be made. Typically this state is reached when a student has corrupted networking or firewall rules. If the condition persists after a machine reset, or is intermittent, please open a support case.

Depending on the state of your machine, a selection of the following actions will be available to you.

Machine Actions

Action	Description
Start Station	Start ("power on") the machine.
Stop Station	Stop ("power off") the machine, preserving the contents of its disk.
Reset Station	Stop ("power off") the machine, resetting the disk to its initial state. Caution: Any work generated on the disk will be lost.

Action	Description
Refresh	Refresh the page will re-probe the machine state.
Increase Timer	Adds 15 minutes to the timer for each click.

The station timer

Your Red Hat Online Learning enrollment entitles you to a certain amount of computer time. In order to help you conserve your time, the machines have an associated timer, which is initialized to 60 minutes when your machine is started.

The timer operates as a "dead man's switch," which decrements as your machine is running. If the timer is winding down to 0, you may choose to increase the timer.

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 **ttty2**).



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

AUTOMATING INSTALLATION WITH KICKSTART

Overview	
Goal	To automate the installation of Red Hat Enterprise Linux systems with Kickstart.
Objectives	<ul style="list-style-type: none">• Explain Kickstart concepts and architecture.• Create a Kickstart configuration file.
Sections	<ul style="list-style-type: none">• Defining the Anaconda Kickstart System (and Practice)• Deploying a New Virtual System with Kickstart (and Practice)
Chapter Test	<ul style="list-style-type: none">• Automating Installation with Kickstart

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.



Note

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

- **url1**: Specifies the location for the installation media.

Example:

```
url1 --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$YlBb0tXBkWoMuRfb0.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 **systemd** target.

Example:

```
services --disabled=network,iptables,ip6tables --enabled=NetworkManager,firewalld
```

- **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:

The first part of the file consists of the installation commands, like disk partitioning and installation source.

```
#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
```

The second part contains the **%packages** section, detailing which packages and package groups should be installed, and which packages shouldn't be installed.

```
%packages
@core
chrony
cloud-init
dracut-config-generic
dracut-norescue
firewalld
grub2
kernel
rsync
tar
-NetworkManager
-plymouth

%end
```

The last part contains any **%pre** and **%post** installation scripts.

```
%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/documentation/>

Practice: Kickstart File Syntax and Modification

Match the following Kickstart commands with their descriptions in the table.

%packages	%post	auth	clearpart	network
part	rootpw	services	timezone	url

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.	
Defines the default authentication credentials for the superuser.	

Description	Command
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.	%packages
Required Kickstart command that configures how users access the system.	auth
Location of the software used by Kickstart to install a system.	url
Scripting in a Kickstart configuration file that is executed after the software is installed on a system.	%post
Kickstart command that specifies which partitions should be cleared before installation.	clearpart
Modifies which services will start by default at system boot.	services
Defines the default authentication credentials for the superuser.	rootpw
Kickstart command that specifies the size, format, and name of a disk partition.	part
Kickstart command used to specify NTP servers.	timezone

Description	Command
Determines the network configuration for the installation and the target system.	network

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.

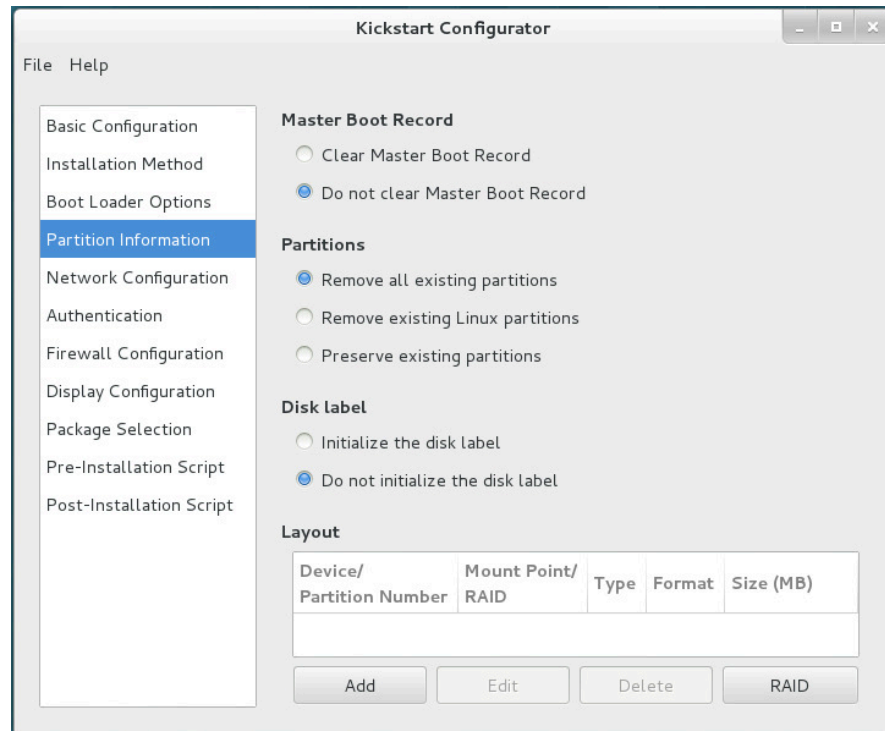


Figure 1.1: Configuring storage with `system-config-kickstart`

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.
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`

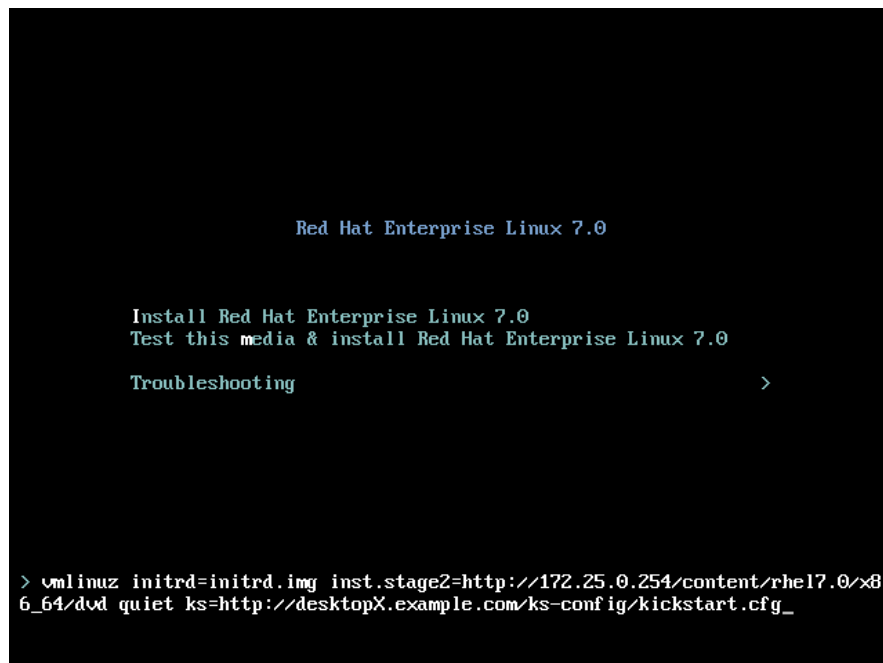


Figure 1.2: Specifying the Kickstart file location during PXE boot

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.



Note

The package selection feature of the `system-config-kickstart` utility is currently disabled due to the following bug (https://bugzilla.redhat.com/show_bug.cgi?id=1272068).



References

`ksvalidator`(1) and `system-config-kickstart`(8) man pages

Practice: Installing a System Using Kickstart

In this lab, you will create a Kickstart configuration file, confirm it is syntactically correct, and publish it for use.

Resources	
Files:	/root/anaconda-ks.cfg
Machines:	desktopX

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.

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
```

Chapter Test: Automating Installation with Kickstart

The steps to install a Red Hat Enterprise Linux server using Kickstart follow. Indicate the order in which the steps should be taken.

- ☐ a. Check the configuration file for syntax errors with **ksvalidator**.
- ☐ b. Boot Anaconda from installation media.
- ☐ c. Use a text editor to add logical volume management commands to the Kickstart configuration file.
- ☐ d. Specify the **ks=** option to point the installer to the Kickstart configuration file.
- ☐ e. Use **system-config-kickstart** to create a Kickstart configuration file.
- ☐ f. Publish the Kickstart configuration file via HTTP, FTP, or NFS.

Solution

The steps to install a Red Hat Enterprise Linux server using Kickstart follow. Indicate the order in which the steps should be taken.

- 3 a. Check the configuration file for syntax errors with **ksvalidator**.
- 5 b. Boot Anaconda from installation media.
- 2 c. Use a text editor to add logical volume management commands to the Kickstart configuration file.
- 6 d. Specify the **ks=** option to point the installer to the Kickstart configuration file.
- 1 e. Use **system-config-kickstart** to create a Kickstart configuration file.
- 4 f. Publish the Kickstart configuration file via HTTP, FTP, or NFS.

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.



CHAPTER 2

USING REGULAR EXPRESSIONS WITH GREP

Overview	
Goal	To write regular expressions using grep to isolate or locate content in text files.
Objectives	<ul style="list-style-type: none">• Create regular expressions to match text patterns.• Use grep to locate content in files.
Sections	<ul style="list-style-type: none">• Regular Expression Fundamentals (and Practice)• Matching Text with grep (and Practice)• Using grep with Logs (and Practice)
Lab	<ul style="list-style-type: none">• Using Regular Expressions with grep

Regular Expressions Fundamentals

Objectives

After completing this section, students should be able to:

- Create regular expressions that match desired data.
- Use **grep** to apply regular expressions to text files.

Writing regular expressions

Regular expressions is a pattern-matching language used for enabling applications to sift through data looking for specific content. In addition to **vim**, **grep**, and **less** using regular expressions, programming languages such as Perl, Python, and C all use regular expressions when using pattern-matching criteria.

Regular expressions are a language of their own, which means they have their own syntax and rules. This section will take a look at the syntax used in creating regular expressions, as well as showing some examples of using regular expressions.

A simple regular expression

The simplest regular expression is an exact match. An exact match is when the characters in the regular expression match the type and order in the data that is being searched.

Suppose that a user was looking through the following file of data looking for all occurrences of the pattern **cat**:

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

cat is an exact match of a **c**, followed by an **a**, followed by a **t**. Using **cat** as the regular expression while searching the previous file gives the following matches:

```
cat
concatenate
category
educated
vindication
```

Using line anchors

The previous section used an exact match regular expression on a file of data. Note that the regular expression would match the data no matter where on the line it occurred: beginning, end, or middle of the word or line. One way that can be used to control the location of where the regular expression looks for a match is a *line anchor*.

Use a **^**, a beginning of line anchor, or **\$**, an end of line anchor. Using the file from earlier:

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

To have the regular expression match **cat**, but only if it occurs at the beginning of the line in the file, use **^cat**. Applying the regular expression **^cat** to the data would yield the following matches:

```
cat
category
```

If users only wanted to locate lines in the file that ended with dog, use that exact expression and an end of line anchor to create the regular expression **dog\$**. Applying **dog\$** to the file would find two matches:

```
dog
chilidog
```

If users wanted to make sure that the pattern was the only thing on a line, use both the beginning and end of line anchors. **^cat\$** would locate only one line in the file, one with a beginning of a line, a **c**, followed by an **a**, followed with a **t**, and ending with an end of line.

Another type of anchor is the *word boundary*. **\<** and **\>** can be used to respectively match the beginning and end of a word.

Wildcards and multipliers

Regular expressions use a **.** as the unrestricted wildcard character. A regular expression of **c.t** will look for data containing a **c**, followed by any one character, followed by a **t**. Examples of data that would match this regular expression's pattern are cat, cot, and cut, but also c5t and cQt.

Another type of wildcard used in regular expressions is a set of acceptable characters at a specific character position. When using an unrestricted wildcard, users could not predict the character that would match the wildcard; however, if users wanted to only match the words cat, cot, and cut, but not odd items like c5t or cQt, replace the unrestricted wildcard with one where acceptable characters are specified. If the regular expression was changed to **c[aou]t**, it would be specifying that the regular expression should match patterns that start with a **c**, are followed by an **a** or an **o** or a **u**, followed by a **t**.

Multipliers are a mechanism used often with wildcards. Multipliers apply to the previous character in the regular expression. One of the more common multipliers used is *****. A *****, when used in a regular expression, modifies the previous character to mean zero to infinitely many of that character. If a regular expression of **c.*t** was used, it would match ct, cat, coat, culvert, etc.; any data that started with a **c**, then zero to infinitely many characters, ending with a **t**.

Another type of multiplier would indicate the number of previous characters desired in the pattern. An example of using an explicit multiplier would be **c.\{2\}t**. Using this regular

expression, users are looking for data that begins with a **c**, followed by exactly any two characters, ending with a **t**.



Note

In the previous examples, Bash regex syntax is being used. There are some slight differences in the syntax used for regular expressions between different implementations (Bash, Python, Perl, etc.).



References

regex(7) man page

Practice: Match the Regular Expression

Match the following words to the regular expression that would uniquely match them in the table.

Error	Installed	^Au.*U	^i	error	s\$
-------	-----------	--------	----	-------	-----

Word or phrase	Regular expression
Aug 19 13:45:41 Updated: lvm2-libs-2.02.95-10.el6_3.3.x86_64	
Aug 19 17:33:15 Installed: wireshark-gnome-1.2.15-2.el6_2.1.x86_64	
io scheduler deadline registered	
Jan 27 10:38:47 serverX NetworkManager[2179]: ifcfg-wlan: error: Missing SSID	
Jan 25 16:02:46 serverX pulseaudio[30014]: main.c: Unable to contact D-Bus: org.freedesktop.DBus.Error.NoServer: Connection refused	
Jan 27 10:39:57 serverX ntpd[2464]: time reset -0.252602 s	

Solution

Match the following words to the regular expression that would uniquely match them in the table.

Word or phrase	Regular expression
Aug 19 13:45:41 Updated: lvm2-libs-2.02.95-10.el6_3.3.x86_64	<code>^Au.*U</code>
Aug 19 17:33:15 Installed: wireshark-gnome-1.2.15-2.el6_2.1.x86_64	<code>Installed</code>
io scheduler deadline registered	<code>^i</code>
Jan 27 10:38:47 serverX NetworkManager[2179]: ifcfg-wlan: error: Missing SSID	<code>error</code>
Jan 25 16:02:46 serverX pulseaudio[30014]: main.c: Unable to contact D-Bus: org.freedesktop.DBus.Error.NoServer: Connection refused	<code>Error</code>
Jan 27 10:39:57 serverX ntpd[2464]: time reset -0.252602 s	<code>s\$</code>

Matching Text with **grep**

Objectives

After completing this section, students should be able to:

- Use the **grep** command with common options.
- Use **grep** to search files and data from piped commands.

Using **grep**

grep is a command provided as part of the distribution which utilizes regular expressions to isolate matching data.

grep usage

The basic usage of **grep** is to provide a regular expression and a file on which the regular expression should be matched.

```
[student@serverX ~]$ grep 'cat$' /usr/share/dict/words
```



Note

Because regular expressions often contain shell metacharacters (such as \$, *, and others), it is recommended practice to use single quotes (') to encapsulate the regular expression on the command line.

grep can be used in conjunction with other commands using a |.

```
[root@serverX ~]# ps aux | grep '^student'
```

grep options

grep has many useful options for adjusting how it uses the provided regular expression with data.

Option	Function
-i	Use the regular expression provided; however, do not enforce case sensitivity (run case-insensitive).
-v	Only display lines that DO NOT contain matches to the regular expression.
-r	Apply the search for data matching the regular expression recursively to a group of files or directories.
-A <NUMBER>	Display <NUMBER> of lines after the regular expression match.
-B <NUMBER>	Display <NUMBER> of lines before the regular expression match.
-e	With multiple -e options used, multiple regular expressions can be supplied and will be used with a logical or.

There are many other options to **grep** as well, but these are some that are used frequently.

grep examples

For the next few examples, use the following file contents, stored in a file named **dogs-n-cats**.

```
[student@serverX ~]$ cat dogs-n-cats
# This file contains words with cats and dogs
Cat
dog
concatenate
dogma
category
educated
boondoggle
vindication
Chilidog
```

Regular expressions are case-sensitive by default; using the **-i** option with **grep** will cause it to treat the regular expression without case sensitivity.

```
[student@serverX ~]$ grep -i 'cat' dogs-n-cats
# This file contains words with cats and dogs
Cat
concatenate
category
educated
vindication
```

Sometimes, users know what they are not looking for, instead of what they are looking for. In those cases, using **-v** is quite handy. In the following example, all lines, case insensitive, that do not contain the regular expression 'cat' will display.

```
[student@serverX ~]$ grep -i -v 'cat' dogs-n-cats
dog
dogma
boondoggle
Chilidog
```

Another practical example of using **-v** is needing to look at a file, but not wanting to be distracted with content in comments. In the following example, the regular expression will match all lines that begin with a # or ; (typical characters that indicate the line will be interpreted as a comment).

```
[student@serverX ~]$ grep -v '^[#;]' <FILENAME>
```

There are times where users need to look for lines that contain information so different that users cannot create just one regular expression to find all the data. **grep** provides the **-e** option for these situations. In the following example, users will locate all occurrences of either 'cat' or 'dog'.

```
[student@serverX ~]$ grep -e 'cat' -e 'dog' dogs-n-cats
# This file contains words with cats and dogs
dog
concatenate
```

```
dogma
category
educated
boondoggle
vindication
Chillidog
```



References

grep (1) man page

Practice: Using grep with Logs

In this lab, you will use regular expressions and **grep** to locate specific log entries in log files.

Resources:	
Files:	/var/log/messages
Machines:	serverX

Outcomes:

Using regular expressions and the **grep** command, you can isolate specific messages or groups of messages based on the search criteria provided.

Before you begin

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

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

1. Elevate your privileges to gain a root login using **su -**.

1.1.

```
[student@serverX ~]$ su -
```

2. Craft a regular expression and use **grep** to display all logs in **/var/log/messages** from the **Start Time** reported by **lab grep setup**.
 - 2.1. The following commands assume a start time provided by the **lab grep** script of April 1 15:53.

Check the current time so we know not only the starting time, but the ending time of the messages we are looking for.

```
[root@serverX ~]# date
Tue Apr  1 15:54:55 EDT 2014
```

2.2.

```
[root@serverX ~]# grep '^Apr 1 15:5[34]' /var/log/messages
Apr  1 15:53:25 serverX ima_daemon[14847]: logging ACCESS:
927265f3c0e95f4ae6294451060d0717
Apr  1 15:53:25 serverX ima_daemon[14848]: logging ACCESS:
b4866e8f2ec0058abe1dc0a142e0b737
Apr  1 15:53:25 serverX ima_daemon[14849]: logging ACCESS:
7afa51b31aabca065dd358cc475d8863
... Output Truncated ...
```

3. Modify your regular expression to locate the first **ERROR** message.

3.1.

```
[root@serverX ~]# grep '^Apr 1 15:5[34].*ERROR' /var/log/messages | head -n 1
Apr  1 15:53:30 serverX database[14877]: bad entry ERROR:
2e28564860d5c6e5151a31fd923c7b61 invalid
```

-
4. Log messages are generated by applications. There is no adopted standard on what keywords or information should be provided as part of a log message.

Using an option to **grep**, look for all logs after the start time that contain the word **ERROR**, ignoring the case of the regular expression.

- 4.1.

```
[root@serverX ~]# grep -i '^Apr 1 15:5[34].*ERROR' /var/log/messages
```

5. Use the **-v** option with **grep**, as well as a regular expression, to locate the **ERROR** message that does not contain a checksum in its message body.

- 5.1. In this situation, it may be useful to use one **grep** and regular expression to meet some of the criteria, and another to further filter the results to get the desired content.

```
[root@serverX ~]# grep '^Apr 1 15:5[34].*ERROR' /var/log/messages | grep -v '[a-z0-9]\{32\}'
```

Lab: Using Regular Expressions with grep

In this lab, you will use regular expressions and **grep** with text files to locate requested data.

Resources:	
Files:	http://classroom.example.com/pub/materials/awesome_logs
Machines:	serverX

Outcomes:

Follow the clues and help Dr. Zingruber recover the lost "artwork."

Dr. Zingruber: "Hello! I hear you are the person to talk to about Red Hat Enterprise Linux systems administration help?"

Yes, that is me.

Dr. Zingruber: "Maybe you can help me, then; you are my last hope. Something terrible has happened. There has been a heist at the Museum of Awesome!"

What was stolen?

Dr. Zingruber: "It was a work by Wander van Gogh."

Wander van Gogh? Never heard of him.

Dr. Zingruber: "I am not surprised. He is a descendant of Vincent van Gogh, but is far, FAR more insane. This is one of his most important pieces, which is why we have it at the Museum of Awesome."

I see. When was the piece taken?

Dr. Zingruber: "It was **August 8 sometime between 1:00pm and 3:00pm.**"

Wait, what? It was taken August 8, and you are just now investigating?

Dr. Zingruber: "Yes, well, to be frank, no one really noticed it was missing until now. You see, while the piece was in the Museum of Awesome, it was in the Hall of Mildly Awesome. If you go to the Museum of Awesome, are you going to the Hall of Mildly Awesome or the Cavern of Supreme Awesome? Because of its placement, no one really looks at it, and just between you and me, it kind of creeps me out."

Um... Okay. So what else can you tell me about the heist?

Dr. Zingruber: "Well, we do have a variety of logs for different things. You can download them from http://classroom.example.com/pub/materials/awesome_logs. I think the place to start your investigation would be **door.log** around the time of the event."

Before you begin

- Reset your **serverX** system.

1. Download the logs to your machine, and change directory to the logs directory.

-
2. Use **grep** to search through the **door.log**. Follow any further instructions you may find in the logs.

Solution

In this lab, you will use regular expressions and **grep** with text files to locate requested data.

Resources:	
Files:	http://classroom.example.com/pub/materials/awesome_logs
Machines:	serverX

Outcomes:

Follow the clues and help Dr. Zingruber recover the lost "artwork."

Dr. Zingruber: "Hello! I hear you are the person to talk to about Red Hat Enterprise Linux systems administration help?"

Yes, that is me.

Dr. Zingruber: "Maybe you can help me, then; you are my last hope. Something terrible has happened. There has been a heist at the Museum of Awesome!"

What was stolen?

Dr. Zingruber: "It was a work by Wander van Gogh."

Wander van Gogh? Never heard of him.

Dr. Zingruber: "I am not surprised. He is a descendant of Vincent van Gogh, but is far, FAR more insane. This is one of his most important pieces, which is why we have it at the Museum of Awesome."

I see. When was the piece taken?

Dr. Zingruber: "It was **August 8 sometime between 1:00pm and 3:00pm.**"

Wait, what? It was taken August 8, and you are just now investigating?

Dr. Zingruber: "Yes, well, to be frank, no one really noticed it was missing until now. You see, while the piece was in the Museum of Awesome, it was in the Hall of Mildly Awesome. If you go to the Museum of Awesome, are you going to the Hall of Mildly Awesome or the Cavern of Supreme Awesome? Because of its placement, no one really looks at it, and just between you and me, it kind of creeps me out."

Um... Okay. So what else can you tell me about the heist?

Dr. Zingruber: "Well, we do have a variety of logs for different things. You can download them from http://classroom.example.com/pub/materials/awesome_logs. I think the place to start your investigation would be **door.log** around the time of the event."

Before you begin

- Reset your **serverX** system.

1. Download the logs to your machine, and change directory to the logs directory.

```
[root@serverX ~]# wget -r -l 1 -np http://classroom.example.com/pub/materials/awesome_logs
[root@serverX ~]# cd classroom.example.com/pub/materials/awesome_logs
```


2. Use **grep** to search through the **door.log**. Follow any further instructions you may find in the logs.

Dr. Zingruber noted that the theft occurred between 1:00 p.m. and 3:00 p.m., or in 24-hour format (used by logs), 13:00 to 15:00. Our regular expression should use the date and hour field of the time to get the relevant entries.

```
[root@serverX awesome_logs]# grep '^Aug *8 1[345]' door.log
```

Note that there are TWO space characters between **Aug** and **8** in the log file date format. To address this, you can use two spaces in your regular expression or a multiplier on the space character. You may have to look through the matched data to find what you are looking for. If you cannot, try the following:

```
[root@serverX awesome_logs]# grep '^Aug *8 14.*OPEN' door.log
... Output Truncated ...
Aug 8 14:37:03 alarm_monitor activity: back door: OPEN Dr Zingruber: "Oh yes...
Aug 8 14:40:01 alarm_monitor activity: back door: OPEN look here, you can see
Aug 8 14:41:26 alarm_monitor activity: back door: OPEN the door stayed open.
Aug 8 14:43:55 alarm_monitor activity: back door: OPEN Now that we know a more
Aug 8 14:46:20 alarm_monitor activity: back door: OPEN exact time, we should
Aug 8 14:48:31 alarm_monitor activity: back door: OPEN check wall.log for the
Aug 8 14:51:30 alarm_monitor activity: back door: OPEN same period.
... Output Truncated ...
```

In the **door.log** entries, we were referred to the **wall.log** file, but we now have a more narrow time. Use **grep** to look between time codes 14:37 and 14:51.

```
[root@serverX awesome_logs]# grep '^Aug *8 14:[345]' wall.log
```

Note that, again, there are TWO space characters between **Aug** and **8** in the log file date format. To address this, you can use two spaces in your regular expression or a multiplier on the space character. You may have to look through the matched data to find what you are looking for. If you cannot, try the following:

```
[root@serverX awesome_logs]# grep '^Aug *8 14.*ALERT' wall.log
Aug 8 14:37:03 alarm_monitor ALERT: Mildly Awesome: Dr. Zingruber: Ah, yes here
Aug 8 14:40:01 alarm_monitor ALERT: Mildly Awesome: it is, looks like they
Aug 8 14:41:26 alarm_monitor ALERT: Mildly Awesome: digitized the image. We
Aug 8 14:43:55 alarm_monitor ALERT: Mildly Awesome: should check proxy.log at
Aug 8 14:46:20 alarm_monitor ALERT: Mildly Awesome: 14:40. The digitalized
Aug 8 14:48:31 alarm_monitor ALERT: Mildly Awesome: image will be on the 24
Aug 8 14:51:30 alarm_monitor ALERT: Mildly Awesome: lines following the log.
```

In the **wall.log** entries, we were referred to the **proxy.log** file, but we now have an exact time. Use **grep** to look between time code 14:40. Additionally, not only do we want the line for time code 14:40, but also the 24 lines that follow that log entry.

```
[root@serverX awesome_logs]# grep -A 24 '14:40' proxy.log
```

You should now have recovered the "artwork."

```

Aug  8 14:40:03 Outbound data Captured...Dr. Zingruber: You found it, thank you!
.....MMMMMMMMMMMMMMN~.....
.....:MMMMMMMMMMMMMMMMMMMMMM?.....
.....DMMMMMMN88MMMMNZZZZMMMMMMN.....
.....+MMMMMMZZZZZZZZZZMM8ZMMMMMM?.....
.....MMMMMMZZZZZZZZZZZZZZMMMMMM.....
.....MMMMMMMMZOMMMMMZZZZZZZZMMMMMM.....
.....MMMMMMMMMOZZNZZZZZZZZZZZMMMMMMMM.....
.....MMMMMDDDNMZZZZZZZZZZZZZZZMMMMMMMM.....
.....+MM$ZZZZZ8MMZZZZZZZZZZZZZMMMMMM~.....
.....MMMZZZZZZDMMMMMMNZZZZZZZZZNZ8MMMMMM.....
.....MMMOZZZZZZZMMMMMMZZZZZZZZZZZZDMMMM.....
.....,MMMMMZZZZZZZZZZZMMZZZZZZZZZZZZZMMZ.....
.....DMMMMMMMZZZZZZZZZZZZZZZZZZZZZZZMMMD.....
.....ZMMMMMMMMMDZZZZZZZZZZZZZZZZZZZZMMMN.....
.....,MMMMMMN...:MNZZZZZZZZZZZZZZZZZZMM7.....
.....MMMMMM?... M, MMMZZZZZZZZZZZZZZMMMMMM.....
.....MMMMMN7MMI..... MMMMMMMNNNNMMMMMMMMMM.....
.....      .7MM      IDZ=..$, =I MMMMMMMMM.....
.....MMMMMMMMMM.....MMMMMMMMMM.....
.....IMMM,...NMMMMMMMM Z88.....
.....MMN.....
.....IMD.....
.....$M.....

```

Summary

Regular Expressions Fundamentals

Write regular expressions to match data.

Matching Text with **grep**

Using **grep** with regular expressions to isolate text data.



CHAPTER 3

CREATING AND EDITING TEXT FILES WITH VIM

Overview	
Goal	Introduce the <code>vim</code> text editor.
Objectives	<ul style="list-style-type: none">• Explain the three main modes of <code>vim</code>.• Open, edit, and save text files.• Use editor shortcuts.
Sections	<ul style="list-style-type: none">• The <code>vim</code> Text Editor (and Practice)• Basic <code>vim</code> Workflow (and Practice)• Editing with <code>vim</code> (and Practice)
Lab	<ul style="list-style-type: none">• Edit a system file with <code>vim</code>

The vim Text Editor

Objectives

After completing this section, students should be able to:

- Explain the three main modes of **vim**.

Introduction to vim

Editing text files is one of the most common tasks a system administrator will perform on a Linux system. As such, there is a wide variety of text editors available. One of the older, but most widely used, editors is **vi**. **vi** stands for *Visual Interface*, as it was one of the first text editors to actually display the working document while it was being edited. Before that, most editors were line-based (such as **ed**, and the still widely used **ex**). In regular use, **vi** and **vim** are normally referred to as *v.i.* (two letters) and *vim*.

VI IMproved

The version of **vi** that is shipped with Red Hat Enterprise Linux 7 is called **vim**. **vim** stands for *VI IMproved*, as **vim** comes with many features not found in the original **vi**, while still remaining (mostly) backward-compatible. Among the new features are popular options, such as syntax highlighting, completion modes, and spell-checking.

vim is highly extensible. It supports scripting in multiple languages, file-type plug-ins, different text-completion modes, and many other options. It can be adapted to almost any role, and has been. There are extensions and macros available on the Internet for almost any purpose, from helping to edit a certain type of file (such as DocBook), completion plus introspection for almost all programming languages in existence, to more mundane tasks such as managing ToDo lists.



Important

When an unprivileged user invokes the command **vi** on a Red Hat Enterprise Linux 7 machine, the command that is executed will be **vim**. This is done with an alias that is set from **/etc/profile.d/vim.sh** when the shell starts.

This alias is *not* set for users with a UID less than or equal to **200**. These users will execute **vi**, which is **vim** in **vi** compatible mode. This means that any features not found in classic **vi** will be disabled.

It is recommended to always execute the **vim** command whenever the newer features are wanted, and to not rely on an alias that might not be available. This is recommended especially when users also regularly have to work as **root**.

Why learn vim?

Every system administrator will have a preference for a text editor. Some will prefer **gedit**, others like **nano**, and there even are people who prefer **emacs**. Even if one already has an editor of choice, it is important to be familiar with the basics of **vim** or **vi** for one simple reason: It's the editor that one can count on to be installed on whatever system is being worked on.

Different versions of **vim**

Three distinct variations of **vim** can be installed on an Red Hat Enterprise Linux system. Each version has its own use case, and variations can be installed side by side. The variations come in these three packages:

- *vim-minimal*: This package only provides **vi** and related commands (like **rvi**, the restricted version that cannot spawn commands or a shell). This is the version included in a minimal installation of Red Hat Enterprise Linux 7.
- *vim-enhanced*: This package provides the **vim** command (and friends), providing features such as syntax highlighting, file-type plug-ins, and spell checking.
- *vim-X11*: This package provides **gvim**, a version of **vim** that runs in its own graphical window instead of in a terminal. One of the big features of **gvim** is the menu bar, useful when one is learning **vim** or can't remember a specific command. (Note: Depending on the terminal type and **vim** per-user configuration, it can be possible to use a mouse inside a regular **vim** session as well.)

A modal editor

vim is not the easiest editor to learn. This is partly because all commands in **vim** are geared toward speed and efficiency, and not ease of remembrance, and partly because **vim** is a *modal* editor. *Modal editor* means that the function of certain commands and key presses changes based on what mode is active.

vim has three primary modes:

Function	Mode
Command mode	This mode is used for file navigation, cut and paste, and simple commands. Undo, redo, and others are also performed from this mode.
Insert mode	This mode is used for normal text editing. Replace mode is a variation on insert mode that replaces text instead of inserting it.
Ex mode	This mode is used to save, quit, and open files, as well as search & replace and other more complex operations. From this mode, it is possible to insert the output of programs into the current file, configure vim , and much more. Everything that is possible using ex can be done from this mode.



References

vim(1) man page

vim built-in help

Practice: vim Modes

Match the items that follow to their counterparts in the table.

Command mode	Ex mode	Insert mode
---------------------	----------------	--------------------

Function	Mode
This mode is used for file navigation, cut and paste, and simple commands.	
This mode is used for normal text editing.	
This mode is used to save, quit, and open files, as well as search & replace and other more complex operations.	

Solution

Match the items that follow to their counterparts in the table.

Function	Mode
This mode is used for file navigation, cut and paste, and simple commands.	Command mode
This mode is used for normal text editing.	Insert mode
This mode is used to save, quit, and open files, as well as search & replace and other more complex operations.	Ex mode

Basic vim Workflow

Objectives

After completing this section, students should be able to:

- Open text files.
- Move the cursor.
- Insert and replace text.
- Save files.
- Get help.

Editor basics

No matter what editor you use, you should always be able to perform the following three tasks:

- Open a new or existing file.
- Make changes and/or insert new text.
- Save the file and exit the editor.

Opening files

The easiest way to open a file in **vim** is to specify it as an argument on the command line. For example, to open the file called **/etc/hosts**, you could execute the following command:

```
[root@serverX ~]# vim /etc/hosts
```



Note

If you try to open a file that does not exist, but the directory you specify is available, **vim** will inform you that you are editing a **[New File]**, and create the file when you first save it.

After opening a file, **vim** will start in *command* mode. At the bottom left of the screen, you will see information about the opened file (filename, number of lines, number of characters). At the bottom right, you will see the current cursor position (line, character), and what part of the file is being displayed (**All** for all, **Top** for the first lines of a file, **Bot** for the bottom of a file, or a percentage to indicate where in the file you are). The bottom line is called the *Ruler* in **vim** terms.

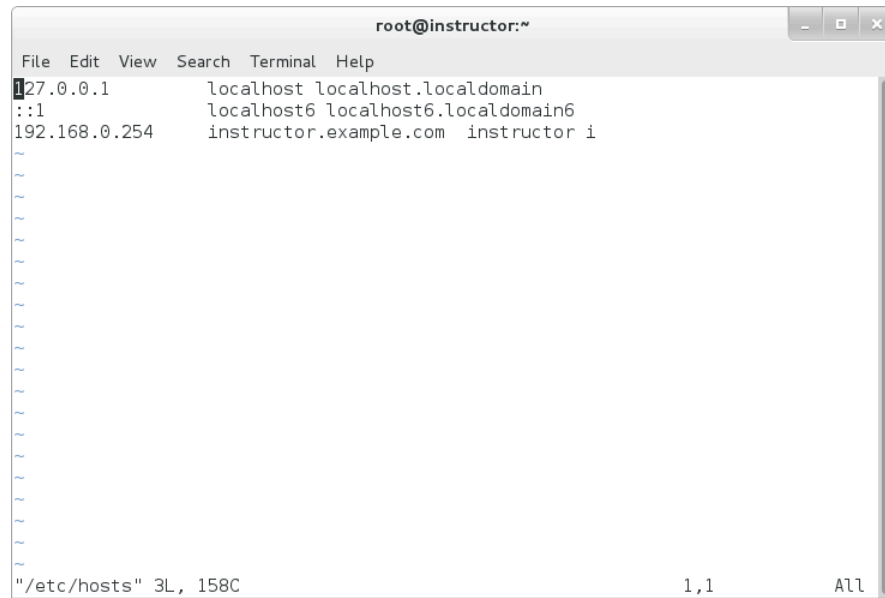


Figure 3.1:
vim displaying a freshly opened file

Editing text

If you have ever used **vi** or **vim** before, you might have noticed that in *command* mode, most keys don't exactly do what you would expect. This is because in *command* mode, keys are not mapped to insert the characters you press, but rather to perform commands like cursor movements, copy-and-paste actions, and more.

To switch to *insert* mode, there are commands available to you, each assigned to a different key on your keyboard:

Key	Result
i	Switch to <i>insert</i> mode, and start inserting <i>before</i> the current cursor position (insert).
a	Switch to <i>insert</i> mode, and start inserting <i>after</i> the current cursor position (append).
I	Move the cursor to the <i>start</i> of the current line and switch to <i>insert</i> mode.
A	Move the cursor to the <i>end</i> of the current line and switch to <i>insert</i> mode.
R	Switch to <i>replace</i> mode, starting at the character under your cursor. In <i>replace</i> mode, text is not inserted, but each character you enter replaces a character in the current document. (vim and vi also come with more powerful replacement commands; these are discussed in another section.)
o	Open a new line <i>below</i> the current one, and switch to <i>insert</i> mode.
O	Open a new line <i>above</i> the current one, and switch to <i>insert</i> mode.

Whenever you are in *insert* or *replace* mode, the ruler will display **--INSERT--** or **--REPLACE--**. To return to *command* mode, you can press **Esc**.

The version of **vi** and **vim** that ships with Red Hat Enterprise Linux is configured to recognize and use the normal cursor keys, as well as keys like **PgUp** and **End** while in both *insert* and

command mode. This is not the default behavior on all installations of **vi**. In fact, older versions of **vi** did not recognize cursor keys at all, and only allowed you to move the cursor from within *command* mode using keys like **hjkL**.

In the following table, you will find some of the keys you can use from *command* mode to move your cursor:

Key	Result
h	Cursor left one position
l	Cursor right one position
j	Cursor down one line
k	Cursor up one line
^	Move to the beginning of the current line.
\$	Move to the end of the current line.
gg	Move to the first line of the document.
G	Move to the last line of the document.



Note

Pressing **Esc** will always cancel the current command, or return to *command* mode. It is an often-seen practice to press **Esc** twice (or even more) to ensure a return to *command* mode.

Saving files

Saving files in **vim** is done from *ex* mode. You can enter *ex* mode by pressing **:** (a colon) from within *command* mode. After entering *ex* mode, the ruler will display a colon (**:**) and wait for a command to be typed. Commands are completed by pressing **Enter**.

The following is a short list of commands to save and quit your current file from *ex* mode. This is by no means a full list of commands that can be used.

Command	Result
:wq	Save and quit the current file.
:x	Save the current file if there are unsaved changes, then quit.
:w	Save the current file and remain in editor.
:w <filename>	Save the current file under a different file name.
:q	Quit the current file (only if there are no unsaved changes).
:q!	Quit the current file, ignoring any unsaved changes.

A short summary of the previous table is that **w** saves (writes), **q** quits, and **!** forces an action (do-what-I-say-not-what-I-want).

Getting help

vim comes with extensive online help, available in the editor itself. Typing **:help** from *command* mode will launch the first screen, which includes the help needed to navigate the help.

Help for a specific subject can be obtained by typing **:help *subject*** from *command* mode.

Help screens open in a new *split window*, and can be closed with **:q**. To learn more about split windows, use **:help windows**.

There is also a semi-interactive tutor available. Starting the command **vimtutor** from the command line will launch a guided tour of **vim** that takes a new user through the basics in about an hour.



References

vim(1) man page

vim built-in help

Practice: Basic vim Workflow

In this lab, you will edit a text file using **vim**.

Resources	
Machines:	desktopX

Outcomes

A successfully edited text file.

Before you begin

A working **student** account on **desktopX**.

1. Log into your **desktopX** system as **student** and open a terminal.
2. Open the (new) file **/home/student/vim-practice.txt** in **vim**. You do not have to create this file first.

2.1.

```
[student@desktopX ~]$ vim vim-practice.txt
```

3. Insert the following text:

```
This is my vim practice file.
There are many like it, but this one is mine.
```

- 3.1. Press **i** or **a** to go into *insert* mode.
- 3.2. Type the text shown previously.
- 3.3. Press **Esc** to go back into *command* mode.
4. Insert a new line at the bottom with the following contents:

```
More lines, I want more lines!
```

- 4.1. Press **o** to open a new line beneath the current one and immediately switch to *insert* mode.
- 4.2. Type the line to be added.
- 4.3. Press **Esc** to go back into *command* mode.
5. Save and quit your file.
 - 5.1. From *command* mode, enter **:wq**, then press **Enter**.

Editing with vim

Objectives

After completing this section, students should be able to:

- Use movement shortcuts.
- Copy and paste text.
- Use search and replace.
- Undo (and redo) their actions.

Movement

In addition to the boring old single character/line cursor movements one can perform in *command* mode, there are also quite a few advanced movement commands to help users navigate documents more efficiently. These shortcuts allow the cursor to be moved per word, sentence, or paragraph. Keep in mind that unlike regular cursor movements, these commands only work in *command* mode, and not in *insert* mode.

Key	Result
w	Move cursor to beginning of next word (W includes punctuation).
b	Move cursor to beginning of previous word (B includes punctuation).
(Move cursor to beginning of current or previous sentence.
)	Move cursor to beginning of next sentence.
{	Move to beginning of current or previous paragraph.
}	Move cursor to beginning of next paragraph.

All movement commands can be prefixed by typing a number, e.g., **5w** to move the cursor five words, or **12j** to move the cursor 12 lines down. In fact, every single command (including switching to *insert* mode) can be repeated a fixed number of times by typing the number of repeats before the actual command. In **vim** terminology, this is referred to as the *count*.

Replacing text

vim allows users to easily replace large (and small) amounts of text using a “change” command. The “change” command works by pressing the **c** key, followed by a cursor movement; for example, **cw** to change from the current cursor position to the end of the current word. The text to be replaced is deleted (and put on the unnamed register), and the **vim** switches to *insert* mode.

There are a few shortcuts available to make editing even more efficient:

- Pressing **c** twice (**cc**) will start replace in a *line-wise* fashion, replacing the entire line (or multiple lines when prefixed with a number). This same trick applies to a number of other commands (such as delete) as well.
- Most movement commands can be prefixed with **i** and **a** to select the *inner* or *a* version of the movement. For example, **ciw** will replace the entire current word, not just from the current cursor position, and **caw** will do the same, but including any surrounding white space.

- To replace to the end of the line, one can use **c\$**, but **C** does the same. (This trick also applies to various other commands, such as deleting.)
- To just replace the character under the cursor, press **r** followed by the new character.
- To change the case of the character under the cursor, press **~**.

Deleting text

Deleting text works the same as replacing text. The command for deleting text is **d**, and all the same movements that are valid for changing text apply to deleting as well, including **D** to delete from the cursor to the end of the line.

To just delete the character under the cursor, use **x**.

Copy and paste

vim uses slightly different terminology to describe copy and paste operations than most people are used to currently. A copy operation is called *yank*, and paste is called *put*. This is reflected in the keyboard commands assigned to these operations: *yank* is **y** followed by a movement, and *put* operations are performed with **p** and **P**.

Yank operations follow the same general schema as replace and delete operations: A user optionally types the number of times to repeat an operation, followed by **y**, followed by a movement. For example, **5yaw** will copy the current word and the next four (for a total of five). Pressing **yy** will yank the entire line, etc.

Putting (pasting) is done with the **p** and **P** commands; lowercase **p** will put *after* the current cursor (or below the current line when line-wise data is being pasted), while uppercase **P** puts *before* the current cursor position, or above the current line. Like all other commands, a put command can be prefixed with the number of times to paste the register.

Multiple registers

Instead of just one clipboard for copy and paste, **vim** has 26 named *registers*, and a number of special purpose registers as well. Having multiple registers available allows users to more efficiently cut and paste, without having to worry about losing data or moving the cursor around too much. If a register to use is not specified, the “unnamed” register will be used. Normal registers are called **a** to **z**, and are selected by putting "**registername**" between the *count* for a command and the actual command; for example, to copy the current line and the next two into the **t** register, one can use the command **3"tyy**.

To put out of a named register, simply put "**registername**" in front of the put command; for example, **"sp** will put after the cursor out of the **s** register.



Important

Whenever a named register is used, the unnamed register will be updated as well.

Delete and change operations can be prefixed with a register selection as well. When no register is specified, only the unnamed register will be used. When the uppercase version of a register is used, the text that is being cut or yanked is appended to that register instead of overwriting it.

Special registers

There are 10 *numbered* registers, "**0**" through "**9**". Register "**0**" will always have a copy of the most recent yanked text, while register "**1**" will have a copy of the most recent deleted text. When new text is changed or deleted, the contents of "**1**" will shift into "**2**", "**2**" into "**3**", etc.



Important

Unlike the named registers, the content of the numbered registers is **not** saved between sessions.

Visual mode

To avoid having to constantly count the number of lines, words, or characters to specify for commands, **vim** also comes with a *Visual* (select) mode. After entering *visual* mode (indicated by **--VISUAL--** in the ruler), any cursor movements will start selecting text. Any change, delete, or yank commands issued in *visual* mode do not need a cursor movement part, but will instead work on the selected text.

Visual mode comes in three flavors: character-based (started with **v**), line-based (started with **V**), and block-based (started with **Ctrl+V**). When using **gvim**, the mouse can also be used to select text.

Any **ex** commands issued in *visual* mode will by default work on the selected text as well.

Searching

Searching in the current document can be started in two ways: by pressing **/** to search forward from the cursor position, or by pressing **?** to search backward from the current cursor position. After entering search mode, a regular expression can be typed to search for, and pressing **Enter** will jump to the first match (if any).

To search for the next or previous match, use **n** and **N** respectively.

Bonus shortcut: ***** will instantly search forward for the word under the cursor.

Search and Replace

Search and replace in **vim** is implemented in **ex** mode, and uses the same syntax as one would use with **sed** for search and replace, including the capability to search using regular expressions:

ranges/pattern/string/flags

range can be a line number (**42**), a range of line numbers (**1, 7** for lines 1-7), a search term (**/README\.txt/**), **%** for all the lines in the current document (search and replace normally only works on the current line), or **'<, '>** for the current *visual* selection.

Two of the most common **flags** are **g**, to enable replacing more than one occurrence of **pattern** per line, and **i**, to make the current search case-insensitive.

Search and Replace example

For example, to search for every occurrence of the word "cat" and replace it with "dog" in all lines, regardless of case, but only if it's a full word, and not in something like "catalog", one could use the following command:

```
:%s/\<cat\>/dog/gi
```

Undo and redo

To allow for human imperfection, **vim** is fitted with an undo/redo mechanism. Simply pressing **u** in *command* mode will undo the last action. If too much has been undone, pressing **Ctrl+r** will redo the last undo.

Bonus awesomeness: Pressing **.** (period) from *command* mode will redo the last edit action, but on the current line. This can be used to easily perform the same edit action multiple times.



References

vim(1) man page

vim built-in help

Practice: Edit a File with vim

In this lab, you will edit a file using **vim**.

Resources	
Machines:	desktopX
Files:	/usr/share/doc/vim-common-*/README.txt

Outcomes

A copy of the **vim README.txt** which has been edited according to the instructions in this practice exercise.

Before you begin

N/A

1. Log into your **desktopX** system as **student** and open a terminal.
2. Create a copy of the file **/usr/share/doc/vim-common-*/README.txt** in your home directory.

2.1.

```
[student@desktopX ~]$ cp /usr/share/doc/vim-common-*/README.txt .
```

3. Open **/home/student/README.txt** in **vim**.

3.1.

```
[student@desktopX ~]$ vim README.txt
```

4. Jump to the section titled **MAIN AUTHOR**, then put your cursor on the **A** in **AUTHOR**.
 - 4.1. From *command* mode, type the following, then press **Enter**. This will jump to the first occurrence of the text:

```
/MAIN AUTHOR
```

- 4.2. Press **w** to move the cursor one word to the right; this should put you on the **A** in **AUTHOR**.
5. Change this occurrence of the word **AUTHOR** to **ROCKSTAR**.
 - 5.1. From *command* mode, type **cw** to change the word under the cursor.
 - 5.2. Type **ROCKSTAR**.
 - 5.3. Press **Esc** to return to *command* mode.
6. Undo your previous edit.
 - 6.1. Press **u** to undo your last edit.
7. Redo (i.e., undo your undo) your last edit.

- 7.1. Press **Ctrl+r** to redo your last undo.
8. Using *visual* mode, make a copy of the **INSTALLATION** paragraph (including header) and place it at the end of the file.

- 8.1. Move the cursor to the start of the **INSTALLATION** section by searching for **^INSTALLATION**. From *command* mode, type:

```
/^INSTALLATION
```

- 8.2. Enter *visual line* mode by pressing **V**.
- 8.3. Move the cursor to the end of section by typing **3}**. This will move the cursor three paragraphs down, selecting our entire section. (The heading counts as a paragraph.)
- 8.4. Press **y** to *yank* (copy) the selected lines to the unnamed buffer.
- 8.5. Move the cursor to the end of the document by pressing **G**.
- 8.6. *Put* (paste) the unnamed buffer below the current line by pressing **p**.
9. In the entire document, replace each occurrence of **README** with **PLEASE_READ_ME**.
- 9.1. From *command* mode, type the following:

```
:%s/README/PLEASE_READ_ME/g
```

- The **:** enters *Ex* mode.
 - **%** indicates that we want to work on every line in the document.
 - **s/README/PLEASE_READ_ME/** is the search and replace command.
 - The trailing **g** indicates that this replace operation can be performed more than once per line.
10. Exit without saving your changes.
 - 10.1. From *Command* mode type **:q!**.
- The **:** enters *ex* mode, the **q** indicates we want to quit, and the **!** tells **vim** to force the quit, since we have unsaved changes.
11. Clean up by removing your **README.txt**.

- 11.1.

```
[student@desktopX ~]$ rm README.txt
```

Lab: Edit a System File with vim

In this lab, you will create and edit a new system file using **vim**.

Resources	
Files:	/etc/motd
Machines:	desktopX

Outcomes

An updated **/etc/motd** file on **desktopX**.

Before you begin

N/A

You have been asked to update the *Message-Of-The-Day* (MOTD) file on **desktopX**. This file is called **/etc/motd**, and its contents are displayed to users upon a successful login on the command line.

1. Update the **/etc/motd** file on **desktopX** to read exactly as it read in below text block without replacing the value of "X" in this step:

```
desktopX.example.com

Please be careful.
```

2. Test your changes by using **ssh** to connect to the **student** account on **localhost**. If all goes well, you should see your new message after authentication. Close the **ssh** connection when you are done testing.
3. Edit **/etc/motd** again. This time, replace the **X** in **desktopX.example.com** with your actual station number, using search and replace. You are also asked to repeat the "**Please be careful.**" line two more times.
4. Test your changes by using **ssh** to connect to **student@localhost** again.

Solution

In this lab, you will create and edit a new system file using **vim**.

Resources	
Files:	<code>/etc/motd</code>
Machines:	<code>desktopX</code>

Outcomes

An updated `/etc/motd` file on **desktopX**.

Before you begin

N/A

You have been asked to update the *Message-Of-The-Day* (MOTD) file on **desktopX**. This file is called `/etc/motd`, and its contents are displayed to users upon a successful login on the command line.

1. Update the `/etc/motd` file on **desktopX** to read exactly as it read in below text block without replacing the value of "X" in this step:

```
desktopX.example.com
Please be careful.
```

- 1.1. Log into your **desktopX** system as **student** and open a terminal.
- 1.2. Since `/etc/motd` is a system file, you will need to elevate your privileges.

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

- 1.3. Open `/etc/motd` in **vim**.

```
[root@desktopX ~]# vim /etc/motd
```

- 1.4. Enter *insert* mode by pressing **i** or **a**, then type the following text:

```
desktopX.example.com
Please be careful.
```

- 1.5. Press **Esc** to exit *insert* mode and return to *command* mode, then type `:wq` to enter *ex* mode to save and quit.
2. Test your changes by using **ssh** to connect to the **student** account on **localhost**. If all goes well, you should see your new message after authentication. Close the **ssh** connection when you are done testing.

- 2.1.


```
[root@desktopX ~]# ssh student@localhost
The authenticity of host 'localhost (::1)' can't be established.
```

```

RSA 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' (RSA) to the list of known hosts.
student@localhost's password: student
desktopX.example.com

Please be careful.
[student@desktopX ~]$ exit

```

3. Edit **/etc/motd** again. This time, replace the **X** in **desktopX.example.com** with your actual station number, using search and replace. You are also asked to repeat the “**Please be careful.**” line two more times.

- 3.1. Open **/etc/motd** in **vim**. Make sure that you are still working as **root**.

```
[root@desktopX ~]# vim /etc/motd
```

- 3.2. Use search and replace to replace **X** with your actual station number. The example that follows assumes that you are station number **99**.

From *command* mode, enter *ex* mode and replace all occurrences of **X** with **99** by typing the following:

```
:%s/X/99/g
```

- 3.3. Move your cursor to line number three by typing the following from *command* mode:

```
:3
```

- 3.4. Yank (copy) the current line, then put (paste) it twice, by typing **yy2p**.

The **yy** part yanks the current line, and **2p** puts it twice.

- 3.5. Save and quit by typing **:wq**.

4. Test your changes by using **ssh** to connect to **student@localhost** again.

- 4.1.

```

[root@desktopX ~]# ssh student@localhost
student@localhost's password: student
desktop99.example.com

Please be careful.
Please be careful.
Please be careful.
[student@desktopX ~]$ exit

```

Summary

The vim Text Editor

- **vim** has three main modes.
 - *Command* mode for file navigation and simple commands.
 - *Insert* mode for normal text editing.
 - *Ex* mode for saving, quitting, and performing more complex commands.

Basic **vim** Workflow

- Both the cursor keys and **hjk1** can be used to move the cursor.
- **Escape** exits the current command or mode, press twice to always end in command mode.
- **:w** saves, **:q** quits, **:wq** saves and quits.

Editing with vim

- Fast cursor commands: **wb () {}**.
- **c** enter change mode.
- **d** and **y** to cut and copy, **p** to paste.



CHAPTER 4

SCHEDULING FUTURE LINUX TASKS

Overview	
Goal	Schedule tasks to automatically execute in the future.
Objectives	<ul style="list-style-type: none">• Schedule one-time tasks with at.• Schedule recurring jobs with cron.• Schedule recurring system jobs.• Manage temporary files.
Sections	<ul style="list-style-type: none">• Scheduling One-Time Tasks with at (and Practice)• Scheduling Recurring Jobs with cron (and Practice)• Scheduling System cron Jobs (and Practice)• Managing Temporary Files (and Practice)
Chapter Test	<ul style="list-style-type: none">• Scheduling Future Linux Tasks

Scheduling One-Time Tasks with at

Objective

After completing this section, students should be able to schedule one-time tasks with **at**.

Scheduling future tasks

From time to time, an administrator (or end user) wants to run a command, or set of commands, at a set point in the future. Examples include the office worker who wants to schedule an email to his boss, as well as the system administrator working on a firewall configuration who puts a “safety” job in place to reset the firewall settings in ten minutes' time, unless he deactivates the job before then.

These scheduled commands are often called *tasks* or *jobs*.

Scheduling one-time tasks with at

One of the solutions available to users of a Red Hat Enterprise Linux system for scheduling future tasks is **at**. This is not a standalone tool, but rather a system daemon (**atd**), with a set of command-line tools to interact with the daemon (**at**, **atq**, and more). In a default Red Hat Enterprise Linux installation, the **atd** daemon will be installed and enabled automatically. The **atd** daemon can be found in the **at** package.

Users (including **root**) can queue up jobs for the **atd** daemon using the command-line tool **at**. The **atd** daemon provides 26 queues, **a** to **z**, with jobs in alphabetically later queues getting less system priority (higher *nice* levels, discussed in a later chapter).

Scheduling jobs

A new job can be scheduled by using the command **at <TIMESPEC>**. **at** will then read the commands to execute from **stdin**. For larger commands, and typo-sensitive commands, it is often easier to use input redirection from a script file, e.g., **at now +5min < myscript**, than typing all the commands by hand in a terminal window. When entering commands by hand, you can finish your input by pressing **Ctrl+D**.

The **<TIMESPEC>** allows for many powerful combinations, giving users an (almost) free-form way of describing exactly when a job should be run. Typically, they start with a time, e.g., **02:00pm**, **15:59**, or even **teatime**, followed by an optional date or number of days in the future.

Some examples of combinations that can be used are listed in the following text. For a complete list, see the **timespec** definition in the references.

- **now +5min**
- **teatime tomorrow** (teatime is **16:00**)
- **noon +4 days**
- **5pm august 3 2016**

Inspecting and managing jobs

Inspecting jobs

To get an overview of the pending jobs for your user, use the command **atq** or, alternatively, the alias **at -l**.

Running this command gives the following output:

```
[student@desktopX ~]$ atq
28 Mon Feb  2 05:13:00 2015 a student
29 Mon Feb  3 16:00:00 2014 h student
27 Tue Feb  4 12:00:00 2014 a student
```

This shows four columns for every job scheduled to run in the future:

- The job number, **28** in the first line.
- The date and time scheduled for that job, **Mon Feb 2 05:13:00 2015** in the first line.
- The queue for the job, **a** in the first line, but **h** in the second.
- The owner of the job (and the user as which the job will run), **student** in all our lines.



Important

Normal, unprivileged users can only see and control their own jobs. **root** can see and manage all jobs.

To inspect the actual commands that will run when a job is executed, use the command **at -c <JOBNUMBER>**. This output will first show the *environment* for the job being set up to reflect the environment of the user who created the job at the time it was created, followed by the actual commands to be run.

Removing jobs

The **atrm <JOBNUMBER>** will remove a scheduled job. This is useful when a job is no longer needed; for example, when a remote firewall configuration succeeded, and does not need to be reset.



References

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

/usr/share/doc/at-*/timespec

Practice: Scheduling One-Time Tasks with at

In this lab, you will schedule one-time tasks for the future.

Resources	
Machines:	desktopX

Outcomes

Three jobs scheduled for the future, with one executed, and two removed again.

1. Log into your **desktopX** machine as **student** and open a terminal window.
2. Schedule a task for three minutes in the future. The task should write a timestamp to **/home/student/myjob**.

2.1.

```
[student@desktopX ~]$ echo "date > ~/myjob" | at now +3min
```

3. Inspect the list of tasks scheduled for execution in the future for your user.

3.1.

```
[student@desktopX ~]$ atq
1      Thu Jan 30 05:13:00 2014 a student
```

4. Wait for your job to run, then inspect the contents of **/home/student/myjob**.

- 4.1. Repeatedly run **atq** until your job disappears from the list, or (if you only have one pending job and like scripting):

```
[student@desktopX ~]$ while [ $(atq | wc -l) -gt 0 ]; do sleep 1s; done
```

4.2.

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

5. Schedule a job to run at **16:00** tomorrow, using the **g** queue. This job should create a new file called **/home/student/tea**.

5.1.

```
[student@desktopX ~]$ at -q g teatime tomorrow
at> touch /home/student/tea
at> Ctrl+D
```

6. Schedule a job, this time in the **b** queue, to run at 16:05 tomorrow. This job should create the file **/home/student/cookies**.

6.1.

```
[student@desktopX ~]$ at -q b 16:05 tomorrow
at> touch /home/student/cookies
at> Ctrl+D
```

7. Inspect your pending jobs. Inspect the actual commands your jobs will run as well.

7.1.

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

```
2      Fri Jan 31 16:00:00 2014 g student
3      Fri Jan 31 16:05:00 2014 b student
```

```
7.2. [student@desktopX ~]$ at -c 2
      [student@desktopX ~]$ at -c 3
```

8. You have decided you don't actually like tea that much. Remove the job that writes the file **/home/student/tea**, but keep the job that writes **/home/student/cookies** (you like cookies).

```
8.1. [student@desktopX ~]$ atrm 2
```

Important: If your job to write **/home/student/tea** had a different number than **2**, use that number in the previous command.

Scheduling Recurring Jobs with cron

Objective

After completing this section, students should be able to schedule recurring jobs with **cron**.

Introduction to cron

Using **at**, one could, in theory, schedule a recurring job by having the job resubmit a new job at the end of its execution. In practice, this turns out to be a bad idea. Red Hat Enterprise Linux systems ship with the **crond** daemon enabled and started by default specifically for recurring jobs. **crond** is controlled by multiple configuration files, one per user (edited with the **crontab**(1) command), and systemwide files. These configuration files give users and administrators fine-grained control over exactly when their recurring jobs should be executed. The **crond** daemon is installed as part of the *cronie* package.

If the commands run from a **cron** job produce any output to either **stdout** or **stderr** that is not redirected, the **crond** daemon will attempt to email that output to the user owning that job (unless overridden) using the mail server configured on the system. Depending on the environment this may need additional configuration.

Scheduling jobs

Normal users can use the **crontab** command to manage their jobs. This command can be called in four different ways:

Command	Intended use
crontab -l	List the jobs for the current user.
crontab -r	Remove all jobs for the current users.
crontab -e	Edit jobs for the current user.
crontab <filename>	Remove all jobs, and replace with the jobs read from <filename> . If no file is specified, stdin will be used.



Note

root can use the option **-u <username>** to manage the jobs for another user. It is not recommended to use the **crontab** command to manage system jobs; instead, the methods described in the next section should be used.

Job format

When editing jobs with the **crontab -e**, an editor will be started (**vi** by default, unless the **EDITOR** environment variable has been set to something different). The file being edited will have one job per line. Empty lines are allowed, and comments start their line with a hash symbol (**#**). Environment variables can also be declared, using the format **NAME=value**, and will affect all lines *below* the line where they are declared. Common environment variables in a **crontab** include **SHELL** and **MAILTO**. Setting the **SHELL** variable will change which shell is used to execute the commands on the lines below it, while setting the **MAILTO** variable will change will email address output (if any) will be mailed to.



Important

Sending email may require additional configuration of the local mail server or SMTP relay on a system.

Individual jobs consist of six fields detailing when and what should be executed. When all five of the first fields match the current date and time, the command in the last field will be executed. These fields are (in order):

- Minutes
- Hours
- Day-of-Month
- Month
- Day-of-Week
- Command



Important

When the “Day-of-Month” and “Day-of-Week” fields are both other than *****, the command will be executed when **either** of these two fields match. This can be used, for example, to run a command on the 15th of every month, and every Friday.

The first five of these fields all use the same syntax rules:

- ***** for “Don't Care”/always
- A number to specify a number of minutes or hours, a date, or a weekday. (For weekdays, **0** equals Sunday, **1** equals Monday, **2** equals Tuesday, etc. **7** also equals Sunday.)
- **x-y** for a range, **x** to **y** inclusive
- **x,y** for lists. Lists can include ranges as well, e.g., **5,10-13,17** in the “Minutes” column to indicate that a job should run at 5 minutes past the hour, 10 minutes past, 11 minutes past, 12 minutes past, 13 minutes past, and 17 minutes past.
- ***/x** to indicate an interval of **x**, e.g., ***/7** in the minutes column will run a job exactly every seven minutes.

Additionally, three-letter English abbreviations can be used for both month and weekdays, e.g., Jan, Feb and Tue, Wed.

The last field contains the command to be executed. This command will be executed by **/bin/sh**, unless a **SHELL** environment variable has been declared. If the command contains an unescaped percentage sign (%) that percentage sign will be treated as a newline, and everything after the percentage sign will be fed to the command on **stdin**.

Example cron jobs

Some example **cron** jobs:

- **0 9 2 2 * /usr/local/bin/yearly_backup**

Execute the command **/usr/local/bin/yearly_backup** at exactly 9 a.m. on February 2nd, every year.

- *** /7 9-16 * Jul 5 echo "Chime"**

Send an email containing the word **Chime** to the owner of this job, every seven minutes between 9 a.m. and 5 p.m., on every Friday in July.

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

Run the command **/usr/local/bin/daily_report** every weekday at two minutes before midnight.

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

Every workday (Monday to Friday), at 9 a.m. sharp, send a mail message to **boss@example.com** using **mutt**.



References

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

Practice: Scheduling Recurring Jobs with cron

In this lab, you will schedule a recurring job using **cron**.

Resources

Machines:

desktopX

Outcomes

A recurring job is scheduled, and then removed again.

1. Log into your **desktopX** machine as **student**.
2. Schedule a recurring job that...
 - ...runs as your **student** user.
 - ...runs every two minutes between **09:00** and **16:59** on Monday to Friday.
 - ...appends the current date and time to the file **/home/student/my_first_cron_job**.

- 2.1. Start the **crontab** editor.

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

- 2.2. Insert the following line:

```
*/2 9-16 * * 1-5 date >> /home/student/my_first_cron_job
```

- 2.3. Save your changes and quit the editor (**:wq**).

3. Inspect all of your scheduled **cron** jobs.

- 3.1.

```
[student@desktopX ~]$ crontab -l
```

4. Wait for your job to run at least once or twice, then inspect the contents of the **/home/student/my_first_cron_job** file.

- 4.1.

```
[student@desktopX ~]$ cat ~/my_first_cron_job
```

5. Remove *all* of the **cron** jobs for student.

- 5.1.

```
[student@desktopX ~]$ crontab -r
```

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

In this lab, you will work with recurring system jobs.

Resources	
Files:	<ul style="list-style-type: none"> • <code>/etc/crontab</code> • <code>/etc/cron.d/*</code> • <code>/etc/cron.{hourly,daily,weekly,monthly}/*</code>
Machines:	<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. Save your changes and exit.
- 3.6. Monitor the files in **/var/log/sa** to see when their sizes and timestamps change.

```
[root@desktopX ~]# watch ls -l /var/log/sa
```

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 **rw-r-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

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	
Files:	<ul style="list-style-type: none"> • /etc/tmpfiles.d/ • /usr/lib/tmpfiles.d/tmp.conf
Machines:	serverX

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
```

Chapter Test: Scheduling Future Linux Tasks

Match the descriptions to the relevant **cron** or **at** jobs.

Early on Christmas morning

Every Thursday at 5:00 p.m.

Every Wednesday at 12:30 p.m.

Just after midnight on every Monday and every 1st of the month.

Next Thursday at 5:00 p.m.

Next Wednesday at 12:30 p.m.

Job	Time description
30 6 25 12 * /usr/local/bin/open_presents	
30 12 * * 3 reboot	
0 17 * * 4 rm -rf /home/student	
echo reboot at 12:30 wednesday	
3 0 1 * 1 /sbin/dump 0uf /dev/st0 /home	
echo "userdel -r student" at 17:00 thursday	

Solution

Match the descriptions to the relevant **cron** or **at** jobs.

Job	Time description
30 6 25 12 * /usr/local/bin/open_presents	Early on Christmas morning
30 12 * * 3 reboot	Every Wednesday at 12:30 p.m.
0 17 * * 4 rm -rf /home/student	Every Thursday at 5:00 p.m.
echo reboot at 12:30 wednesday	Next Wednesday at 12:30 p.m.
3 0 1 * 1 /sbin/dump 0uf /dev/st0 /home	Just after midnight on every Monday and every 1st of the month.
echo "userdel -r student" at 17:00 thursday	Next Thursday at 5:00 p.m.

Summary

Scheduling One-Time Tasks with at

- **at** schedules future jobs.
- **atq** lists scheduled jobs.
- **at -c** inspects scheduled jobs.
- **atrm** removes scheduled future jobs.

Scheduling Recurring Jobs with cron

- **crontab -e** edits a user crontab.
- Six columns in a crontab: Minutes, Hours, Day-of-Month, Month, Day-of-Week, and Command.

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 5

MANAGING PRIORITY OF LINUX PROCESSES

Overview	
Goal	To influence the relative priorities at which Linux processes run.
Objectives	<ul style="list-style-type: none">• Describe nice levels.• Set nice levels on new and existing processes.
Sections	<ul style="list-style-type: none">• Process Priorities and "nice" Concepts (and Practice)• Using nice and renice to Influence Process Priority (and Practice)
Lab	<ul style="list-style-type: none">• Managing Priority of Linux Processes

Process Priority and "nice" Concepts

Objectives

After completing this section, students should be able to describe *nice* levels and their effects.

Linux process scheduling and multitasking

Modern computer systems range from low-end processors that can only execute one single instruction at a time to high-performing supercomputers with hundreds of CPUs each and multiple cores on each CPU, performing hundreds of instructions in parallel. But all of these systems tend to have one thing in common: They always need to run more processes than they actually have cores.

The way Linux (and other operating systems) can actually run more processes (and threads) than there are actual processing units available is by employing a technique called *time-slicing*. The operating system *process scheduler* will rapidly switch between processes on a single core, giving a user the impression that there are more processes running at the same time.

The part of the Linux kernel that performs this switching is called the *process scheduler*.

Relative priorities

Since not every process is as important as another one, the scheduler can be told to use different scheduling policies for different processes. The scheduling policy used for most processes running on a regular system is called **SCHED_OTHER** (also called **SCHED_NORMAL**), but there are other policies available for different purposes.

Since not all processes are created equally, processes running with the **SCHED_NORMAL** policy can be given a relative priority. This priority is called the *nice* value of a process, and there are exactly **40** different levels of niceness a process can have.

These nice levels range from **-20** to **19**. By default, processes will inherit their nice level from their parent, which is usually **0**. Higher nice levels indicate less priority (the process easily gives up its CPU usage for others), while lower nice levels indicate a higher priority (the process is less inclined to give up the CPU). If there is no contention for resources—for example, when there are fewer active processes than available CPU cores—even processes with a high nice level will still use all available CPU resources they can. But when there are more processes requesting CPU time than available cores, the processes with a higher nice level will receive less CPU time than those with a lower nice level.



Figure 5.1: Nice levels and how they are reported by top

Nice levels and permissions

Since setting a low nice level on a CPU-hungry process might negatively impact the performance of other processes running on the same system, only **root** (more detailed: users with the

CAP_SYS_NICE capability) is allowed to set negative nice levels and lower the nice level on existing processes.

Regular, unprivileged users are only allowed to set positive nice levels. Furthermore, they are only allowed to *raise* the nice level on their existing process, but cannot *lower* them.



Important

There are more ways to influence process priority and resource usage than just nice levels. There are alternate scheduler policies and settings, *control groups (cgroups)*, and more. Nice levels are, however, the easiest to use, and can be used by regular users as well as system administrators.



References

nice(1) and **sched_setscheduler**(2) man pages

Practice: Process Priority and "nice" Concepts

Match the following items to their description in the table.

-20 - +19	High nice level	Negative nice level
Regular users	root	

Description	Item
These kinds of processes easily give up their CPU resources for others.	
These kinds of processes attempt to keep CPU usage to themselves.	
Cannot assign negative nice levels	
Can renice processes belonging to other users	
The complete range of nice levels	

Solution

Match the following items to their description in the table.

Description	Item
These kinds of processes easily give up their CPU resources for others.	High nice level
These kinds of processes attempt to keep CPU usage to themselves.	Negative nice level
Cannot assign negative nice levels	Regular users
Can renice processes belonging to other users	root
The complete range of nice levels	-20 - +19

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 ksm         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

In this exercise, you will experience the influence that nice levels have on relative process priorities.

Resources	
Machines:	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 -n 10 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 -o pid,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

In this lab, you will search for processes with high CPU consumption and adjust their nice levels.

Resources	
Files:	<code>/usr/local/bin/lab_nice</code>
Machines:	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.
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	
Files:	<code>/usr/local/bin/lab_nice</code>
Machines:	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

Process Priority and "nice" Concepts

- All processes on a Linux system have a relative priority.
- The *nice*ness of a process influences its priority.

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 6

CONTROLLING ACCESS TO FILES WITH ACCESS CONTROL LISTS (ACLs)

Overview	
Goal	To manage file security using POSIX access control lists (ACLs).
Objectives	<ul style="list-style-type: none">• Describe POSIX access control lists.• Manage POSIX access control lists.
Sections	<ul style="list-style-type: none">• POSIX Access Control Lists (ACLs) (and Practice)• Securing Files with ACLs (and Practice)
Lab	<ul style="list-style-type: none">• Controlling Access to Files with Access Control Lists (ACLs)

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 after installation 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* "**rw****x**" flags as:

- **user:** Shows the *user* ACL settings, which are the same as the standard *user* file settings; **rw****x**.
- **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:rw-    #effective:rw-  3
```

- ❶ Group-owner permissions. **controller** has **rw**x, 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 **rw**x, 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::rw-
user:james:---
user:1005:rw-
group::rw-
group:sodor:r-x
group:2210:rw-
mask::rw-
other::---
default:user::rw-
default:user:james:---
default:group::rw-
default:group:sodor:r-x
default:mask::rw-
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** for restoring ACL or for copying ACL from source. For example to restore ACLs from a backup, use **getfacl -R dir1 > file1** to generate a recursive ACL output dump file for the directory and its content. This output can then be used for recovery of original ACLs by passing the saved output and an input to **setfacl** command. For an example: **setfacl --set-file=file1** to do a mass update to the same directory in the current path.

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

Match the following items to their counterparts in the table.

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

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.	
Read, write, execute permissions for a directory granted to a named group.	

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

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

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:	
Files:	<code>/shares/steamies/*, /shares/steamies/display_engines.sh</code>
Machines:	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:----
```

Lab: Controlling Access to Files with Access Control Lists (ACLs)

In this lab, you will update a collaborative directory to have the correct group ownership and permissions. You will add ACLs to allow another group to have appropriate permissions, while limiting permissions for a specific user.

Resources:	
Files:	/shares/cases/*
Machines:	serverX

Outcomes:

- Members of the **bakerstreet** group will have correct access permissions to the **cases** directory.
- Members of the **scotlandyard** group will have read/write access to the **cases** directory, except user **jones**, who only has read access. All members of the **scotlandyard** group should have execute on the directory.
- New files and directories will automatically get the correct group ownership, ACL, and file permissions.

Before you begin

- Reset your serverX system (*see note*).
- Log into and set up your server system (*see note*).

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

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



Note

If you reset your server for the "Using ACLs to Grant and Limit Access" practice exercise and you have not tampered with the **/shares/cases** directory, then you do NOT need to reset the server or rerun the lab setup for this lab.

The Baker Street detective agency is setting up a collaborative share directory to hold case files, which members of the **bakerstreet** group will have read and write permissions on.

The lead detective, Sherlock Holmes, has decided that members of the **scotlandyard** group should also be able to read and write to the share directory. However, Holmes thinks that Inspector Peter Jones (a member of the **scotlandyard** group) is an imbecile, and as such, Jones should have his access to the directory restricted to read-only.

Mrs. Hudson has limited Linux skills and was only able to create the share directory and copy some files into it. Being tea time, she has asked you to complete the job, while she organizes tea and biscuits for Holmes and Watson.

Your task is to complete the setup of the share directory. The directory and all of its contents should be owned by the **bakerstreet** group, with the files updated to read and write for the owner and group (**bakerstreet**). Other users should have no permissions. You also need to provide read and write permissions for the **scotlandyard** group, with the exception of **jones**, who only gets read permissions. Make sure your setup applies to existing and future files.

Important information:

- Share directory: **/shares/cases**
- **bakerstreet** group: **holmes, watson**
- **scotlandyard** group: **lestrade, gregson, jones**
- Two files exist in the directory: **adventures.txt** and **moriarty.txt**.
- All five user passwords are **redhat**.
- All changes should occur to directory **cases** and its files; do not adjust the **shares** directory.

When you are done, run the command **lab acl grade** from your machine to verify your work.

1. The **cases** directory and its content should belong to group **bakerstreet**. New files added in the **cases** directory should automatically belong to the group **bakerstreet**. Existing files should be set to **rw** for user and group. (**Hint:** do not use **setfacl**.)
2. Add ACLs to the **cases** directory (and its contents) that allow members of the **scotlandyard** group to have read/write access on the files and execute on the directory. Restrict user **jones** to read access on the files and execute on the directory.
3. Add ACLs that ensure any new file or directory in the **cases** directory have the correct permissions applied for ALL authorized users and groups.
4. Verify that you have made your ACL and file system changes correctly.

Use **ls** and **getfacl** to review your settings on **/shares/cases**.

From **student**, use **sudo -i -u user** to switch to both **holmes** and **lestrade**. Verify that you can write to a file, read from a file, make a directory, and write to a file in the new directory. Use **ls** to check the new directory permissions and **getfacl** to review the new directory ACLs.

From **student**, use **sudo -i -u jones** to switch users. Try writing to a file (*it should fail*) and try to make a new directory (*it should fail*). As **jones**, you should be able to read from the **adventures.txt** file in the **cases** directory and read from the "test" file written in either of the new directories created by **holmes** and **lestrade**.



Note

The set of tests above are some of the tests you could perform to check that access permissions are correct. You should devise appropriate access validation tests for your environment.

5. When you are done, run the command **lab ac1 grade** from your **serverX** machine to verify your work.

Solution

In this lab, you will update a collaborative directory to have the correct group ownership and permissions. You will add ACLs to allow another group to have appropriate permissions, while limiting permissions for a specific user.

Resources:	
Files:	/shares/cases/*
Machines:	serverX

Outcomes:

- Members of the **bakerstreet** group will have correct access permissions to the **cases** directory.
- Members of the **scotlandyard** group will have read/write access to the **cases** directory, except user **jones**, who only has read access. All members of the **scotlandyard** group should have execute on the directory.
- New files and directories will automatically get the correct group ownership, ACL, and file permissions.

Before you begin

- Reset your serverX system (*see note*).
- Log into and set up your server system (*see note*).

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

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



Note

If you reset your server for the "Using ACLs to Grant and Limit Access" practice exercise and you have not tampered with the **/shares/cases** directory, then you do NOT need to reset the server or rerun the lab setup for this lab.

The Baker Street detective agency is setting up a collaborative share directory to hold case files, which members of the **bakerstreet** group will have read and write permissions on.

The lead detective, Sherlock Holmes, has decided that members of the **scotlandyard** group should also be able to read and write to the share directory. However, Holmes thinks that Inspector Peter Jones (a member of the **scotlandyard** group) is an imbecile, and as such, Jones should have his access to the directory restricted to read-only.

Mrs. Hudson has limited Linux skills and was only able to create the share directory and copy some files into it. Being tea time, she has asked you to complete the job, while she organizes tea and biscuits for Holmes and Watson.

Your task is to complete the setup of the share directory. The directory and all of its contents should be owned by the **bakerstreet** group, with the files updated to read and write for the

owner and group (**bakerstreet**). Other users should have no permissions. You also need to provide read and write permissions for the **scotlandyard** group, with the exception of **jones**, who only gets read permissions. Make sure your setup applies to existing and future files.

Important information:

- Share directory: **/shares/cases**
- **bakerstreet** group: **holmes, watson**
- **scotlandyard** group: **lestrade, gregson, jones**
- Two files exist in the directory: **adventures.txt** and **moriarty.txt**.
- All five user passwords are **redhat**.
- All changes should occur to directory **cases** and its files; do not adjust the **shares** directory.

When you are done, run the command **lab acl grade** from your machine to verify your work.

1. The **cases** directory and its content should belong to group **bakerstreet**. New files added in the **cases** directory should automatically belong to the group **bakerstreet**. Existing files should be set to **rw** for user and group. (**Hint:** do not use **setfacl**.)
 - 1.1. Use **chgrp** to recursively update group ownership on the directory and its contents.

```
[root@serverX ~]# chgrp -R bakerstreet /shares/cases
```

- 1.2. Use **chmod** to update the **setgid** flag on the directory.

```
[root@serverX ~]# chmod g+s /shares/cases
```

- 1.3. Use **chmod** to update all existing file permissions to **rw** for owner and group.

```
[root@serverX ~]# chmod 660 /shares/cases/*
```

2. Add ACLs to the **cases** directory (and its contents) that allow members of the **scotlandyard** group to have read/write access on the files and execute on the directory. Restrict user **jones** to read access on the files and execute on the directory.
 - 2.1. Use **setfacl** to recursively update the existing **cases** directory and its content. Grant the group **scotlandyard** read, write, and conditional execute permissions.

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

- 2.2. Use **setfacl** to recursively update the existing **cases** directory and its content. Grant the user **jones** read and conditional execute permissions.

```
[root@serverX ~]# setfacl -Rm u:jones:rX /shares/cases
```

3. Add ACLs that ensure any new file or directory in the **cases** directory have the correct permissions applied for ALL authorized users and groups.

- 3.1. Use **setfacl** to update the *default* permissions for members of the **scotlandyard** group. Default permissions are read, write, and execute (needed for proper subdirectory creation and access).

```
[root@serverX ~]# setfacl -m d:g:scotlandyard:rwx /shares/cases
```

- 3.2. Use **setfacl** to update the *default* permissions for **scotlandyard** user **jones**. Default permissions are read and execute (needed for proper subdirectory access).

```
[root@serverX ~]# setfacl -m d:u:jones:rx /shares/cases
```

4. Verify that you have made your ACL and file system changes correctly.

Use **ls** and **getfacl** to review your settings on **/shares/cases**.

From **student**, use **sudo -i -u user** to switch to both **holmes** and **lestrade**. Verify that you can write to a file, read from a file, make a directory, and write to a file in the new directory. Use **ls** to check the new directory permissions and **getfacl** to review the new directory ACLs.

From **student**, use **sudo -i -u jones** to switch users. Try writing to a file (*it should fail*) and try to make a new directory (*it should fail*). As **jones**, you should be able to read from the **adventures.txt** file in the **cases** directory and read from the "test" file written in either of the new directories created by **holmes** and **lestrade**.

- 4.1. Use **ls** to check the **cases** directory and its content. Look for group ownership, directory and file permissions, the directory **setgid** flag, and the "+" indicating ACLs exist.

```
[root@serverX ~]# ls -ld /shares/cases
drwxrws---+ 2 root bakerstreet 46 Mar 18 06:56 /shares/cases
[root@serverX ~]# ls -l /shares/cases
total 16
-rw-rw----+ 1 root bakerstreet 22 Mar 18 06:56 adventures.txt
-rw-rw----+ 1 root bakerstreet  8 Mar 18 06:56 do_NOT_delete.grading.txt
-rw-rw----+ 1 root bakerstreet 38 Mar 18 06:56 moriarty.txt
```

- 4.2. Use **getfacl** and review its output. Look for the named user and named group entries in both the standard and default ACLs.

```
[root@serverX ~]# getfacl /shares/cases
getfacl: Removing leading '/' from absolute path names
# file: shares/cases
# owner: root
# group: bakerstreet
# flags: -s-
user::rwx
user:jones:r-x
group::rwx
group:scotlandyard:rwx
mask::rwx
other:---
default:user::rwx
```

```
default:user:jones:r-x
default:group::rwx
default:group:scotlandyard:rwx
default:mask::rwx
default:other:----
```

- 4.3. Perform the following operations as **holmes**. Repeat as **lestrade**, replacing any reference to **holmes** in each of the commands. Check that you get the expected access behavior.

```
[student@serverX ~]$ sudo -i -u holmes
[holmes@serverX ~]$ cd /shares/cases
[holmes@serverX cases]$ echo hello > holmes.txt
[holmes@serverX cases]$ cat adventures.txt
The Adventures of ...
[holmes@serverX cases]$ mkdir holmes.dir
[holmes@serverX cases]$ echo hello > holmes.dir/test.txt
[holmes@serverX cases]$ ls -ld holmes.dir
drwxrws---+ 2 holmes bakerstreet 21 Mar 18 07:35 holmes.dir
[holmes@serverX cases]$ ls -l holmes.dir
total 8
-rw-rw----+ 1 holmes bakerstreet 6 Mar 18 07:39 test.txt
[holmes@serverX cases]$ getfacl holmes.dir
# file: holmes.dir
# owner: holmes
# group: bakerstreet
# flags: -s-
user::rwx
user:jones:r-x
group::rwx
group:scotlandyard:rwx
mask::rwx
other:----
default:user::rwx
default:user:jones:r-x
default:group::rwx
default:group:scotlandyard:rwx
default:mask::rwx
default:other:----

[holmes@serverX cases]$ exit
logout
[student@serverX ~]$
```

- 4.4. Perform the following operations as **jones**. Check that you get the expected access behavior.

```
[student@serverX ~]$ sudo -i -u jones
[jones@serverX ~]$ cd /shares/cases
[jones@serverX cases]$ echo hello > jones.txt
-bash: jones.txt: Permission denied
[jones@serverX cases]$ cat adventures.txt
The Adventures of ...
[jones@serverX cases]$ mkdir jones.dir
mkdir: cannot create directory 'jones.dir': Permission denied
[jones@serverX cases]$ cat holmes.dir/test.txt
hello
[jones@serverX cases]$ exit
logout
```

```
[student@serverX ~]#
```



Note

The set of tests above are some of the tests you could perform to check that access permissions are correct. You should devise appropriate access validation tests for your environment.

5. When you are done, run the command **lab ac1 grade** from your **serverX** machine to verify your work.

5.1.

```
[student@serverX ~]$ lab ac1 grade
```

Summary

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 7

MANAGING SELINUX SECURITY

Overview	
Goal	To manage the Security Enhanced Linux (SELinux) behavior of a system to keep it secure in case of a network service compromise.
Objectives	<ul style="list-style-type: none">• Explain the basics of SELinux permissions.• Change SELinux modes with setenforce.• Change file contexts with semanage and restorecon.• Manage SELinux booleans with setsebool.• Examine logs and use sealert to troubleshoot SELinux violations.
Sections	<ul style="list-style-type: none">• Enabling and Monitoring SELinux (and Practice)• Changing SELinux Modes (and Practice)• Changing SELinux Contexts (and Practice)• Changing SELinux Booleans (and Practice)• Troubleshooting SELinux (and Practice)
Lab	<ul style="list-style-type: none">• Managing SELinux Security

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.

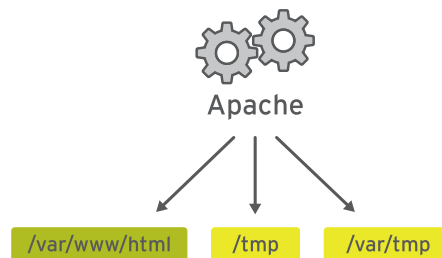


Figure 7.1: Apache service without SELinux protection

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.

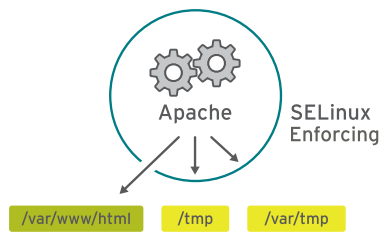


Figure 7.2: Apache service with SELinux protection

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.

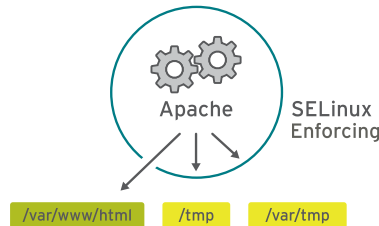


Figure 7.3: SELinux enforcing mode

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.

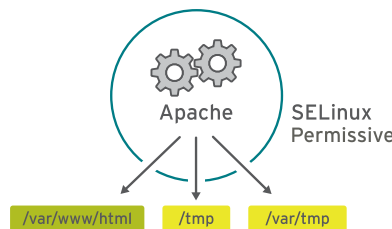


Figure 7.4: SELinux permissive mode

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 ...]
```



Note

Many Boolean names have changed from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7.



References

selinux(8), **getenforce**(8), **ls**(1), **ps**(1), and **getsebool**(8) man pages

Practice: SELinux Concepts

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.

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

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.

Following is a reference table for basic file context operation options of **semanage fcontext** command:

semanage fcontext options to add, remove or list SELinux file contexts

option	description
-a, --add	Add a record of the specified object type
-d, --delete	Delete a record of the specified object type
-l, --list	List records of the specified object type

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:httpd_sys_content_t:s0 /var/www/html/file2
[root@serverX ~]# semanage fcontext -l
...
/var/www(/.*)?      all files      system_u:object_r:httpd_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:httpd_sys_content_t:s0
[root@serverX ~]# ls -Z /var/www/html/file*
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0
    /var/www/html/file1
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0
    /var/www/html/file2
```

The following example shows how to use **semanage** to add a context for a new directory.

```
[root@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 httpd_sys_content_t '/virtual(/.*)?'
[root@serverX ~]# restorecon -RFvv /virtual
[root@serverX ~]# ls -Zd /virtual/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/
```

```
[root@serverX ~]# ls -Z /virtual/  
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 index.html
```



References

chcon(1), **restorecon**(8), **semanage**(8), and **semanage-fcontext**(8) man pages

Practice: Changing SELinux Contexts

In this lab, you will persistently change the SELinux context of a directory and its contents.

Resources	
Files:	<code>/etc/httpd/conf/httpd.conf</code>
Machines:	<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

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	
Files:	<code>/etc/httpd/conf.d/userdir.conf</code>
Machines:	<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 two distinct lines with the **UserDir** directive to read as follows:

```
#UserDir disabled
UserDir public_html
```

```
[root@serverX ~]# vi /etc/httpd/conf.d/userdir.conf
[root@serverX ~]# grep '#UserDir' /etc/httpd/conf.d/userdir.conf
#UserDir disabled
[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>
```

Even though the contents of **file3** are expected, the web server returns a **permission denied** error. Inspecting both **/var/log/audit/audit.log** and **/var/log/messages** can reveal some extra information about this error.

```
[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
```

Both log files indicate that an SELinux denial is the culprit. The **sealert** command detailed in **/var/log/messages** can provide some extra information, including a possible fix.

```
[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

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 ~]# less /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

```

- 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

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 grade** command 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 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 grade** command 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 8

CONNECTING TO NETWORK-DEFINED USERS AND GROUPS

Overview	
Goal	To configure systems to use central identity management services.
Objectives	Use centralized identity management services.
Sections	<ul style="list-style-type: none">• Using Identity Management Services (and Practice)
Lab	<ul style="list-style-type: none">• Connecting to Network-defined Users and Groups

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/openldap/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 *sss* 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.

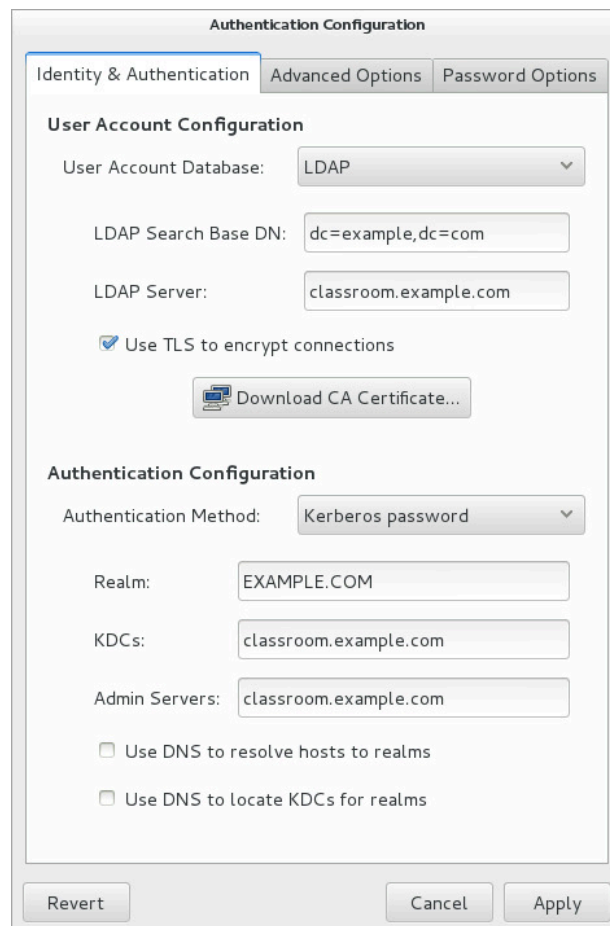


Figure 8.1: Main **authconfig-gtk** window

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.



Warning

Due to a known bug (https://bugzilla.redhat.com/show_bug.cgi?id=1184639) in the **authconfig-6.2.8-8.el7** results in a bogus `krb5_realm` value as `#` in the `/etc/sssd/sssd.conf`. When using **authconfig-gtk**, this issue can be avoided by removing the default `#` value from **Realm** field. Also when using **authconfig** command line, make sure to use `--krb5realm=` option to enter a realm when applicable and verify files `/etc/krb5.conf` and `/etc/sssd/sssd.conf` for broken values.

Red Hat product support has provided an errata version **authconfig-6.2.8-10.el7** which resolves this issue for production environments. Refer: <https://rhn.redhat.com/errata/RHBA-2015-2403.html>

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** 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 ~]$ sudo 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., **ipausers@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 **sss** service.



References

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

Practice: Connecting to a Central LDAP and Kerberos Server

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

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>

- 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
```

- 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 **student** password (**student**) 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: Connecting to Network-defined Users and Groups

In this lab, you will configure your **desktopX** system to become a client of the IPA server running on **serverX**.

Resources:

Machines:

desktopX and **serverX**

Outcomes:

Your **desktopX** system should use the network users and groups defined by the IPA server running on **serverX** for both user information and authentication.

Before you begin

If you haven't already done so at the start of the previous exercise:

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

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

Always perform this step:

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

In your company's quest for a central user information and authentication system, you have settled on using an IPA server for central user management. Another department has already configured an IPA server on your **serverX** machine. This IPA server is configured with all the relevant DNS SRV records for the following settings:

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

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 ipaclient grade** on your **desktopX** machine to verify your work.

1. Install the *ipa-client* package on your **desktopX** machine.
2. 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.
3. 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.
4. Run **lab ipaclient grade** on your **desktopX** machine to verify your work.

Solution

In this lab, you will configure your **desktopX** system to become a client of the IPA server running on **serverX**.

Resources:

Machines:

desktopX and **serverX**

Outcomes:

Your **desktopX** system should use the network users and groups defined by the IPA server running on **serverX** for both user information and authentication.

Before you begin

If you haven't already done so at the start of the previous exercise:

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

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

Always perform this step:

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

In your company's quest for a central user information and authentication system, you have settled on using an IPA server for central user management. Another department has already configured an IPA server on your **serverX** machine. This IPA server is configured with all the relevant DNS SRV records for the following settings:

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

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 ipaclient grade** on your **desktopX** machine to verify your work.

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

1.1.

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

2. 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.

2.1.

```
[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,dc=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.
```

3. 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.

3.1.

```
[student@desktopX ~]$ ssh ipausers@desktopX.example.com
ipausers@desktopX.example.com's password: password
Password expired. Change your password now.
Creating home directory for ipausers.
WARNING: Your password has expired.
You must change your password now and login again!
Changing password for user ipausers.
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 ipausers@desktopX.example.com
ipausers@desktopX.example.com's password: redhat123
Last login: Wed Feb 26 05:19:15 2014 from desktopX.example.com
~sh-4.2$ logout
```

4. Run **lab ipaclient grade** on your **desktopX** machine to verify your work.

4.1.

```
[student@desktopX ~]$ lab ipaclient grade
```

Summary

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 9

ADDING DISKS, PARTITIONS, AND FILE SYSTEMS TO A LINUX SYSTEM

Overview	
Goal	To create and manage disks, partitions, and file systems from the command line.
Objectives	<ul style="list-style-type: none">• Manage simple partitions and file systems.• Manage swap space.
Sections	<ul style="list-style-type: none">• Adding Partitions, File Systems, and Persistent Mounts (and Practice)• Adding and Enabling Swap Space (and Practice)
Lab	<ul style="list-style-type: none">• Adding Disks, Partitions, and File Systems to a Linux System

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):
```

- 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

- 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
```

- 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.
```

- 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.
```

- 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**.

- 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):
```

- 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

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:

Machines:

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 swapspace 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 ~]# swapon /dev/vdb1
[root@serverX ~]# free
            total        used        free      shared    buffers     cached
Mem:      1885252      792116      1093136        17092         692      292096
-/+ buffers/cache:      499328      1385924
Swap:      262140           0      262140

```

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

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 swapspace 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

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{KMGTP}: Enter
Last sector (2048-20971486, default = 20971486) or {+-}size{KMGTP}: +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

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 10

MANAGING LOGICAL VOLUME MANAGEMENT (LVM) STORAGE

Overview	
Goal	To manage logical volumes from the command line.
Objectives	<ul style="list-style-type: none">• Describe logical volume management components and concepts.• Manage logical volumes.• Extend logical volumes.
Sections	<ul style="list-style-type: none">• Logical Volume Management Concepts (and Practice)• Managing Logical Volumes (and Practice)• Extending Logical Volumes (and Practice)
Lab	<ul style="list-style-type: none">• Managing Logical Volume Management (LVM) Storage

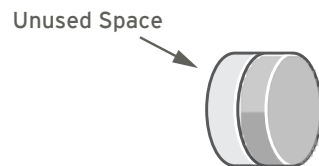
Logical Volume Management Concepts

Objectives

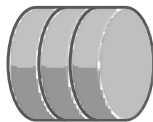
After completing this section, students should be able to describe LVM components.

Logical volume management (LVM) concepts

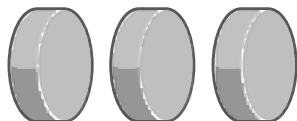
Logical volumes and logical volume management make it easier to manage disk space. If a LVM-hosted file system needs more space, it can be allocated to its logical volume from the free space in its volume group and the file system can be resized. If a disk starts to fail, a replacement disk can be registered as a physical volume with the volume group and the logical volume's extents can be migrated to the new disk.



4. Create logical volume (LV)



3. Create volume group (VG)



2. Create physical volume (PV)



1. Partition physical storage

Figure 10.1: Logical volume management components

LVM Definitions

- *Physical devices* are the storage devices used to persist data stored in a logical volume. These are block devices and could be disk partitions, whole disks, RAID arrays, or SAN disks. A device must be initialized as an LVM physical volume in order to be used with LVM. The entire "device" will be used as a physical volume.
- *Physical volumes (PV)* are used to register underlying physical devices for use in volume groups. LVM automatically segments PVs into *physical extents (PE)*; these are small chunks of data that act as the smallest storage block on a PV.
- *Volume groups (VG)* are storage pools made up of one or more physical volumes. A PV can only be allocated to a single VG. A VG can consist of unused space and any number of logical volumes.
- *Logical volumes (LV)* are created from free physical extents in a volume group and provide the "storage" device used by applications, users, and the operating system. LVs are a collection of *logical extents (LE)*, which map to physical extents, the smallest storage chunk of a PV. By default, each LE will map to one PE. Setting specific LV options will change this mapping; for example, *mirroring* causes each LE to map to two PEs.

Practice: Logical Volume Management Concepts

Match the following items to their counterparts in the table.

Disk, partition, RAID array	Logical extent	Logical volume (LV)
Physical extent	Physical volume (PV)	Volume group (VG)

Component description	Component
Formatted with a file system and mounted for use at runtime	
Maps to a physical storage device, such as a disk or partition	
Storage chunk of a LV, typically maps to a PE	
Used to identify a pool of PVs for use in creating one or more LVs	
Name used for the storage chunk of a PV, also the smallest storage chunk for a LV	
Potential candidates for use as a single PV	

Solution

Match the following items to their counterparts in the table.

Component description	Component
Formatted with a file system and mounted for use at runtime	Logical volume (LV)
Maps to a physical storage device, such as a disk or partition	Physical volume (PV)
Storage chunk of a LV, typically maps to a PE	Logical extent
Used to identify a pool of PVs for use in creating one or more LVs	Volume group (VG)
Name used for the storage chunk of a PV, also the smallest storage chunk for a LV	Physical extent
Potential candidates for use as a single PV	Disk, partition, RAID array

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**, **gdisk** or **parted** 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
```

1 PV Name maps to the device name.

- 2 **VG Name** shows the volume group where the PV is allocated.
- 3 **PV Size** shows the physical size of the PV, including any unusable space.
- 4 **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.

- 5 **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  1
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  2
PE Size                 4.00 MiB

Total PE                253  3
Alloc PE / Size         175 / 700.00 MiB
Free PE / Size           78 / 312.00 MiB  4
VG UUID                 3snNw3-CF71-CcYG-L1k1-p6EY-rHEv-xfUSez
```

- 1 **VG Name** is the name of this volume group.
- 2 **VG Size** is the total size of the storage pool available for logical volume allocation.
- 3 **Total PE** is the total size expressed in PE units.
- 4 **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  1
```

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)**, **pvdisk(8)**, **vgdisplay(8)**, **lvdisplay(8)**, **fdisk(8)**, **gdisk(8)**, **parted(8)**, **partprobe(8)**, and **mkfs(8)** man pages

Practice: Adding a Logical Volume

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**).

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 **pvccreate** to add the two new partitions as PVs.

```
[root@serverX ~]# pvccreate /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-hvXMCM
```

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 vgroup** will show you information about the volume group named **vgroup**.

```
[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**, **gdisk**, or **parted** 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 vgroupname is used to verify that there are sufficient physical extents available for use.

2. Extend the logical volume.

lvextend -l +extents /dev/vgroupname/lvname extends the logical volume **/dev/vgroupname/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), **pvccreate(8)**, **pvmove(8)**, **vgdisplay(8)**, **vgextend(8)**, **vgreduce(8)**, **vgdisplay(8)**, **vgextend(8)**, **vgreduce(8)**, **lvextend(8)**, **fdisk(8)**, **gdisk(8)**, **parted(8)**, **partprobe(8)**, **xfs_growfs(8)**, and **resize2fs(8)** man pages

Practice: Extending a Logical Volume

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

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 697M   21M  677M   3% /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

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 **pvcreeate** to initialize the partition as a PV.

```
[root@serverX ~]# pvcreeate /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

Logical Volume Management Concepts

- Logical volume management building blocks include storage devices, physical volumes made of physical extents, volume groups to pool PVs and host logical volumes, and a suitable file system added to the LV; for example, **xfs** or **ext4**.

Managing Logical Volumes

- **pvccreate**, **pvremove**, and **pvdisplay** 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 11

ACCESSING NETWORK STORAGE WITH NETWORK FILE SYSTEM (NFS)

Overview	
Goal	To use autofs and the command line to mount and unmount network storage with NFS.
Objectives	<ul style="list-style-type: none">• Mount, access, and unmount network storage with NFS.• Automount and access network storage with NFS.
Sections	<ul style="list-style-type: none">• Mounting Network Storage with NFS (and Practice)• Automounting Network Storage with NFS (and Practice)
Lab	<ul style="list-style-type: none">• Accessing Network Storage with Network File System (NFS)

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 7 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

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:	
Files:	<code>nfs_ldapuserX.txt</code> and <code>nfs_student.txt</code>
Machines:	desktopX and serverX

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
```

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**.



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
```

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

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:	
Files:	nfs_ldapuserX.txt
Machines:	desktopX and serverX

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
```

Lab: Accessing Network Storage with Network File System (NFS)

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.
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
```

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**.



CHAPTER 12

ACCESSING NETWORK STORAGE WITH SMB

Overview	
Goal	To use autofs and the command line to mount and unmount SMB file systems.
Objectives	<ul style="list-style-type: none">• Mount, automount, and unmount SMB file systems.
Sections	<ul style="list-style-type: none">• Accessing Network Storage with SMB (and Practice)
Lab	<ul style="list-style-type: none">• Accessing Network Storage with SMB

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

In this lab, you will create a mount entry in **/etc/fstab** and mount it.

Resources:	
Files:	samba.txt in the server directory, for testing.
Machines:	desktopX and serverX

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 SMB

In this lab, you will install packages to support automounting CIFS shares and create three automounts.

Resources:	
Files:	samba.txt in each share directory, for testing.
Machines:	desktopX and serverX

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:	
Files:	samba.txt in each share directory, for testing.
Machines:	desktopX and serverX

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 current group memberships for the **student** user.

```
[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

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 13

CONTROLLING AND TROUBLESHOOTING THE RED HAT ENTERPRISE LINUX BOOT PROCESS

Overview	
Goal	To troubleshoot the Red Hat Enterprise Linux boot process.
Objectives	<ul style="list-style-type: none">• Describe the Red Hat Enterprise Linux boot process.• Repair common boot issues.• Repair file system issues at boot.• Repair boot loader problems.
Sections	<ul style="list-style-type: none">• The Red Hat Enterprise Linux Boot Process (and Practice)• Repairing Common Boot Issues (and Practice)• Repairing File System Issues at Boot (and Practice)• Repairing Boot Loader Issues (and Practice)
Lab	<ul style="list-style-type: none">• Controlling and Troubleshooting the Red Hat Enterprise Linux Boot Process

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
graphical.target	System supports multiple users, graphical and text-based logins.
multi-user.target	System supports multiple users, text-based logins only.
rescue.target	sulogin prompt, basic system initialization completed.
emergency.target	sulogin prompt, initramfs pivot complete and system root mounted on / read-only.

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

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.

1.1.

```
[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.

3.1.

```
[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'
```

3.2.

```
[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.



Note

Pre-built images may place multiple `console=` arguments to the kernel to support a wide array of implementation scenarios. The caveat with `rd.break` is that while many of the kernel messages will be sent to all consoles, the prompt will ultimately use whichever console is last. If you do not get your prompt, you may want to temporarily re-order the `console=` arguments.

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 -m2755 /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

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**.

```
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	systemd will attempt a fsck . If the problem is too serious for an automatic fix, the user will be prompted to run fsck manually from an emergency shell.
Non-existent device/UUID referenced in /etc/fstab	systemd 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 /etc/fstab	systemd creates the mount point if possible; otherwise, it drops to an emergency shell.
Incorrect mount option specified in /etc/fstab	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

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 **systemd.unit=emergency.target** 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

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
      [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
```

Chapter Test: Controlling and Troubleshooting the Red Hat Enterprise Linux Boot Process

The following steps are all performed during the boot process of a Red Hat Enterprise Linux 7 system. Re-order them so that they indicate the order in which they occur.

- ☐ a. A kernel and initramfs are loaded from disk.
- ☐ b. All units for the **default** target are started.
- ☐ c. The boot loader presents the user with a menu.
- ☐ d. The kernel initializes and launches **/sbin/init** from the initramfs.
- ☐ e. The system firmware loads the boot loader.
- ☐ f. The system root file system is mounted read-only on **/sysroot**.
- ☐ g. Basic hardware initialization takes place.
- ☐ h. The root file system is switched, and control is passed over to a new **systemd** instance.
- ☐ i. The boot loader loads its configuration from disk.

Solution

The following steps are all performed during the boot process of a Red Hat Enterprise Linux 7 system. Re-order them so that they indicate the order in which they occur.

- 4 a. A kernel and initramfs are loaded from disk.
- 9 b. All units for the **default** target are started.
- 3 c. The boot loader presents the user with a menu.
- 5 d. The kernel initializes and launches **/sbin/init** from the initramfs.
- 1 e. The system firmware loads the boot loader.
- 7 f. The system root file system is mounted read-only on **/sysroot**.
- 6 g. Basic hardware initialization takes place.
- 8 h. The root file system is switched, and control is passed over to a new **systemd** instance.
- 2 i. The boot loader loads its configuration from disk.

Summary

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 14

LIMITING NETWORK COMMUNICATION WITH FIREWALLD

Overview	
Goal	To configure a basic firewall.
Objectives	<ul style="list-style-type: none">• Configure a basic firewall using <code>firewalld</code>, <code>firewall-config</code>, and <code>firewall-cmd</code>.
Sections	<ul style="list-style-type: none">• Limiting Network Communication (and Practice)
Lab	<ul style="list-style-type: none">• Limiting Network Communication with <code>firewalld</code>

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 **ssh** 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
trusted	Allow all incoming traffic.
home	Reject incoming traffic unless related to outgoing traffic or matching the ssh , mdns , ipp-client , samba-client , or dhcpv6-client pre-defined services.
internal	Reject incoming traffic unless related to outgoing traffic or matching the ssh , mdns , ipp-client , samba-client , or dhcpv6-client pre-defined services (same as the home zone to start with).
work	Reject incoming traffic unless related to outgoing traffic or matching the ssh , ipp-client , or dhcpv6-client pre-defined services.
public	Reject incoming traffic unless related to outgoing traffic or matching the ssh or dhcpv6-client pre-defined services. <i>The default zone for newly-added network interfaces.</i>
external	Reject incoming traffic unless related to outgoing traffic or matching the ssh 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.
dmz	Reject incoming traffic unless related to outgoing traffic or matching the ssh pre-defined service.
block	Reject all incoming traffic unless related to outgoing traffic.
drop	Drop all incoming traffic unless related to outgoing traffic (do not even respond with ICMP errors).

For a list of 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
ssh	Local SSH server. Traffic to 22/tcp
dhcpv6-client	Local DHCPv6 client. Traffic to 546/udp on the fe80::/64 IPv6 network
ipp-client	Local IPP printing. Traffic to 631/udp.
samba-client	Local Windows file and print sharing client. Traffic to 137/udp and 138/udp.
mdns	Multicast DNS (mDNS) local-link name resolution. Traffic to 5353/udp to the 224.0.0.251 (IPv4) or ff02::fb (IPv6) multicast addresses.



Note

Many 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.

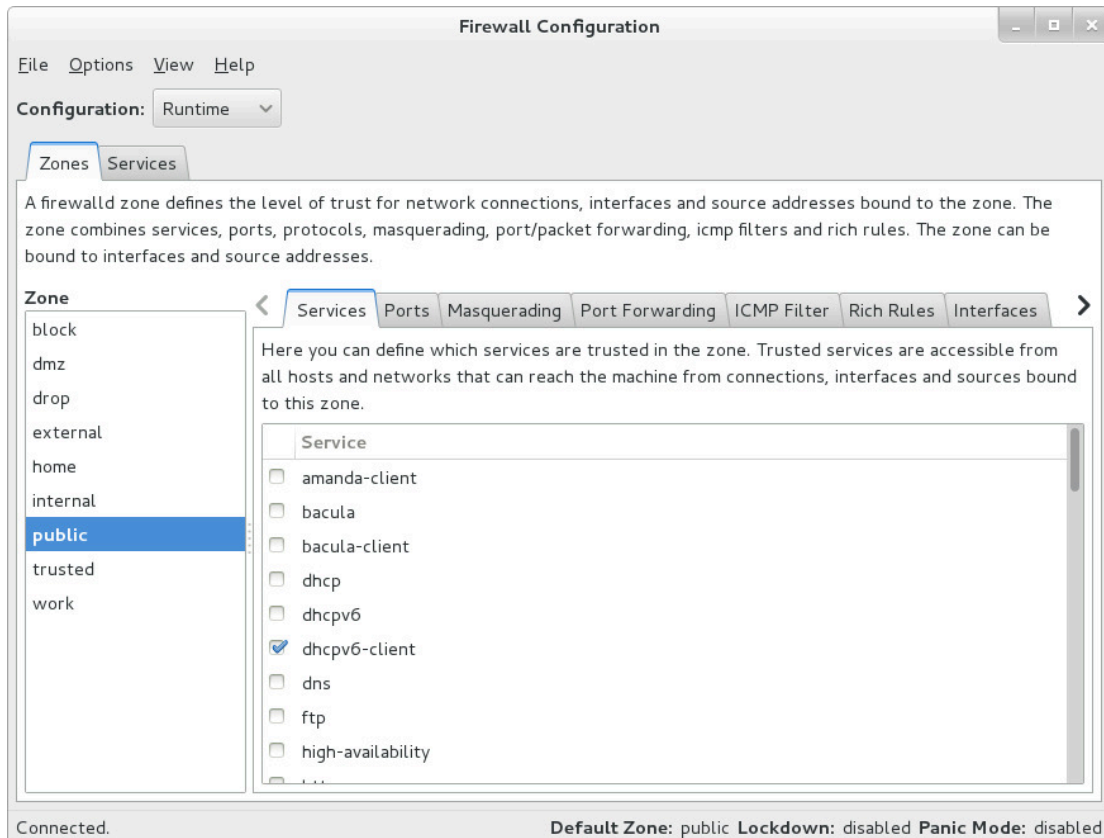


Figure 14.1: The main firewall configuration screen

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.

firewall-cmd commands	Explanation
--get-default-zone	Query the current default zone.
--set-default-zone=<ZONE>	Set the default zone. This changes both the runtime and the permanent configuration.
--get-zones	List all available zones.
--get-active-zones	List all zones currently in use (have an interface or source tied to them), along with their interface and source information.
--add-source=<CIDR> [--zone=<ZONE>]	Route all traffic coming from the IP address or network/netmask <CIDR> to the specified zone. If no --zone= option is provided, the default zone will be used.
--remove-source=<CIDR> [--zone=<ZONE>]	Remove the rule routing all traffic coming from the IP address or network/netmask <CIDR> from the specified zone. If no --zone= option is provided, the default zone will be used.
--add-interface=<INTERFACE> [--zone=<ZONE>]	Route all traffic coming from <INTERFACE> to the specified zone. If no --zone= option is provided, the default zone will be used.
--change-interface=<INTERFACE> [--zone=<ZONE>]	Associate the interface with <ZONE> 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> [--zone=<ZONE>]	Allow traffic to <SERVICE> . If no --zone= option is provided, the default zone will be used.
--add-port=<PORT/PROTOCOL> [--zone=<ZONE>]	Allow traffic to the <PORT/PROTOCOL> port(s). If no --zone= option is provided, the default zone will be used.
--remove-service=<SERVICE> [--zone=<ZONE>]	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> [--zone=<ZONE>]	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

In this lab, you will configure a basic firewall.

Resources

Machines:

serverX and **desktopX**

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** and **vnc** 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 **student** password, enter **student**.

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.
 - 7.3. **Important:** Also add a checkmark in front of the **vnc-server** service. Failing to do so will lock out your graphical interface when you activate the firewall. If you do accidentally lock yourself out, recover by using **ssh -X serverX firewall-config** from your **desktopX** machine.
8. Activate your firewall configuration by selecting **Options > Reload Firewalld** 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

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

Machines:

serverX and **desktopX**

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.

Important: The graphical interface used in the Red Hat Online Learning environment needs port **5900/TCP** to remain available as well. This port is also known under the service name **vnc-server**. If you accidentally lock yourself out from your **serverX**, you can either attempt to recover by using **ssh** to your **serverX** machine from your **desktopX** machine, or reset your **serverX** machine. If you elect to reset your **serverX** machine, you will have to run the setup scripts for this lab again. The configuration on your machines already includes a custom zone called **ROL** that opens these ports.

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

Machines:	serverX and desktopX
------------------	------------------------------------

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.

Important: The graphical interface used in the Red Hat Online Learning environment needs port **5900/TCP** to remain available as well. This port is also known under the service name **vnc-server**. If you accidentally lock yourself out from your **serverX**, you can either attempt to recover by using **ssh** to your **serverX** machine from your **desktopX** machine, or reset your **serverX** machine. If you elect to reset your **serverX** machine, you will have to run the setup scripts for this lab again. The configuration on your machines already includes a custom zone called **ROL** that opens these ports.

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 15

COMPREHENSIVE REVIEW OF SYSTEM ADMINISTRATION II

Overview	
Goal	To practice and demonstrate knowledge and skills learned in Red Hat System Administration II.
Objectives	<ul style="list-style-type: none">• Review the course chapters to reinforce knowledge and skills.
Sections	<ul style="list-style-type: none">• Red Hat System Administration II Comprehensive Review
Lab	<ul style="list-style-type: none">• Comprehensive Review of System Administration II

Red Hat System Administration II

Comprehensive Review

Objectives

After completing this section, students should be able to demonstrate knowledge and skill of the topics covered in each chapter.

Reviewing Red Hat System Administration II

Before beginning the comprehensive review for this course, students should be comfortable with the topics covered in each chapter.

Students can refer to earlier sections in the textbook for extra study.

Chapter 1, Automating Installation with Kickstart

To automate the installation of Red Hat Enterprise Linux systems with Kickstart.

- Explain Kickstart concepts and architecture.
- Create a Kickstart configuration file.

Chapter 2, Using Regular Expressions with grep

To write regular expressions using **grep** to isolate or locate content in text files.

- Create regular expressions to match text patterns.
- Use **grep** to locate content in files.

Chapter 3, Creating and Editing Text Files with vim

Introduce the **vim** text editor.

- Explain the three main modes of **vim**.
- Open, edit, and save text files.
- Use editor shortcuts.

Chapter 4, Scheduling Future Linux Tasks

Schedule tasks to automatically execute in the future.

- Schedule one-time tasks with **at**.
- Schedule recurring jobs with **cron**.
- Schedule recurring system jobs.
- Manage temporary files.

Chapter 5, Managing Priority of Linux Processes

To influence the relative priorities at which Linux processes run.

- Describe nice levels.
- Set nice levels on new and existing processes.

Chapter 6, Controlling Access to Files with Access Control Lists (ACLs)

To manage file security using POSIX access control lists (ACLs).

- Describe POSIX access control lists.
- Manage POSIX access control lists.

Chapter 7, Managing SELinux Security

To manage the Security Enhanced Linux (SELinux) behavior of a system to keep it secure in case of a network service compromise.

- Explain the basics of SELinux permissions.
- Change SELinux modes with `setenforce`.
- Change file contexts with `semanage` and `restorecon`.
- Manage SELinux booleans with `setsebool`.
- Examine logs and use `sealert` to troubleshoot SELinux violations.

Chapter 8, Connecting to Network-defined Users and Groups

To configure systems to use central identity management services.

- Use centralized identity management services.

Chapter 9, Adding Disks, Partitions, and File Systems to a Linux System

To create and manage disks, partitions, and file systems from the command line.

- Manage simple partitions and file systems.
- Manage swap space.

Chapter 10, Managing Logical Volume Management (LVM) Storage

To manage logical volumes from the command line.

- Describe logical volume management components and concepts.
- Manage logical volumes.
- Extend logical volumes.

Chapter 11, Accessing Network Storage with Network File System (NFS)

To use `autofs` and the command line to mount and unmount network storage with NFS.

- Mount, access, and unmount network storage with NFS.
- Automount and access network storage with NFS.

Chapter 12, Accessing Network Storage with SMB

To use `autofs` and the command line to mount and unmount SMB file systems.

- Mount, automount, and unmount SMB file systems.

Chapter 13, Controlling and Troubleshooting the Red Hat Enterprise Linux Boot Process

To troubleshoot the Red Hat Enterprise Linux boot process.

- Describe the Red Hat Enterprise Linux boot process.
- Repair common boot issues.
- Repair file system issues at boot.

- Repair boot loader problems.

Chapter 14, Limiting Network Communication with firewalld

To configure a basic firewall.

- Configure a basic firewall using **firewalld**, **firewall-config**, and **firewall-cmd**.



References

Get information on more classes available from Red Hat at

<http://www.redhat.com/training/>

Lab: Comprehensive Review of System Administration II

In this lab, you will configure a system using the skills taught in this course.

Resources:	
Files:	http://serverX.example.com/logfile
Machines:	serverX and desktopX

Outcomes:

Two systems configured according to the specified requirements that follow.

Before you begin

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

```
[student@serverX ~]$ lab sa2-review setup
```

- Reset your **desktopX** system.

You have been tasked with configuring a new system for your company: **desktopX**. The system should be configured according to the following requirements.

- The system should authenticate users using **LDAP** and **Kerberos** using the following settings:

Name	Value
LDAP server	classroom.example.com
Search Base	dc=example,dc=com
Use TLS	Yes
TLS CA Cert	http://classroom.example.com/pub/example-ca.crt
Kerberos Realm	EXAMPLE.COM
Kerberos KDC	classroom.example.com
Kerberos Admin Server	classroom.example.com

For testing purposes, you can use the user **ldapuserX**, with the password **kerberos**.

- Home directories for your LDAP users should be automatically mounted on access. These home directories are served from the NFS share **classroom.example.com:/home/guests**.
- **serverX** exports a CIFS share called **westeros**. This share should be mounted automatically at boot on the mount point **/mnt/westeros**. To mount this share, you will need to use the username **tyrion** with the password **slapjoffreyslap**. This password should not be stored anywhere an unprivileged user can read it.

- **serverX** exports an NFSv4 share called **/essos**. This share needs to be mounted read-write at boot on **/mnt/essos** using Kerberos authentication, encryption, and integrity checking.

A keytab for your system can be downloaded from **<http://classroom.example.com/pub/keytabs/desktopX.keytab>**.

- Configure a new 512 MiB logical volume called **arya** in a new 2 GiB volume group called **stark**.

This new logical volume should be formatted with an XFS file system, and mounted persistently on **/mnt/underfoot**.

- Your system should be outfitted with a new 512 MiB swap partition, automatically activated at boot.
- Create a new group called **kings**, and four new users belonging to that group: **stannis**, **joffrey**, **renly**, and **robb**.
- Create a new directory **/ironthron**, owned by **root:root** with permissions **700**.

Configure this directory so that users in the **kings** group have both read and write privileges on it, with the exception of the user **joffrey**, who should only be granted read privileges.

These restrictions should also apply to all new files and directories created under the **/ironthron** directory.

- Install the **httpd** and **mod_ssl** packages, then enable and start the **httpd.service** service.
- Open up port **12345/tcp** in the default zone for the firewall running on your system.
- Create a new directory called **/docroot**. Make sure that the SELinux context for this directory is set to **public_content_t**, and that this context will survive a relabel operation.
- **http://serverX.example.com/logfile** contains the logs for a recent project. Download this file, then extract all lines ending in **ERROR** or **FAIL** to the file **/home/student/errors.txt**. All lines should be kept in the order in which they appear in the log file.
- Your system should have a new directory used to store temporary files named **/run/veryveryvolatile**. Whenever **systemd-tmpfiles --clean** is run, any file older than **5** seconds should be deleted from that directory.

This directory should have permissions **1777**, and be owned by **root:root**.

All changes must survive a reboot. When you are done configuring your system, you can test your work by rebooting your **desktopX** machine and running the following command:

```
[student@desktopX ~]$ lab sa2-review grade
```

Solution

In this lab, you will configure a system using the skills taught in this course.

Resources:	
Files:	http://serverX.example.com/logfile
Machines:	serverX and desktopX

Outcomes:

Two systems configured according to the specified requirements that follow.

Before you begin

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

```
[student@serverX ~]$ lab sa2-review setup
```

- Reset your **desktopX** system.

You have been tasked with configuring a new system for your company: **desktopX**. The system should be configured according to the following requirements.

- The system should authenticate users using **LDAP** and **Kerberos** using the following settings:

Name	Value
LDAP server	classroom.example.com
Search Base	dc=example,dc=com
Use TLS	Yes
TLS CA Cert	http://classroom.example.com/pub/example-ca.crt
Kerberos Realm	EXAMPLE.COM
Kerberos KDC	classroom.example.com
Kerberos Admin Server	classroom.example.com

For testing purposes, you can use the user **ldapuserX**, with the password **kerberos**.

- Home directories for your LDAP users should be automatically mounted on access. These home directories are served from the NFS share **classroom.example.com:/home/guests**.
- **serverX** exports a CIFS share called **westeros**. This share should be mounted automatically at boot on the mount point **/mnt/westeros**. To mount this share, you will need to use the username **tyrion** with the password **slapjoffreyslap**. This password should not be stored anywhere an unprivileged user can read it.
- **serverX** exports an NFSv4 share called **/essos**. This share needs to be mounted read-write at boot on **/mnt/essos** using Kerberos authentication, encryption, and integrity checking.

A keytab for your system can be downloaded from **http://classroom.example.com/pub/keytabs/desktopX.keytab**.

- Configure a new 512 MiB logical volume called **arya** in a new 2 GiB volume group called **stark**.

This new logical volume should be formatted with an XFS file system, and mounted persistently on **/mnt/underfoot**.

- Your system should be outfitted with a new 512 MiB swap partition, automatically activated at boot.
- Create a new group called **kings**, and four new users belonging to that group: **stannis**, **joffrey**, **renly**, and **robb**.
- Create a new directory **/ironthron**, owned by **root:root** with permissions **700**.

Configure this directory so that users in the **kings** group have both read and write privileges on it, with the exception of the user **joffrey**, who should only be granted read privileges.

These restrictions should also apply to all new files and directories created under the **/ironthron** directory.

- Install the **httpd** and **mod_ssl** packages, then enable and start the **httpd.service** service.
- Open up port **12345/tcp** in the default zone for the firewall running on your system.
- Create a new directory called **/docroot**. Make sure that the SELinux context for this directory is set to **public_content_t**, and that this context will survive a relabel operation.
- **http://serverX.example.com/logfile** contains the logs for a recent project. Download this file, then extract all lines ending in **ERROR** or **FAIL** to the file **/home/student/errors.txt**. All lines should be kept in the order in which they appear in the log file.
- Your system should have a new directory used to store temporary files named **/run/veryveryvolatile**. Whenever **systemd-tmpfiles --clean** is run, any file older than **5** seconds should be deleted from that directory.

This directory should have permissions **1777**, and be owned by **root:root**.

All changes must survive a reboot. When you are done configuring your system, you can test your work by rebooting your **desktopX** machine and running the following command:

```
[student@desktopX ~]$ lab sa2-review grade
```

1. The system should authenticate users using **LDAP** and **Kerberos** using the following settings:

Name	Value
LDAP server	classroom.example.com
Search Base	dc=example,dc=com
Use TLS	Yes
TLS CA Cert	http://classroom.example.com/pub/example-ca.crt
Kerberos Realm	EXAMPLE.COM
Kerberos KDC	classroom.example.com

Name	Value
Kerberos Admin Server	classroom.example.com

For testing purposes, you can use the user **ldapuserX**, with the password **kerberos**.

- 1.1. Install the *authconfig-gtk* and *sssd* packages.

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

- 1.2. Run **authconfig-gtk**, and enter the information provided. Do not forget to uncheck the **Use DNS to locate KDCs for realms** option.

```
[student@desktopX ~]$ sudo authconfig-gtk
```

2. Home directories for your LDAP users should be automatically mounted on access. These home directories are served from the NFS share **classroom.example.com:/home/guests**.

- 2.1. Install the *autofs* package.

```
[student@desktopX ~]$ sudo yum install autofs
```

- 2.2. Create a new file called **/etc/auto.master.d/guests.autofs** with the following contents:

```
/home/guests /etc/auto.guests
```

- 2.3. Create a new file called **/etc/auto.guests** with the following contents:

```
* -rw, sync classroom.example.com:/home/guests/&
```

- 2.4. Start and enable the **autofs.service** service.

```
[student@desktopX ~]$ sudo systemctl enable autofs.service
[student@desktopX ~]$ sudo systemctl start autofs.service
```

3. **serverX** exports a CIFS share called **westeros**. This share should be mounted automatically at boot on the mount point **/mnt/westeros**. To mount this share, you will need to use the username **tyrion** with the password **slapjoffreyslap**. This password should not be stored anywhere an unprivileged user can read it.

- 3.1. Install the *cifs-utils* package.

```
[student@desktopX ~]$ sudo yum install cifs-utils
```

- 3.2. Create the mount point.

```
[student@desktopX ~]$ sudo mkdir -p /mnt/westeros
```

- 3.3. Create a credentials file named **/root/tyrion.creds** with the following content, then set the permissions on that file to **0600**:

```
username=tyrion
password=slapjoffreyslap
```

```
[student@desktopX ~]$ sudo chmod 0600 /root/tyrion.creds
```

- 3.4. Add the following line to **/etc/fstab**:

```
//serverX.example.com/westeros /mnt/westeros cifs creds=/root/tyrion.creds 0 0
```

- 3.5. Mount all file systems, and inspect the mounted file system.

```
[student@desktopX ~]$ sudo mount -a
[student@desktopX ~]$ cat /mnt/westeros/README.txt
```

4. **serverX** exports an NFSv4 share called **/essos**. This share needs to be mounted read-write at boot on **/mnt/essos** using Kerberos authentication, encryption, and integrity checking.

A keytab for your system can be downloaded from **<http://classroom.example.com/pub/keytabs/desktopX.keytab>**.

- 4.1. Create the mount point.

```
[student@desktopX ~]$ sudo mkdir -p /mnt/essos
```

- 4.2. Download the keytab for your system.

```
[student@desktopX ~]$ sudo wget -O /etc/krb5.keytab http://
classroom.example.com/pub/keytabs/desktopX.keytab
```

- 4.3. Add the following line to **/etc/fstab**:

```
serverX.example.com:/essos /mnt/essos nfs sec=krb5p,rw 0 0
```

- 4.4. Start and enable the **nfs-secure.service** service.

```
[student@desktopX ~]$ sudo systemctl enable nfs-secure.service
[student@desktopX ~]$ sudo systemctl start nfs-secure.service
```

- 4.5. Mount all file systems.


```
[student@desktopX ~]$ sudo mount -a
```

5. Configure a new 512 MiB logical volume called **arya** in a new 2 GiB volume group called **stark**.

This new logical volume should be formatted with an XFS file system, and mounted persistently on **/mnt/underfoot**.

- 5.1. Create a 2 GiB partition on your secondary disk.

```
[student@desktopX ~]$ sudo 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 0xcade6cae.

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): Enter
First sector (2048-20971519, default 2048): Enter
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20971519, default 20971519): +2G
Partition 1 of type Linux and of size 2 GiB is set

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'

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

- 5.2. Turn the new partition into a physical volume.

```
[student@desktopX ~]$ sudo pvcreate /dev/vdb1
```

- 5.3. Build a new volume group using the new physical volume.

```
[student@desktopX ~]$ sudo vgcreate stark /dev/vdb1
```

- 5.4. Create a new 512 MiB logical volume (LV) in the new volume group.

```
[student@desktopX ~]$ sudo lvcreate -n arya -L 512M stark
```

- 5.5. Format the new LV with an XFS file system.

```
[student@desktopX ~]$ sudo mkfs -t xfs /dev/stark/arya
```

5.6. Create the mount point.

```
[student@desktopX ~]$ sudo mkdir -p /mnt/underfoot
```

5.7. Add the following line to **/etc/fstab**:

```
/dev/stark/arya /mnt/underfoot xfs defaults 1 2
```

5.8. Mount all file systems.

```
[student@desktopX ~]$ sudo mount -a
```

6. Your system should be outfitted with a new 512 MiB swap partition, automatically activated at boot.

6.1. Create a new 512 MiB partition on your secondary disk and set the partition type to **82**.

```
[student@desktopX ~]$ sudo 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): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (2-4, default 2): Enter
First sector (4196352-20971519, default 4196352): Enter
Using default value 4196352
Last sector, +sectors or +size{K,M,G} (4196352-20971519, default
20971519): +512M
Partition 2 of type Linux and of size 512 MiB is set

Command (m for help): t
Partition number (1,2, default 2): Enter
Hex code (type L to list all codes): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'

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

6.2. Format the new partition as swap.

```
[student@desktopX ~]$ sudo mkswap /dev/vdb2
```

6.3. Retrieve the UUID for your new swap partition.

```
[student@desktopX ~]$ sudo blkid /dev/vdb2
```

6.4. Add the following line to **/etc/fstab**; make sure to use the UUID you found in the previous step.

```
UUID="xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx" swap swap defaults 0 0
```

6.5. Activate all swaps.

```
[student@desktopX ~]$ sudo swapon -a
```

7. Create a new group called **kings**, and four new users belonging to that group: **stannis**, **joffrey**, **renly**, and **robb**.

7.1. Create the **kings** group.

```
[student@desktopX ~]$ sudo groupadd kings
```

7.2. Create the four users, and add them to the **kings** group.

```
[student@desktopX ~]$ for NEWUSER in stannis joffrey renly robb; do
> sudo useradd -G kings ${NEWUSER}
> done
```

8. Create a new directory **/ironthron**, owned by **root:root** with permissions **0700**.

Configure this directory so that users in the **kings** group have both read and write privileges on it, with the exception of the user **joffrey**, who should only be granted read privileges.

These restrictions should also apply to all new files and directories created under the **/ironthron** directory.

8.1. Create the directory with the correct permissions.

```
[student@desktopX ~]$ sudo mkdir -m 0700 /ironthron
```

8.2. Add an ACL on **/ironthron** granting users in the **kings** group read and write privileges. Do not forget to add execute permissions as well, since this is a directory.

```
[student@desktopX ~]$ sudo setfacl -m g:kings:rwX /ironthron
```

- 8.3. Add an ACL for the user **joffrey**, with only read and execute permissions.

```
[student@desktopX ~]$ sudo setfacl -m u:joffrey:r-x /ironthrone
```

- 8.4. Add the two previous ACLs as default ACLs as well.

```
[student@desktopX ~]$ sudo setfacl -m d:g:kings:rwX /ironthrone
[student@desktopX ~]$ sudo setfacl -m d:u:joffrey:r-x /ironthrone
```

9. Install the *httpd* and *mod_ssl* packages, then enable and start the **httpd.service** service.

- 9.1. Install the *httpd* and *mod_ssl* packages.

```
[student@desktopX ~]$ sudo yum install httpd mod_ssl
```

- 9.2. Start and enable the **httpd.service** service.

```
[student@desktopX ~]$ sudo systemctl start httpd.service
[student@desktopX ~]$ sudo systemctl enable httpd.service
```

10. Open up port **12345/tcp** in the default zone for the firewall running on your system.

- 10.1. Open port **12345/tcp** in the permanent configuration of the default zone for your firewall.

```
[student@desktopX ~]$ sudo firewall-cmd --permanent --add-port=12345/tcp
```

- 10.2. Reload your firewall to activate your changes.

```
[student@desktopX ~]$ sudo firewall-cmd --reload
```

11. Create a new directory called **/docroot**. Make sure that the SELinux context for this directory is set to **public_content_t**, and that this context will survive a relabel operation.

- 11.1. Create the **/docroot** directory.

```
[student@desktopX ~]$ sudo mkdir /docroot
```

- 11.2. Add a new default file context for the **/docroot** directory and all its descendants.

```
[student@desktopX ~]$ sudo semanage fcontext -a -t public_content_t '/docroot(/.*)?'
```

- 11.3. Relabel the **/docroot** directory.

```
[student@desktopX ~]$ sudo restorecon -RvF /docroot
```

12. **http://serverX.example.com/logfile** contains the logs for a recent project. Download this file, then extract all lines ending in **ERROR** or **FAIL** to the file **/home/student/errors.txt**. All lines should be kept in the order in which they appear in the log file.

12.1. Download the log file.

```
[student@desktopX ~]$ wget http://serverX.example.com/logfile
```

- 12.2. Extract every line that ends in either **ERROR** or **FAIL** into the file **/home/student/errors.txt**, while keeping the line order intact.

```
[student@desktopX ~]$ grep -e 'ERROR$' -e 'FAIL$' logfile > /home/student/errors.txt
```

13. Your system should have a new directory used to store temporary files named **/run/veryveryvolatile**. Whenever **systemd-tmpfiles --clean** is run, any file older than 5 seconds should be deleted from that directory.

This directory should have permissions **1777**, and be owned by **root:root**.

- 13.1. Create a new file called **/etc/tmpfiles.d/veryveryvolatile.conf** with the following content:

```
d /run/veryveryvolatile 1777 root root 5s
```

- 13.2. Have **systemd-tmpfiles** create the directory.

```
[student@desktopX ~]$ sudo systemd-tmpfiles --create
```

14. Verify your work by rebooting your **desktopX** machine and running the following command on your **desktopX** system:

```
[student@desktopX ~]$ lab sa2-review grade
```

If any requirement comes up as "FAIL", revisit that requirement, and then reboot and grade again.

Summary

Red Hat System Administration II Comprehensive Review

- Review chapters to validate knowledge level.
- Review practice exercises to validate skill level.